Special Feature 01
本当のクラウドって、こういうことさ

Special Feature 02
ワイヤリングのしくみを知る

Software Design

[ソフトウェア デザイン]
OSとネットワーク、
IT環境を支える
エンジニアの総合誌

2013 March 03

2013年3月18日発行
毎月1回18日発行
通巻335号
（発刊269号）
1985年7月1日
第三種郵便物認可
ISSN 0916-6297
定価
1,280円

もっとクラウド

オープン環境で
スキルアップ!

を活用してみませんか?

Special Feature 01

OpenStack
CloudStack
Cloud Foundry
Scalr
Eucalyptus

NEW GENERATION CLOUD

Special Feature 02
光、ギガビット、高速ネットワークを体験!
実践!ワイヤリングの教科書

Extra Feature 01
ハードディスクが要らなくなる日は近い?
「SSDストレージ」爆発的普及の理由

Extra Feature 02
サーバマシンがフロアからなくなる?
社内LAN撲滅運動!

この個所は、雑誌発行時には広告が掲載されていました。編集の都合上、
総集編では収録致しません。

この個所は、雑誌発行時には広告が掲載されていました。編集の都合上、
総集編では収録致しません。

この個所は、雑誌発行時には広告が掲載されていました。編集の都合上、
総集編では収録致しません。

第51回

IT エンジニア必須の 最新用語解説

TEXT：㈲オングス 杉山 貴章 SUGIYAMA Takaaki
takaaki@ongs.co.jp

テーマ募集
本連載では、最近気になる用語など、今後取り上げてほしいテーマを募集しています。sd@gihyo.co.jp 宛にお送りください。

HipHop VM

「HipHop VM」とは

「HipHop Virtual Machine（HipHop VM）」は、Facebook 社が開発を進めている PHP 用の高速実行環境です。Facebook ではシステムの実装の大部分で PHP が使われていますが、爆発的なユーザ数の増加に既存の PHP の枠組みでは対応し切れないという問題を抱えており、そのためにアプリケーション実行環境の高速化に向けた独自の取り組みを進めてきました。その成果物のひとつが HipHop VM です。

同社による PHP アプリケーションの高速化に対する取り組みは、2010 年より「HipHop for PHP」として進められてきました。HipHop for PHP では、大きく分けて 2 通りのアプローチによる実装が行われました。

ひとつが PHP コードをいったん C++ コードに変換し、そこからバイナリコードを生成する「HipHop PHP-to-C++ compiler（HPHPc）」です。生成されたバイナリコードはネイティブ実行が可能なため、通常の PHP インタプリタに比べて圧倒的に高速な実行を実現できます。その反面、静的なコード解析しか行うことができないことや、事前コンパイルが必要なために PHP の手軽さが損なわれるといったデメリットがありました。

もうひとつのアプローチが独自の PHP インタプリタ実装である「HipHop PHP interpreter（HPHPi）」の開発で、HPHPc によって生成されたバイナリコードに比べれば低速なものの、PHP の手軽さを損なうことなく、従来よりも速いアプリケーション実行を実現しています。

同社では社内のソフトウェア開発には HPHPi を使用し、外部向けの本番環境に配備する際には HPHPc によるバイナリコードを使用するといった形でこの 2 つの実装を使い分けてきたそうです。

実行速度向上へのチャレンジ

HipHop VM は、この両者の良い部分を組み合わせた開発・実行環境を構築する目的で生まれました。HipHop VM では、PHP コードから動的にバイナリコードを生成しながら実行するエンジンを備えているため、HPHPc のように事前コンパイルを行わなくてもネイティブ環境による高速なアプリケーション実行が実現できます。

具体的には、まず読み込んだ PHP コードから抽象構文木を構築します。さらに、その抽象構文木をもとにして HipHop バイトコード（HHBC）と呼ばれる中間コードを生成します。HipHop VM はこの HHBC を実行するインタプリタマシンとして動作します。インタプリタには JIT コンパイル手法が導入されており、コード実行中に動的に最適化を行うことで高いパフォーマンスを実現しているとのことです。PHP コード自体は標準の言語仕様に基づいて記述できるため、既存の PHP コードでも実行環境を置き換えるだけで高速化を実現できる点が大きな強みと言えます。

HipHop VM が最初に公開されたのは 2011 年 12 月のことですが、当初の課題は実行速度が現行の HPHPc の 0.6 倍程度に留まるという点でした。その後、JIT の改善を中心としたチューニングが進められ、2012 年 11 月には www.facebook.com のホームページの生成速度において HPHPc の処理速度を追い抜いたことが発表されました。この成果は、HPHPi と HPHPc の組み合わせによる開発プロセスから脱却し、将来的には HipHop VM に一本化するという当初の目的に大きく近づいたことを意味しています。

今後はさらなるパフォーマンスの向上を目指すとともに、HipHop VM を活用して社内の PHP 開発スタックの最適化を追求し、Facebook の Web 層をより効率的にしていくとのことです。また、HipHop VM のインストールや開発ワークフローをより簡単かつ柔軟にすることや、一般的な PHP フレームワークのサポートを強化することによって、HipHop による開発をより容易なものにしていく方針が発表されています。

HipHop VM はオープンソースで開発が進められており、ソースコードは GitHub リポジトリより入手することができます。PHP アプリケーションの高速化を目指す開発者にとっては大いに試す価値のあるプロダクトだと言えるでしょう。SD

HipHop for PHP
http://www.hiphop-php.com/wp/
GitHub リポジトリ
https://github.com/facebook/hiphop-php/wiki

この個所は、雑誌発行時には広告が掲載されていました。編集の都合上、
総集編では収録致しません。

オープン環境でスキルアップ!
もっとクラウドを活用してみませんか?

OpenStack, CloudStack, Cloud Foundry, Scalr, Eucalyptus

Part 1 Contents
NEW GENERATION CLOUD

この個所は、雑誌発行時には広告が掲載されていました。編集の都合上、
総集編では収録致しません。

Software Design Contents

第2特集 Special Feature 02

光、ギガビット、高速ネットワークを体験!

実践!ワイヤリングの教科書 069

一般記事 Article

巻頭Editorial PR Editorial PR

アラカルト A La Carte

広告索引 AD INDEX

広告掲載企業への詳しい資料請求は、本誌Webサイトからご応募いただけます。> http://sd.gihyo.jp/

Part 02 Contents

この個所は、雑誌発行時には広告が掲載されていました。編集の都合上、
総集編では収録致しません。

［目次］

Software Design March 2013 [CONTENTS] #03

Software Design Contents

Column

Development

OS/Network

Inside View

Part 93 Contents

Logo Design ロゴデザイン ＞ デザイン集合ゼブラ+坂井 哲也
Cover Design 表紙デザイン ＞ Re:D
Cover Photo 表紙写真 ＞ (c) Tetsuya Tanooka/a.collectionRF/amanaimages
Page Design 本文デザイン ＞ 岩井 栄子、近藤 しのぶ、SeaGrape、安達 恵美子
［トップスタジオデザイン室］轟木 亜紀子、阿保 裕美、佐藤 みどり
［BUCH+］伊勢 歩、横山 慎昌
森井 一三、Re:D、［マップス］石田 昌治

この個所は、雑誌発行時には記事が掲載されていました。編集の都合上、
総集編では収録致しません。

この個所は、雑誌発行時には記事が掲載されていました。編集の都合上、
総集編では収録致しません。

この個所は、雑誌発行時には記事が掲載されていました。編集の都合上、
総集編では収録致しません。

この個所は、雑誌発行時には記事が掲載されていました。編集の都合上、
総集編では収録致しません。

見やすい大判！タイプ別にくわしく、どこよりもわかりやすく解説！

マネして書くだけ 確定申告

平成25年3月締切分

税理士 山本宏　監修
A4 判／192 ページ　定価 1,449 円（1,380 円＋税）
ISBN 978-4-7741-5378-0

こんな人にオススメ！

還付金で得したい人、医療費、住宅購入で出費があった人などこれ１冊で大丈夫。
はじめてでもすぐにできる、確定申告の手続きガイド。マネして書くだけで、そのままちゃんと確定申告。今年の改正点もわかりやすく解説！

おかげさまで改訂7版！まったくの初心者でも安心！

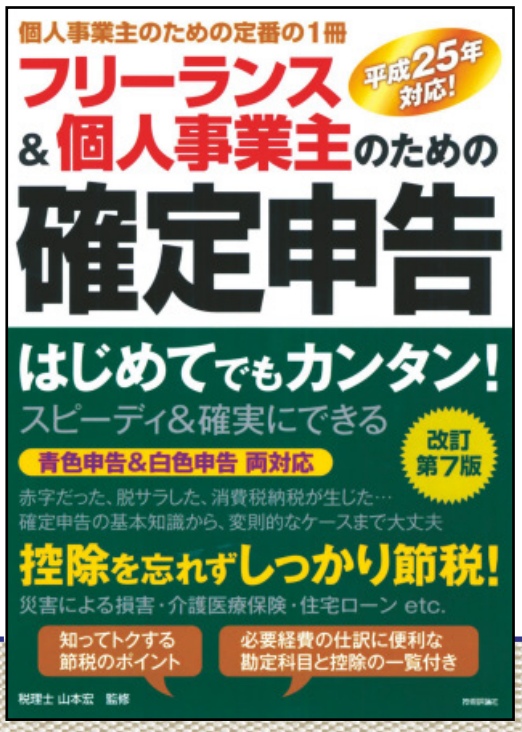
フリーランス＆個人事業主のための 確定申告

改訂第7版

税理士 山本宏　監修
A5 判／240 ページ　定価 1,554 円（1,480 円＋税）
ISBN 978-4-7741-5341-4

こんな人にオススメ！

フリーで商売をしている人、個人事業主、独立したばかりの人。
「青色申告・白色申告」両対応！はじめての方でも迷わず確定申告できます。控除を忘れずしっかり節税するためのノウハウ満載！



技術評論社　当社書籍・雑誌のご購入は全国の書店、またはオンライン書店でお願い致します。
162-0846　東京都新宿区市谷左内町 21-13　販売促進部　TEL.03-3513-6150　FAX.03-3513-6151

定平誠、須藤智　著
A5 判／ 544 ページ
定価 1,764 円（1,680 円+税）
ISBN 978-4-7741-5386-5

栢木厚　著
A5 判／ 456 ページ
定価 1,869 円（1,780 円+税）
ISBN 978-4-7741-5398-8

きたみりゅうじ　著
A5 判／ 656 ページ
定価 2,079 円（1,980 円+税）
ISBN 978-4-7741-5437-4

山本三雄　著
B5 判／ 544 ページ
定価 1,974 円（1,880 円+税）
ISBN 978-4-7741-5446-6

あなたを合格へと導く一冊があります！

大滝みや子、岡嶋裕史　著
A5 判／ 728 ページ
定価 3,129 円（2,980 円+税）
ISBN 978-4-7741-5351-3

加藤昭、芦屋広太、矢野龍王　著
B5 判／ 464 ページ
定価 2,079 円（1,980 円+税）
ISBN 978-4-7741-5352-0

岡嶋裕史　著
A5 判／ 656 ページ
定価 3,129 円（2,980 円+税）
ISBN 978-4-7741-5355-1

エディフィストラーニング株式会社　著
B5 判／ 392 ページ
定価 3,129 円（2,980 円+税）
ISBN 978-4-7741-5356-8

山平耕作　著
A5 判／ 704 ページ
定価 3,465 円（3,300 円+税）
ISBN 978-4-7741-5353-7

エディフィストラーニング株式会社　著
B5 判／ 484 ページ
定価 3,360 円（3,200 円+税）
ISBN 978-4-7741-5321-6

村松倫明　著
A5 判／ 544 ページ
定価 3,360 円（3,200 円+税）
ISBN 978-4-7741-5354-4

内田保男 他　著
A5 判／ 504 ページ
定価 3,360 円（3,200 円+税）
ISBN 978-4-7741-4944-8

技術評論社
当社書籍・雑誌のご購入は全国の書店、またはオンライン書店でお願い致します。
162-0846　東京都新宿区市谷左内町 21-13　販売促進部　TEL.03-3513-6150　FAX.03-3513-6151

紙面版
A4判・16頁
オールカラー

電脳会議

DENNOUKAIGI

一切無料

新規購読会員受付中!

『電脳会議』は情報の宝庫、世の中の動きに遅れるな!

『電脳会議』は、年6回の不定期刊行情報誌です。A4判・16頁オールカラーで、弊社発行の新刊・近刊書籍・雑誌を紹介しています。この『電脳会議』の特徴は、単なる本の紹介だけでなく、著者と編集者が協力し、その本の重点や狙いをわかりやすく説明していることです。平成17年に「通巻100号」を超え、現在200号に迫っている、出版界で評判の情報誌です。

今が旬の情報満載!

新規送付のお申し込みは…

Web検索か当社ホームページをご利用ください。
Google、Yahoo!での検索は、

電脳会議事務局　検索

当社ホームページからの場合は、
https://gihyo.jp/site/inquiry/dennou
と入力してください。

一切無料!

◉「電脳会議」紙面版の送付は送料含め費用は一切無料です。そのため、購読者と電脳会議事務局との間には、権利&義務関係は一切生じませんので、予めご了承下さい。

毎号、厳選ブックガイドも付いてくる!!

『電脳会議』とは別に、1テーマごとにセレクトした優良図書を紹介するブックカタログ（A4判・4頁オールカラー）が2点同封されます。扱われるテーマも、自然科学／ビジネス／起業／モバイル／素材集などなど、弊社書籍を購入する際に役立ちます。

株式会社 技術評論社　電脳会議事務局　〒162-0846 東京都新宿区市谷左内町21-13

スマートフォンアプリ開発者とデザイナのための総合情報誌

Smartphone Design

Software Design 編集部　編
B5 判　168 ページ　定価 1,659 円（本体 1,580 円＋税）
ISBN 978-4-7741-5335-3

ユーザから支持されるスマートフォンアプリ開発のために「デザインする力」がより一層求められている開発現場。悩めるエンジニアとデザイナがともに力を発揮するためのヒントが満載です！

第1特集　どうしてデザインと開発は両立できないのか？
第2特集　スマホ開発者がUnityを理解しておくべき理由
その他　Web＋ネイティブでスピード・コスト・メンテに効くアプリ開発／Windows Phoneアプリ開発入門／PlayStation Mobileアプリ開発／仮想化技術でスマホ＆タブレットを業務に／既存のPCサイトをスマホ用に変換！／Webブラウザでクロス開発できるappMobi　ほか

現場で使える [逆引き+実践] Androidプログラミングテクニック

石原正樹、松尾源、磯村禎孝、森靖晃、奥谷修治　著
A5 判　464 ページ　定価 2,919 円（本体 2,780 円＋税）
ISBN 978-4-7741-5187-8

普及が進むスマートフォンで注目されるAndroid OSですが、組込みシステムの宿命とも言うべき「リソースの制限、バッテリー駆動」といった、プログラミングに関わる制限事項が多数存在します。また、「新しい情報をリアルタイムで追随しにくい」といった問題、さらには「従来型のC/C++の組込み開発をしてきた人や会社はJavaに不慣れ」「Javaに慣れた人や会社は低レベルの理解が足りず参入に苦労」といったノウハウ不足の問題に対し、本書は徹底的にチューニングの方法を解説することでも寄与します。

iPhoneアプリ開発塾

iOS 5.1 & Xcode 4.3 対応

カワサキタカシ　著
B5 変形判　320 ページ　定価 2,919 円（本体 2,780 円＋税）
ISBN 978-4-7741-5105-2

iPhoneアプリの作り方について書かれた本は数多くありますが、「基本はわかっても応用がきかない」「やっぱりiOSプログラミングは難しい」といった声が多いようです。本書は、そんな悩めるiPhoneアプリ開発エンジニア達に人気のポータルサイト『サルでき.jp』（旧ブログ：サルにもできるiPhoneアプリの作り方）の管理人が、Xcodeの読み方、プログラミングの基本はもちろん、サポートページの作り方までを、“どこよりも敷居の低い”書き方で丁寧に解説しています。

技術評論社　当社書籍・雑誌のご購入は全国の書店、またはオンライン書店でお願い致します。
162-0846　東京都新宿区市谷左内町 21-13　販売促進部　TEL.03-3513-6150　FAX.03-3513-6151

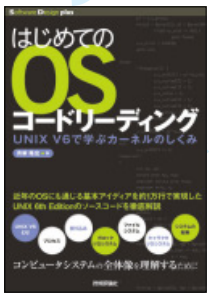
Software Design plus

最新刊！

Software Design plusシリーズは、OSとネットワーク、IT環境を支えるエンジニアの総合誌『Software Design』編集部が自信を持ってお届けする書籍シリーズです。

サウンドプログラミング入門
青木 直史 著
A5判・288ページ
定価 2,980円(本体)+税
ISBN 978-4-7741-5522-7

クラウド時代のネットワーク技術 OpenFlow実践入門
高宮 安仁、鈴木 一哉 著
A5判・336ページ
定価 3,200円(本体)+税
ISBN 978-4-7741-5465-7



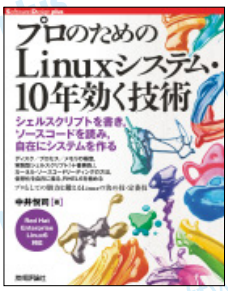
はじめてのOSコードリーディング UNIX V6で学ぶカーネルのしくみ
青柳 隆宏 著
A5判・448ページ
定価 3,200円(本体)+税
ISBN 978-4-7741-5464-0

プロになるためのJavaScript入門
河村 嘉之、川尻 剛 著
B5変形判・480ページ
定価 2,980円(本体)+税
ISBN 978-4-7741-5438-1

Webサービスのつくり方
和田 裕介 著
A5判・208ページ
定価 2,180円(本体)+税
ISBN 978-4-7741-5407-7

Mapion・日本一の地図システムの作り方
㈱マピオン、山岸 靖典、谷内 栄樹、本城 博昭、長谷川 行雄、中村 和也、松浦 慎平、佐藤 亜矢子 著
B5変形判・256ページ
定価 2,580円(本体)+税
ISBN 978-4-7741-5325-4

図解でわかるAndroidアプリケーション開発教科書
三苫 健太 著
B5判・400ページ
定価 3,200円(本体)+税
ISBN 978-4-7741-5189-2



プロのためのLinuxシステム・10年効く技術
中井 悦司 著
B5変形判・352ページ
定価 3,400円(本体)+税
ISBN 978-4-7741-5143-4

サーバ/インフラエンジニア監視管理編 養成読本
Software Design編集部 編
B5判・200ページ
定価 1,980円(本体)+税
ISBN 978-4-7741-5037-6

サーバ/インフラエンジニア仮想化活用編 養成読本
Software Design編集部 編
B5判・200ページ
定価 1,980円(本体)+税
ISBN 978-4-7741-5038-3

もっと自在にサーバを使い倒す 業務に役立つPerl
木本 裕紀 著
A5判・352ページ
定価 2,780円(本体)+税
ISBN 978-4-7741-5025-3

サーバ構築の実際がわかるApache[実践]運用/管理
鶴長 鎮一 著
A5判・344ページ
定価 2,980円(本体)+税
ISBN 978-4-7741-5036-9

プロになるためのデータベース技術入門 MySQL for Windows
木村 明治 著
B5変形判・416ページ
定価 3,180円(本体)+税
ISBN 978-4-7741-5026-0

Webエンジニアのためのデータベース技術[実践]入門
松信 嘉範 著
A5判・336ページ
定価 2,580円(本体)+税
ISBN 978-4-7741-5020-8

2週間でできる! スクリプト言語の作り方
千葉 滋 著
定価 2,580円+税　ISBN 978-4-7741-4974-5

PCのウイルスを根こそぎ削除する方法
本城 信輔 著
定価 1,980円+税　ISBN 978-4-7741-4867-0

Androidエンジニア養成読本
Software Design編集部 編
定価 1,880円+税　ISBN 978-4-7741-4859-5

Vyatta入門 実践ルーティングから仮想化まで
近藤 邦昭、松本 直人、浅間 正和、大久保 修一(日本Vyattaユーザー会) 著
定価 3,200円+税　ISBN 978-4-7741-4711-6

プロのためのLinuxシステム・ネットワーク管理技術
中井 悦司 著
定価 2,880円+税　ISBN 978-4-7741-4675-1

サーバ/インフラエンジニア養成読本
Software Design編集部 編
定価 1,880円+税　ISBN 978-4-7741-4600-3

Linuxエンジニア養成読本
Software Design編集部 編
定価 1,880円+税　ISBN 978-4-7741-4601-0

Nagios統合監視[実践]リファレンス
㈱エクストランス　佐藤 省吾、Team-Nagios 著
定価 3,200円+税　ISBN 978-4-7741-4582-2

プロのためのLinuxシステム構築・運用技術
中井 悦司 著
定価 2,880円+税　ISBN978-4-7741-4501-3

サーバ構築の実例がわかるSamba[実践]入門
髙橋 基信 著
定価 2,480円+税　ISBN 978-4-7741-4405-4

間違いだらけのソフトウェア・アーキテクチャ
トム・エンゲルバーグ 著
長谷川 裕一、土岐 孝平 訳
定価 1,980円+税　ISBN 978-4-7741-4343-9

よくわかるAmazon EC2/S3入門
五十嵐 学、深海 寛信、馬場 俊彰、藤崎 正範 著
定価 2,580円+税　ISBN 978-4-7741-4284-5

Zabbix統合監視[実践]入門
寺島 広大 著
定価 3,500円+税　ISBN 978-4-7741-4213-5

技術評論社　162-0846 東京都新宿区市谷左内町21-13
販売促進部　TEL 03-3513-6150 FAX 03-3513-6151

この個所は、雑誌発行時には広告が掲載されていました。編集の都合上、
総集編では収録致しません。

この個所は、雑誌発行時には広告が掲載されていました。編集の都合上、
総集編では収録致しません。

DIGITAL GADGET

Volume 171

安藤 幸央 —— Yukio Ando ——
EXA CORPORATION
[Twitter] >> @yukio_andoh
[Web Site] >> http://www.andoh.org/

CES2013で家電三昧、ラスベガスでみかけたデジタルガジェット

ヘルスケアグッズ、ネット家電、スマホ周辺機器が台頭、世界最大の家電見本市CES

International CES 2013
http://www.cesweb.org/

世界最大の家電見本市、International CES（Consumer Electronics Show）2013が米国ラスベガスで2013年1月8日～11日の4日間開催されました。

今年はMicrosoftやAppleの展示がないにもかかわらず、全体で3,300社と例年以上の出展者数、そして15万人以上の参加者という巨大な展示会となりました。とくに、iLoungeと銘打ったiPhone/iPad周辺機器を展示する区画は昨年の数倍にもふくれあがり、ケース、ヘッドフォン、外部スピーカー、スタンドなど、スマートフォンやタブレットそのものの販売以上に、周辺機器の大きな市場が構築されていることがうかがえます。

CESでの「スマート」デバイスの様相は昨年、一昨年とは大きく異なりました。従来は「スマート」といえばスマートフォンやスマートTVのように、ネットにつながっているという意味でした。今年の「スマート」はというと、インテリジェントに動作し、人のことを考え、洗練されたものでした。つまり英語本来の意味のスマートなものを示すようになった印象です。

今回のCES 2013で躍進がみてとれたのは、ヘルスケア関連、車載機器、超高解像度4Kテレビ、モバイルアプリやサービスです。とくにモバイル系では生活の隅々にまでFacebook、Twitter、Pinterestが普通に浸透していることがわかりました。逆に展示会場で勢いがなくなったと思われるのは、電子書籍関連デバイス、立体視関連の表示装置でした。電子書籍は全体として市場は伸びつつあるのですが、Amazon Kindleが主流で、その他の端末が何種類かといった状況のようです。

主流商品は日本のニュースなどで取り上げられているものも多いので、ここではちょっと変わった珍しいものを中心に紹介していきましょう。

▲Core-Apps社製イベントスケジューリングアプリのCES 2013専用のもの。今年の新機能は巨大な展示会場で役立つ、今居る場所を把握して目的のブースまでの道のりが示される機能

◀
左／会場入り口に設置される恒例の巨大なCESのロゴ
右／CES会場となったラスベガスコンベンションセンター

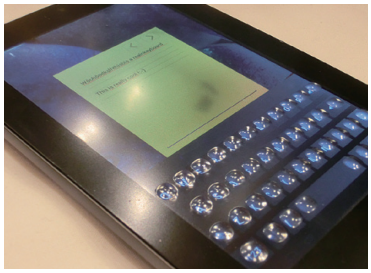
gadget 1

Tactus

http://www.tactustechnology.com

浮き出る物理キーボード

Tactus Technologyのキーボードはタッチスクリーン上に、実際に凸凹したキーボードを出現させる技術です。キー入力時には梱包材のプチプチのようなキーの突起が出現し、キーが必要ないときは瞬時に平面に戻ります。RIMのBlackBerryにある小さな物理キーボードのように、キーを実際に手探りしながら押す感覚が得られます。Tactus自身では製品化の予定はなく、OEM提供で他社のデジタルカメラやリモコン、ゲームパッドなどに搭載する話が進んでいるそうです。凹凸は自由に出現するのではなく、あらかじめソフトウェアキーボードの出現する位置が膨らむよう製作されています。



gadget 2

Flower Power

http://www.parrot.com/flower-power/

植物育成環境監視デバイス

ParrotのFlower PowerはiPhone/iPadアプリとBluetoothで連動する、植物の育成補助用デジタルデバイスです。日照時間、気温、水分量などを15分ごとにセンシングし、植物の種類を登録しておくと、連携アプリに最適な環境条件をアドバイスしてもらえます。観察対象として、6千種類ほどの花、草木、野菜などの栽培方法が用意されているそう。単四乾電池1本で動作し、半年そのままで使い続けられる省エネ設計です。

gadget 3

Nectar

http://www.nectarpower.com/

安全な燃料電池

多くのデジタルガジェットを持ち歩く人にとって、電源の確保は死活問題です。Nectar Mobile Power Systemは燃料電池カートリッジ式のUSBバッテリーです。本体は299.99ドルで7月発売予定で、ブタンを混合した固体酸化物型燃料電池の円筒状のカートリッジ（予価9.99ドル／個）を採用。1個の燃料電池カートリッジで、一般的なスマートフォンを10回から14回ほど満充電できます。たいていの燃料電池は危険物扱いですが、Nectarは航空機に持ち込める許可が得られているそう。再充電こそできませんが、カートリッジを多数持ち歩けば充電せずとも電源が確保し続けられる点が良いところです。

gadget 4

Innovis

http://www.brother.co.jp/product/hsm/embropc/5000/

技巧派デジタルミシン

日本のミシンメーカー、ブラザーからは超絶技巧のデジタルミシンが展示されていました。タッチパネル操作により画像ファイルを取り込み、糸の色や縫い方を調整したうえで、人間業では不可能と思えるような精巧な刺繍が実現できるミシンです。刺繍デザイン用の専用パソコンソフトも用意されています。一度に刺繍可能な面積は縦26cm、横16cm。会場では刺繍とは思えない絵画のような作品が展示されていました。

gadget 5

TrakDot

http://www.trakdot.com

荷物トラッキングデバイス

TrakDotは飛行機旅行の際のスーツケースや、大切な荷物などに取り付けて、荷物のある場所を把握できるデバイスです。GSMが搭載され、単三乾電池2本で動作します。貨物がある程度の高度になるとスイッチが切れ、目的地の空港に到着したら携帯電話にSMSを送信してくれます。また、TrakDot専用サイトやiPhone/Androidアプリでも荷物がいつどこに届いたのかを知ることができます。デバイスそのものは50ドルですが、年間利用料が必要です。海外渡航時の飛行機で、積み込んだ荷物が行方不明になったときなどに有効でしょう。2013年3月末発売予定です。

gadget 6

Urban Weather Station

http://www.netatmo.com/en-US/product

気象ステーション

NetatmoのUrban Weather Stationは屋内外の温度や湿度、気圧、二酸化炭素濃度、騒音といった環境情報をiPhoneやiPadを使ってネットワーク経由で確認できるデバイスです。それぞれの値にはしきい値を設定でき、ある数値を超えたら通知させられます。屋内用センサーと屋外用センサーを合わせて180ドル。日本での販売は未定です。本体に表示装置はなく、状態はiPhoneアプリ「Netatmo」でチェックします。

gadget 7

Party in a Box

http://store.soulelectronics.com/products/speakers.html

iPhone/iPad対応 重低音スピーカー

SOUL ElectronicsのParty in a Boxはヒップホップ好きに喜ばれそうな、サブウーハ搭載の重低音大音量、機能満載のサウンドシステムです。2個の6.5インチサブウーハを含む、8個のスピーカーを搭載。FMラジオや車のシガーアダプタにも対応しています。オプションでバッテリーも搭載でき、外に持ち出して使うことも想定されています。バック形状で肩紐もつけられますが、約15kgと一人では持ち歩くのが辛いくらいの重さと大きさです。価格は999.99ドル。ラジカセを見なくなった今、こういうデバイスのニッチなニーズもあるようです。

gadget 8

T9000 LCD

http://www.samsung.com/uk/#latest-home

Evernote搭載冷蔵庫

SamsungのT9000 LCDは10インチサイズのタッチパネル液晶画面を搭載し、Evernoteなどの各種専用アプリが動作する冷蔵庫です。Evernoteに保存されたレシピを呼び出したり、買い物メモをクラウド共有することができます。Evernote以外にも、カレンダーアプリ、冷蔵庫の中身や賞味期限を記録しておくアプリなども動作していました。残念ながらデモ用の機器で、今のところ発売の予定はないそうです。

gadget 9

Smart Glasses M100

http://www.vuzix.com/consumer/products_m100.html

Google Glass風? ヘッドマウントディスプレイ

VuzixのSmart Glasses M100はAndroidを搭載した片目用ヘッドマウントディスプレイです。Android 4.0 ICS搭載で、視野の16度範囲くらいに16:9サイズ、400×240ピクセルの画面が投影されます。720p HD記録が可能なカメラも搭載。ハンズフリーヘッドセットとしても使え、カメラとディスプレイ併用時には1時間くらいのバッテリーの持ちとのこと。左右の目、どちらにも装着可能でボディカラーは白と黒があります。先行して開発者限定向けのSDK付きバージョンが999ドルで販売の予定。一般販売時は499ドルになる予定です。

gadget 10

HAPIfork

http://www.hapilabs.com

減量フォーク

HAPIforkは減量を手助けしてくれるデジタルフォークです。食事のペースを記録し、食べ方をモニタリングすることによって、食べ過ぎを防ぐのが目的です。たとえばフォークを口に運ぶペースが早すぎると、ブルブル震えて警告してくれます。食べ方についてアドバイスしてくれるスマートフォンアプリともBluetoothで連携するそう。重さは65グラム。4月発売予定で99ドル。CES会場でとても話題になっていたデバイスです。

勢いを増す デジタルガジェットたち

先に紹介したもののほかに、今年人気が高かった展示は、三次元プリンタでした。最も安価なものでも1,000ドル台からとまだまだ高価ですが、白黒のレーザープリンタが登場当時数百万、数十万円していたことを考えると、手頃な価格ではないでしょうか?

また、最近話題になることが多いクラウドファンディング、Kickstarterによるハードウェア開発の資金調達の成功例も数多くCESに展示されていました。小さな会社や個人でも、ソフトウェアのみならず、ハードウェアを含めたいろいろなモノ作りに参入してこられるようになった潮流を感じるとともに、大手企業のハイエンド指向なモノ作り、安価に大量生産する領域の難しさがますます実感されます。

CES会期中には米国WIRED誌の元編集長であり「MAKERS—21世紀の産業革命が始まる」(NHK出版)の著者でもあるクリス・アンダーソンのセッションもありました。とくに小規模な個人レベルでの「モノ作り」とプロトタイピング(試作)のプロセスが大きく変化してきたことが話題になっていました。CES全体としても、今までになかったような新規のハードウェア製品を発表するベンチャー企業も多く見られるようになりました。

時代を担う数々の新しい家電製品がデビューする場であるCES。来年もまた大きく流行が変化していることでしょう。来年のCESでも、未来的すぎない、すぐにでも使える興味深い新製品群を期待しています。SD

Where has all the foolish gone?

最終回

小飼弾の #当日 コードなエッセイ

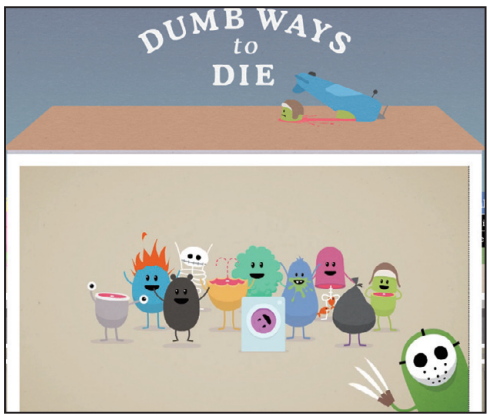
TEXT= 小飼 弾 KOGAI Dan dankogai@dan.co.jp

最初のfool

本連載も、いよいよ最終回。前回はhungryについて語ったので、最後にfoolishについて語ろう。まずは悪いニュースから。

我々は、最初のfoolを持てはやすが、それ以上に2番目以降のfoolに対して冷たい。

Dumb ways to dieという曲がWebでヒットした[注0]。本誌の読者であれば一度は動画をご覧になっているもしれない（私はiPhoneに入れている）。メルボルンの鉄道会社が作った交通安全ソングなのだが、「プラットフォームの縁に立つ」「遮断機が下りた踏切を突っ切る」に加え、「自分勝手に電気工事する」や「飛行機の操縦を独習する」も、「バカな死に方」の1つに入っている。



しかし萌死にキャラたちの死に様を一通り鑑賞した後、ふと気がつくのだ。「では最初に電気工事した奴や、最初に飛行機を飛ばした奴はどうしていたのか」と。もちろん当時は免許なんてないし、誰も知らない以上独習する以外“dumb ways”を進むすべはなかった。最初のfoolが一歩前進するまで、どれほどのfoolsがその一線を超えられず死んでいったか想像を絶する。

しかしその一線を超えた後は、そこが道になっていく。けもの道が舗装されるころには、もう道路交通法が登場して、そこを外れて走るものは文字どおり外道として扱われ、その上を走る際にすら「バカではない」証明、つまり免許が要求されるようになる。

最初のfoolはJobsの言うところの「乾杯すべきcrazy ones」[注1]であるが、それ以降のfoolsはdumb、つまり「調教すべきマヌケ」となるわけだ。どれほど両者の落差が大きいかは、2番目以降を取り締まる法、つまりコードが必ず行き過ぎることからも伺える。学内で無認証のWi-Fiで閲覧可能な論文を集めて学外で公開することのどこが悪いのか、そして誰が傷つくのか法律の素人である私にはわかるすべもないが、それは米国連邦政府の主張によれば、懲役35年、罰金100万ドルに相当する凶悪犯罪なのだそうだ。14歳にしてRSSの仕様を起草したcrazy oneは、26歳にしてdumbとして扱われ、そして自殺

注0） URL http://dumbwaystodie.com/

注1）「Apple Think Different - Steve Jobs Narrated Version」
URL http://www.youtube.com/watch?v=GEPhLqwKo6g

した[注2]。

痛ましい事件ではあるが、しかしAaron Swartzは行き過ぎた法律による最初の犠牲者でもなければ、最後の犠牲者でもあり得ない。本誌の読者とあらばその殿堂にAlan Turingの名が必ずや見出せるはずであるが、codeを自動機械にもたらした彼もまた、(当時の)人倫というcodeを破った咎で罰せられ、自ら命を絶っている。英国政府は自らの「codeのバグ」を認めたものの、そのバギーなcodeを彼に適用したことそのものに対しては、本稿執筆現在いまだ過ちを認めていない。

「智に働けば……」

それでは、the dumbのみを排除し、the foolishを賞揚するようなcodeは書けないのだろうか?

皮肉なことに、Turing自身がそれを否定的に証明している。どんな設問も解こうとする機械は作れても——そして実際こうして作られていても、その機械が解けない問題が必ずある。あるプログラムがfoolishなのかdumbなのかを、有限時間内に判定するプログラムは書けないのだ。

これは「矛盾がない公理システムは、自らが無矛盾であることを証明できない」というゲーデルの不完全性定理のよりエレガントな別解でもあるのだが、この不完全性定理を理解した後、私は1週間飯がのどを通らなかった。仮に天国があり得るとしても、そこが天国であることは証明できないのだ。

しかし、私はある日ふと気がついた。

これは「やってみなくてはわからないことが必ずあり、そしてそれが尽きることはない」という意味でもあるのではないか、と。つまり、"Can you stay foolish forever?"に対する解答は"Yes, we can"なのだ。プログラマが真の意味で失業することは、あり得ないのだ。

ただし、その過程で失望することはいくらでもあり得るし、失望が絶望へと変わり、絶望で絶命してしまうこともいくらでもあることも、Turingは身をもって示してくれたとも言える。

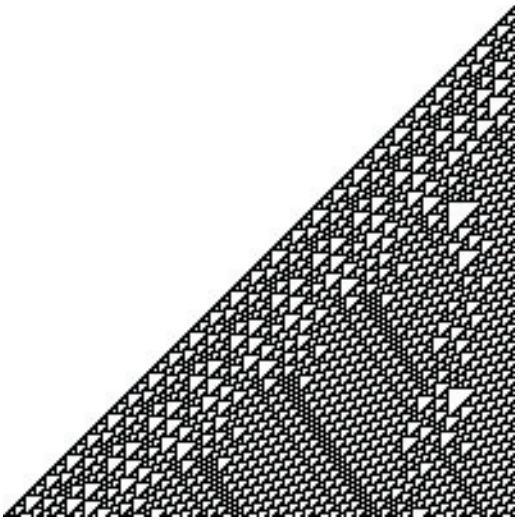
どうすれば、失望を避けられるだろう。dumbであることをひたすら避ければ、今度はfoolishnessが足りないと罵られる。過去最高の決算を出した直後のAppleが時価総額世界一の座をExxon Mobilに明け渡したことを見ると、我々がいかに度し難い生き物であるかを改めて思い知る。全知も全能もあり得ないことをすでに知っているのに、我々自身のベストにさえ満足できないとは。

しかし、これは希望と表裏一体なのだ。ベストよりもよいベターを探す余地が、まだ残っているという。「智に働けば角が立つ、情に棹させば流される、意地を通せば窮屈だ。兎角この世は住みにくい」と漱石は嘆いたけれど、しかしこれは1つ位相をずらすだけで、智に働けば突破口が開け、情に棹させば角は立たず、意地を通せば流されないということになるではないか。

すべての努力が無駄になる世は辛すぎるが、あらゆる努力が報われる世はつまらなすぎる。「カオスの縁」という言葉があるが、静的過ぎず動的過ぎないこの世界が位置するのは、まさにそこではないか。

希望は、絶望の縁にこそある。

あなたがそこにあらんことを。SD



注2) 「Aaron Swartz」(Wikipedia) URL http://en.wikipedia.org/wiki/Aaron_Swartz

Google、Apple、Twitter…

深掘り 裏読み 最新Webトレンド

最終回

協力：TechCrunch Japan
http://jp.techcrunch.com/

▶変わることより変わらないことを見よ
▶TechCrunchの歴史と舞台裏

滑川 海彦 NAMEKAWA Umihiko◉techtrans1999@gmail.com（TechCrunch Japan 翻訳者）
高橋 信夫 TAKAHASHI Nobuo◉nobuo.takahashi@nifty.com（TechCrunch Japan 翻訳者）

本連載では、Webメディア「TechCrunch Japan」（http://jp.techcrunch.com/）の記事を翻訳している2人が毎回、同サイトで取り上げている最新Webサービスや企業・ビジネスに関する膨大なエントリを訳出する中で集積している、米国を中心とした動向、さらにその背景を解き明かしていきます。

変わることより変わらないことを見よ

滑川 海彦

Amazonの創立者、CEOのジェフ・ベゾスは2012年末のre:Inventカンファレンスでこう語りました。

「10年後には世の中はどう変わっていると思うか」とよく質問される。しかし本当は「10年後でも変わらないものは何か」という質問のほうが重要だ。10年後にユーザが「ジェフ、Amazonは安すぎるな。もう少し値段を上げてくれないか」と言うことは絶対にない。ユーザは安いものが好きだ。速く届くのが好きだ。品揃えが豊富なのが好きだ。これは絶対に変わらないんだ。

なるほど名言です。ベゾスには及ばずながら「この先数年後までテクノロジ界で変わらなそうな顔ぶれ」の動向を占ってみましょう。

テクノロジ業界の今後は?

Amazonのベゾス、Googleのラリー・ペイジ（**写真1**）とサーゲイ・ブリン、Facebookのマーク・ザッカーバーグ。不慮の事故や健康問題などがない限りこの4人と3社が5年後も現在同様、あるいは現在以上にテクノロジビジネスの動向に影響を与えていることは間違いありません。ベゾスは物品、書籍の通販からKindle事業、クラウドコンピューティング事業のすべてにわたって万全の態勢を取っており、地位の揺るぎなさでは筆頭かもしれません。

一方新分野へのチャレンジ精神が最も強いのはGoogleです。共同創立者、ペイジはワイアード誌のインタビューで「10倍に成長できる分野にしか興味がない。変化の激しいテクノロジの世界では10%の改良をいくら積み重ねても結局は時代遅れになる。クレージーと言われなければ、それは間違ったことをやっているのだ」と過激なコメントをしています。

なるほど「世界中の道路をすべて写真に撮る」というストリートビューという構想も当初はクレージーと言われました。しかしAppleが独自地図で惨めに失敗したとき、対照的にGoogle Mapsの高精度が改めて絶賛を浴びました。この高精度を支えた大きな要素は、自前の撮影車で撮影した膨大な写真データから自動的に情報を抽出して地図をアップデートしてきた蓄積とそのノウハウでした。自動走行車やGoogle Glassも当初はクレージーと言われましたが、今やトヨタをはじめとする主要自動車メーカが自動走行車をデモするまでになっています。

一方、ザッカーバーグは1月16日にMicrosoftと提携してFacebookの中からBing検索エンジンを利用できるようにしたこと、指定した関係性を持つ相手を自在に検索するグラフ検索という機能の実験も始めたことを発表しました。グラフ検索は簡単に言えば人物検索です。検索結果として所定の条件に合致す

るユーザの友だちが返されます。「インド出身、サンフランシスコ居住、独身の男性は？」といった検索ができるということです。Microsoftは、提携によりFacebookは「従来の5倍もの情報をBingに提供するようになった」としています。しかし検索機能の強化は、Facebookの最大の弱点を増幅する可能性のある諸刃の剣でもあると思います。

日本のFacebookでは、「いいね！」を利用した「格言スパム」が話題になっています[注1]。これは「格言」やあたりさわりのない写真などを投稿し、「いいね！」ボタンの押下により多くのユーザに企業名（と企業へのリンク）が表示されることを狙うものです。きれいな写真に「いいね！」するとそれが興信所の広告で恥をかいた例などもあります。

Facebookのマネタイズ上の問題点というと決まってプライバシー問題が持ち出されますが、プライバシーの概念、常識は時と共に変わります。しかし見たくない広告を見せられて喜ぶユーザはいるはずがありません。その広告に自分や友だちの名前が勝手に（少なくともそのつもりがなくて）使われていた場合、不快さは10倍になるでしょう。

Googleが検索エンジンとして成功した理由は、当時のライバルに比べてスパム耐性が圧倒的に高いことでした。これによってAltaVistaなどの先行検索エンジンを軒並みに打倒し、爆発的成長が始まったのです。そこでスパムの抑圧がGoogleという企業の本質に刷り込まれたのです。Googleではマネタイズを担当したシェリル・サンドバーグの提案に対し、ラリー・ペイジ、サーゲイ・ブリン、エリック・シュミットのトロイカとWebスパム対策責任者のマット・カッツなどが拒否権を持っていました。スパム撲滅は一時の仕事ではなく継続的な努力です。カッツ率いるアンチスパムチームはこれまで定期的に（ときには過激なまでに）アルゴリズムをアップデートしてスパムを抑圧してきました。

ところがFacebookに招聘されたサンドバーグは、ザッカーバーグの分身で絶対的ナンバー2となり、彼女にノーと言えるのはザッカーバーグだけです。サンドバーグより下のレベルの幹部が過激なマネタイズ手法を実験しても、サンドバーグ以外にはそれにストップがかけられない。そして組織のレベルを下るにつれて競争が手段を選ばなくなるのは人性の本然のしからしむるところ……というような事態がFacebookで起きているのではないでしょうか。

ソーシャルネットワークは予想されるよりはるかに囲い込み効果が薄いことは、MySpaceのあっという間の没落でよく証明されています。もちろんザッカーバーグのこと、近い将来、スパムの蔓延こそがFacebookにとって最大の危険だということを認識して対策に本腰を入れると思います。しかしスパム対策もストリートビューと同様、一日で完成するものではない。Facebookはこの面でかなり出遅れていますから、大きなハンディキャップです。「検索でもGoogleを圧倒する」という外野のファンの称賛に喜んでいられる場合ではありません。

さてここまでAppleの名前を挙げませんでした。その理由はペイジの言う「クレージーなリーダー」の不在です。なるほどAppleは数年後も世界屈指の優良大企業であり続けるでしょう。しかし画期的イノベーションは改良からは生まれません。ティム・クック以下の経営陣はそれぞれが極めて有能なマネージャですが、ラリー・ペイジが言うような意味での「クレージー」な人物ではないでしょう。また創立者や大株主ではない雇われCEOの場合、クレージーになりたくても取締役や機関投資家など周囲がなかなかそれを許しません。Appleは次第にMicrosoftのような巨大優良企業になっていくだろうというのが筆者の推測ですが、どうなるでしょう？ 数年後にまた振り返ってみる機会でもあれば幸いです。

Creative Commons by Marcin Mycielski

写真1
GoogleのCEO、ラリー・ペイジ

注1）「気軽な「いいね」で恥をかく。Facebookに張り巡らされ始めた罠について」
URL http://www.landerblue.co.jp/blog/?p=5141

TechCrunchの歴史と舞台裏

高橋 信夫

この連載は今月号で最後です。3年に1ヵ月だけ足りない35回も続けることができて感謝しています。

最終回は、滑川海彦さんと私が毎日翻訳している情報ブログでこのコラムのネタ元でもあるTechCrunchについて、歴史と舞台裏などをご紹介したいと思います。

本家米国のTechCrunchは、2005年6月、マイケル・アリントン氏（**写真2**）が創立しました。同氏は弁護士資格を持ち、いくつかのベンチャー企業の設立に関わった後、シリコンバレーの情報を発信するブログとしてTechCrunchを立ち上げました。最初の記事はこれです。

Technorati Beta Profile

（2005/6/11, Michael Arrington）[注2]

当初は企業を1社ずつ紹介する記事が多く、この記事ではTechnoratiを「Web 2.0の古株で、最初（かつ最高）のリアルタイム検索エンジン」と紹介しています。他のライターが加わったのは2006年2月で、それまでの8ヵ月間は1人で記事を書いていたことになります。なおこの記事は「TechCrunch初投稿」ということで、その後もときどき「Happy Birthday」などのコメントが付いています。最新のものは2011年6月11日付です。中には「本当にこれが最初の記事？ もっと古いのがなかったっけ」などというコメントもあります。

日本版の幕開け

本家から遅れること約1年、2006年5月5日に日本版TechCrunchで初の記事がアップされました。

Microsoft、Froogleの競合をひっそりとローンチ

（2006/5/5, Michael Arrington）[注3]

MicrosoftがGoogleのショッピング検索サイトに対抗するサービスを開始した、という記事です。Froogleという名前は何だかGoogleのまがい物みたいですが、Google純製のサイト名です。現在は単にショッピングというカテゴリになっています。

日本語版TechCrunchを立ち上げたのは、トランスコスモス・アメリカでベンチャー投資などを行っていた西田靖さんです。当初は西田さんが1人で全記事を翻訳・編集するという超人的な運用でした（本家も1人でしたが、ある意味で翻訳はもっとたいへん）。やがてライターが増え、翻訳者を募ることになりました。まず米国在住の方々が加わり、2006年6月には滑川さんが日本側で最初の翻訳者になりました。筆者は同年8月末にスタートし、その後岩谷宏さん、前田博明さんが加わり、2008年以降ほぼこの4名体制で翻訳しています。

日本版が始まった年、Facebookはまだ学生と一部の大企業専用で、翻訳の参考にと入会しようにもできないという有様。Twitterも無名で、iPhoneもなく、スマートフォンとはBlackBerryを指していた時代でした。

速報！GoogleがYouTubeを買収

（2006/10/10, Michael Arrington）

TechCrunch史上最大のスクープは何と言っても2006年10月のGoogleによるYouTubeの買収です。いち早く噂をキャッチしたアリントン氏が「Google/YouTubeの根も葉もない噂」（2006/10/6）という大胆なタイトルで初報したところ、ブログ界は大騒ぎになりました。しかし、その後両社から何の発表もなく読者や同業者からの批判が高まったころ、ウォール・ストリート・ジャーナルやニューヨーク・タイムズもこの件を取り上げ、結局噂は本当になり、10月10日、16.5億ドルの大買収劇となりました。記者会見が日本では深夜だったため、翻訳チームにとっても忘れられないビッグイベントでした。

TechCrunch 40/50

TechCrunchにとって、ある意味で情報ブログ以上に重要なのがスタートアップを世に出すためのイベ

注2）URL http://techcrunch.com/2005/06/11/technorati-new-improved/

注3）URL http://jp.techcrunch.com/archives/microsoft-quietly-launches-froogle-competitor/

ントです。2007年9月、連続起業家で業界の暴れん坊として知られるジェイソン・カラカニス氏と共同でTechCrunch 40を開催しました。厳しい選考を経た有望スタートアップ40社がベンチャーキャピタリストや業界関係者の前でデモとプレゼンを行い、勝者が選ばれます。ブログには逐一状況が報告され、翻訳チームも特別体制で対応しました。2008年には50社へと規模を拡大してTechCrunch 50になり、2009年まで続きましたが、カラカニス氏とアリントン氏の確執から2010年は中止となり、それ以降はTechCrunch Disruptと名前を変え、TechCrunch単独のイベントとして現在に至っています。2008年のTechCrunch 50には日本から頓智ドットの井口尊仁氏が「世界カメラ」を引っ提げて参加し、大きな話題を呼びました。

コメント欄

本家米国版のコメント欄には大量の書き込みがあります。スパムまがいのものもありますが、業界の有名人がコメントを付けたり、ライター本人が返答するなど、活発なやりとりが行われています。ときには本文よりもコメント欄のほうがおもしろいこともありますが、コメントを全部訳すこともできないので日本にお届けできず残念に思っています。日本語版でもコメントが増えてきました。間違いの指摘も多く恐縮しておりますが、建設的な議論が交わされることを期待しています。

マイケル・アリントンという人物

アリントン編集長は広い情報網を活かしてニュースをキャッチするだけでなく、業界へも強い影響力を持っていました。中でもソーシャルゲームで詐欺まがいの方法で荒稼ぎしていた会社にぶつけた怒りは凄まじいものでした。2009年、Zyngaの人気ゲーム、ファームビルなどのバーチャル通貨を獲得できる、という詐欺同然の手法で紹介手数料を稼いでいたOfferpal社の女性CEOを、アリントン氏が公開の場で激しく非難しました。彼女はその場で強く反論しましたが、結局1週間後にCEOの座を追われました。一方同じく非難を受けていたZyngaのCEO、マーク・ピンカス氏はいち早く過ちを認めて詐欺行為を排除しました。

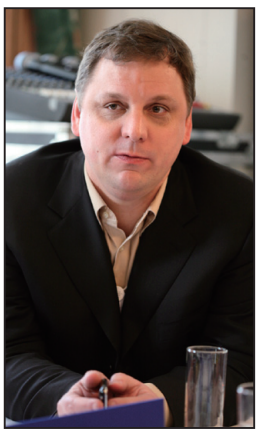
AOL傘下に、そしてアリントン氏の離脱

2010年9月、TechCrunchにとって最大の出来事がありました。会社をAOLに身売りしたのです。これには業界もあっと驚きました。何しろ自由奔放な記事が売りのTechCrunchが、業界大手で、批判の鉾先の1つでもあるAOLの傘下に入るのですから。アリントン氏は、「われわれがTechCrunchをAOLに売った理由。そしてこれから」(2010/9/29)という長文の記事を書いて、「編集権の独立」を守ること、あと3年はここに留まるつもりであることを読者に伝えました。

ところが、そのわずか1年後、アリントン氏は突然TechCrunchを去ることになります。彼は「編集権の独立」(2011/9/6)と題した記事に思いを綴りました。AOLとのいざこざはいくつも見られましたが、辞任の最大の理由は「TechCrunchの編集権の独立が失われた」からだと本人は書いていました。新編集長には元フォーチュン誌の記者で、2007年以来アリントン氏を助けてきたエリック・ションフェルド氏が任命されました。現在はエリック・エルドン氏とアレクシア・ツォッツィス氏が共同編集長を務めています。TechCrunchはマイケル・アリントンという看板を失い、人の出入りも多く心配された時期もありましたが、現在はFacebookを得意とするジョシュ・コンスティン記者などライター陣も再び落ち着いてきました。

これからもTechCrunch Japanをよろしく

2011年4月1日以降、TechCrunch JapanはAOLオンライン・ジャパンの運営となり、西田隆一編集長のもと、翻訳記事だけでなく西田編集長や外部執筆者による日本版オリジナル記事も掲載して好評をいただいています。本連載終了後も、引き続きWebでお会いできれば幸いです。SD



Creative Commons by Robert Scoble

写真2
TechCrunchの創立者、
マイケル・アリントン

はんだづけカフェなう

Raspberry PiでI/Oしてみよう(中編)

text:坪井 義浩　TSUBOI Yoshihiro　ytsuboi@gmail.com　@ytsuboi
協力:㈱スイッチサイエンス　http://www.switch-science.com/

Raspberry Piのいいところ

前回はLED点滅というArduinoでも可能なことを紹介しましたが、今回はRaspberry Piだからできるということをしてみようと思います。ずばり、USBホストとTCP/IPです。

ArduinoでもUSBホストシールドを使えばUSBホスト機能を持つことはできますが、デバイスごとに制御コードを書くのはけっこう骨が折れる作業です。Raspberry PiではLinuxが動いていますから、Linuxでサポートされているデバイスなら割にスンナリ動くことになります。ちなみにRaspberry PiのEthernetは、LAN9512というUSB 2.0 HUBと10/100 Ethernetコントローラを兼ねたチップによってUSB接続で搭載されています。

Ethernetについても、ArduinoではEthernetシールドを使えばTCP/IPを喋らせることができます。しかし、TCP/IPをシールドの上に載っているチップにオフロードしているため、快適とは言えません。また、たとえばTweetさせようとすると、OAuthのような複雑な処理は8bitマイコンには苦しく、本連載の第8回で扱ったように、ゲートウェイを併用して呟くといった仕掛けが必要になってしまいます。こういった複雑なプロトコル処理も、Linuxが動くようなCPUであればボード上で実行させることができます。

ということで、今回はRaspberry Piで「USB接続のWebカメラを使い、撮った写真をTweetする」ということを実現したいと思います。"Raspberry PiでI/Oしてみよう"という主旨から外れていると思う方もいらっしゃるかもしれませんが、「USBカメラでの撮影」がI(入力)で、「Tweetする」というのがO(出力)ということでよろしくお願いします。

また、前回書いたように筆者はPythonが好きなので、開発はPythonを使います。

USBカメラを接続してみる

USBカメラですが、Mac miniに接続して使っていたLogicoolのC615と、UVC(USB Video Class)対応なのでスンナリ動くかなと思って購入したELECOMのUCAM-DLW500TA(500万画素)とUCAM-DLK130T(130万画素)で試してみました。実はLogicoolのWebカメラもUVCのドライバで認識できたので、追加で買う必要がなかったことにあとで気づきました。

カメラを接続してストリーミングするだけであれば、MJPG-streamerというアプリケーションで簡単に実現できます。必要なライブラリをapt-getでインストールし、MJPG-streamerをコンパイルして、カメラを接続するだけでWebブラウザからカメラで撮影した画像を確認できます。

しかし、そんな動いて当たり前のことをしてもおもしろくないので、今回はOpenCVとPythonを使ってスクリプトから写真を撮影してみることにしましょう。

まず、標準ではインストールされていないパッケージをインストールします(図1)。

Raspberry Piはやはり遅いので時間がかかりますが、ビルドするよりはマシと自分に言い

聞かせながら待ちます。次に、SimpleCVというPythonのライブラリをインストールします（図2）。これで準備は完了です。次にちょっとしたスクリプトを書いて試してみましょう（リスト1）。

実行すると、やはりRaspberry Piは遅いので少し待たされますが、カメラで撮影した640×480ピクセルの画像が保存されます。“VIDIOC_QUERYMENU: Invalid argument”と表示される場合もありますが、v4l（Video For Linux）のドライバが出力しているメッセージで、メッセージが表示されてもキャプチャした画像は保存されていました。

なお、Raspberry PiのUSBポートは供給可能な電流がポート当たり140mA程度に制限されています。リセッタブルヒューズという、何度も使えるヒューズのような部品がUSBポートの電源に接続されており、これによってパソコンのUSBポートでは正常に利用できる機器も使えないケースがあります。USB機器がうまく動かない場合はセルフパワードのUSB HUB（ACアダプタで電源供給可能なタイプのUSB HUB）をRaspberry PiのUSBポートに接続し、カメラをHUBに接続することでこの制限を回避できます。

先ほどの筆者が用意した3機種では、LogicoolのC615が筆者が最も満足のいく結果が得られました。

Tweetしてみる

次に、撮影した画像をTwitterのメディアつきTweetで呟かせることに挑戦してみましょう。

PythonでTweetするライブラリは数多くあるのですが、メディア（写真）つきでTweetできるものとなると少なくなってしまいます。今回はtwythonというライブラリを使いました。

```
$ sudo pip install twython
```

これでtwythonのインストールは完了です。あとは、リスト2のようなスクリプトを書いてやります。

▼図1　インストールされていないパッケージをインストールする

```
$ sudo apt-get install python-opencv python-scipy pythonnumpy python-pip
```

▼図2　SimpleCVをインストール

```
$ sudo pip install https://github.com/ingenuitas/SimpleCV/zipball/master
```

▼リスト1　画像を取り込む

```
from SimpleCV import Camera
myCamera = Camera(prop_set={'width':640, 'height': 480})
frame = myCamera.getImage()
frame.save("captureimage.jpg")
```

▼リスト2　つぶやくスクリプト

```
#!/usr/bin/env python
# coding: utf-8

from SimpleCV import Camera
from time import sleep
from twython import Twython

myCamera = Camera(prop_set={'width':640, 'height': 480})
frame = myCamera.getImage()
frame.save("/home/pi/live.jpg")

twitter = Twython(
    twitter_token = 'Twitter  CONSUMER_KEY',
    twitter_secret = 'Twitter  CONSUMER_SECRET',
    oauth_token = 'Twitter  ACCESS_TOKEN',
    oauth_token_secret = 'Twitter  ACCESS_TOKEN_SECRET'
)

twitter.updateStatusWithMedia('/home/pi/live.jpg', status=unicode('てすとてすと','utf-8'))
```

トークンの類については、Twitterの Developersサイト[注1]でアプリケーションの登録をして取得をします。

Twitterのアカウントでログインをしたら、まずアカウント名のところにあるドロップダウンの“My applications”を選択します(図3)。次に、“Create a new application”ボタンをクリックし(図4)、表示された“Create an application”の画面で必要な情報を入力していきます(図5)。

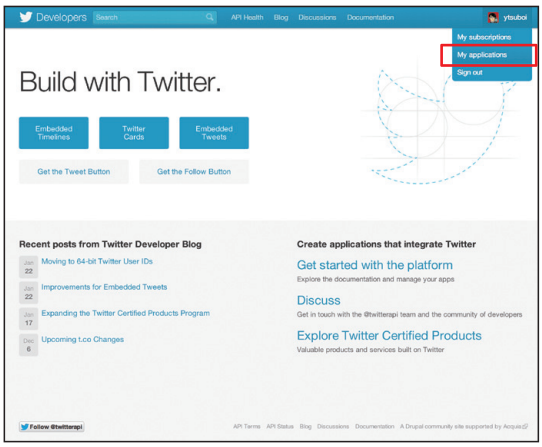
“Application Details”の項目にあるNameやDescriptionといった項目とは別に、Websiteにはアプリケーションの説明ページを入力します。アプリケーションの説明ページは入力必須項目ですので、何かプレースホルダーを入力しておきましょう。あとは“Developer Rules Of The Road”を確認し、CAPTCHAを入力して登録します。

これでアプリケーションの登録が完了します(図6)が、Tweetするためには“Access Level”が“Read only”では困ります。“Settings”タブをクリックし、“Read and Write”に変更しておきましょう。変更をしたら、“Details”タブに戻り、“Your access token”の項にある“Create my access token”ボタンをクリックします。するとアクセストークンが発行されます(図7)。

こうして得られたキーやシークレットをスクリプトに埋め込み、実行するとWebカメラで撮影した画像を呟くことができます(図8)。

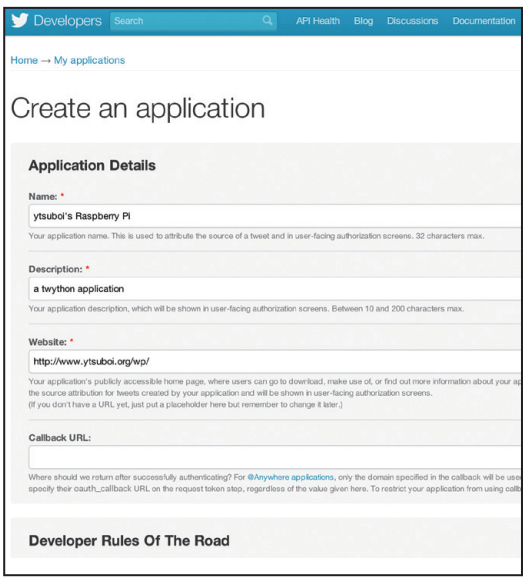
注1) https://dev.twitter.com/apps

▼図3 My applicationsを選択する

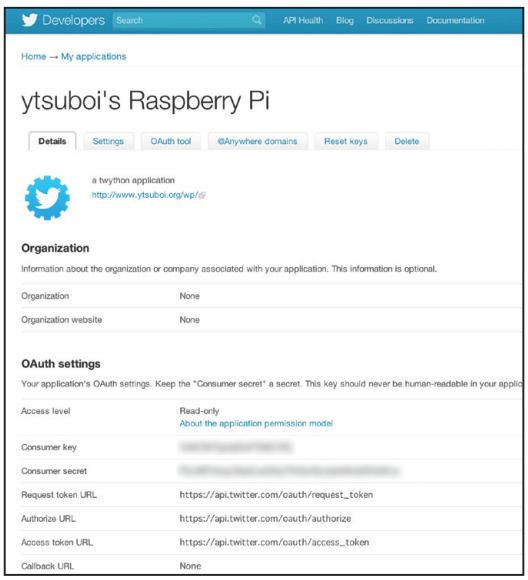


▼図4 Create a new applicationをクリックする

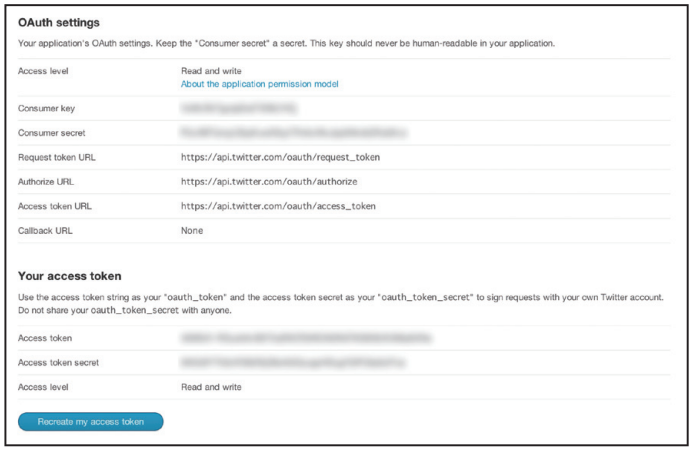
▼図5 Create an application画面に必要事項を入力



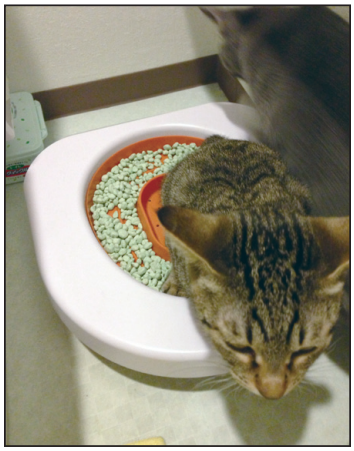
▼図6 アプリケーションの登録が終了



▼図7　アクセストークンが発行される



▼写真1　決定的瞬間を撮りたい



最終的な目的とは……

「Raspberry Piで写真を撮ってTweetをする」ということをここまで進めてきましたが、このようなことをするには目的があります。筆者は昨年の8月から猫を飼い始めたのですが、自動で猫の様子の写真を撮りたいと思ったからです。ただ撮影するだけであれば、ネットワークカメラで簡単に実現ができるのですが、もうちょっと手の込んだことをしたいと思いました（**写真1**）。たとえば、センサを接続して、「猫がトイレに30秒以上いたら撮影し、通知を送る」といったことをしたいのです。

こういった通知と画像アップロードを行うにはTwitterのメディアつきツイートで自分宛にmentionを送れば簡単に実現できます。Twitterであればサーバを用意する必要もありませんし、何か有償のクラウドを契約する必要もありません。

また、Raspberry PiのGPIOにはさまざまなセンサが接続できますから、猫の検出方法も、赤外線センサ、明るさセンサ、重量センサ、NFC（Near Field Communication；近距離無線通信）やRFID（Radio Frequency IDentification；電波による個体識別）による検出など、さまざまな工夫ができます。

今回は写真を撮影してツイートするところまで進めましたが、次回はRaspberry Piに猫検出センサを接続して、Twitterによる猫監視システムの完成を目指したいと思います（**写真2**）。SD

▼図8　自動的画像付きでつぶやく

▼写真2　3匹もいるんだニャオ

ニートなphaのぶらぶら日記

ギークハウスなう

TEXT＝pha　pha.japan@gmail.com

最終回

3年間の連載を振り返って

（ギークハウスでインターネットをするギークたち）

最終回のお知らせ

3年間続いたこの連載も今回が最終回となった。「phaがネットで気になったことについて書く」というゆるいコンセプトで毎回好きなことを書かせてもらっていたんだけど、3年間も毎月書いているとだんだん書くこともなくなってきたし、ちょうどいい区切りという感じだろうか。

紙とWebの違い

雑誌で連載をするのは初めての経験だった。それまではずっとブログを書いていて、ブログがある程度有名になったのをきっかけに連載の話が来たんだけど、実際に書き始めてみると紙とWebでは同じ文章を書くのでもまったく違うなということを感じた。その中でも一番戸惑ったのは「雑誌だと読者の反応が全然わからない！」ということだった。

Webだと、読者の反応は記事を公開して数十秒後くらいからすぐ現れ始める。どのリンクを経由してどれくらいの人が見に来てくれるかはアクセス解析を見れば一発でわかるし、内容についてもTwitterやはてなブックマークなどでいろんな人が感想やツッコミを入れてくれる。そうした反応をニヤニヤしながら見るのはWebで物を書くことの楽しみの1つで、とくに誰に頼まれているわけでもないのにずっとブログを更新し続けるのはそうした反応を見ることが大きなモチベーションになっている。

それに比べて雑誌だと、反応が全然見えて来ない。このSofware Designは全国でそれなりの部数が発行されていてそれなりの数の人が読んでいるはずだけど、ときどきTwitterやブログ検索などでエゴサーチをしてみても雑誌の記事の感想をネットで見かけることはほとんどなかった。2ヵ月に1件、あるかないかくらいだ。それはブログを書くことに比べると、まったく反応のない深い穴の中に文章をひたすら放り込み続けているような気持ちだった。実際には初めて会った人に「Software Designの連載読んでますよ」とか言われることもときどきあるので、可視化されているかいないかの違いだけで、それなりの人には届いているんだろうけれど。

また、紙よりWebのほうが読者数の変動が大きい。Webだと、普段はアクセスがそれほどないサイトでも何かのきっかけで話題になると、リンクが拡散して何万人、何十万人もの人に読まれることがある。それには良い面も悪い面もあって、Webはうまくいけば爆発力はあるけど、あまりそれを意識しすぎると地味な記事を書きにくくなって、つい扇情的なタイトルを付けたり大げさなことを書いたりする誘惑にかられてしまう。だけど地味で着実な記事というものももちろん世界では必要とされているわけだし、そうした地味な仕事を着実に書き続けられるというのではブログより雑誌という形態のほうが良い面もある。

僕自身のやり方としては、雑誌発売からしばらく経ってから自分のWebサイトでも原稿を公開する、ということを始めた。これだとWebの人から感想などを聞くこともできるし、雑誌だけ読んでいる人にも意見を届けることができるし、両方の良いところを取れたんじゃないかと思う。これだと雑誌から原稿料ももらえるし。なんだかんだ言って今の時点では、ブログに記事を書いてそこにアフィリエイトを貼ると

やる気のないニートに
メシをおごる権利が
今ならなんと3000円！

phaがOREPONというサービスでお金を集めたときの画像

いうようなやり方よりも紙媒体に書いて原稿料をもらったほうが収入としては良い。

マイナーなものは2年で化ける

3年、というのはネットの世界ではかなり長い期間だ。この連載をしていた3年の間にもさまざまなネットサービスが生まれては消えていった。その中でも面白いと思ったのは、一部の変わり者の間での流行だと思って取り上げたものが、2年くらいあとになってより一般性を増した形で再度流行る、という現象だ。

たとえば2010年10月号掲載の「新しいカンパを考える」では、「金くれ」というサイトやTwitterで銀行口座をネットに晒してお金を募集するというカルチャーについて紹介した。この動き自体は、一部の変な若者たちがやっているだけの、かなりアングラ性が高くて胡散臭いものだった。そしてその後2012年5月号掲載の「人はネットカンパだけで生きていけるか」ではGumroadという少額決済サービスやクラウドファンディングについて紹介した。クラウドファンディングというのは「インターネットを使って不特定多数の人から少しずつお金を集める」というもので、内容や思想としては「金くれ」カルチャーと通じる。ただ、クラウドファンディングには「金くれ」のようなアングラ感はなく、きちんとした会社がきちんとした目的のためにお金を集めている。つまり綺麗な「金くれ」と呼んでもよいだろう。

また、2010年9月号掲載の「ストリートコンピューティングの発展と衰退」で紹介した「ストリートコンピューティング」は、2009年から2010年ごろに流行した、秋葉原の路上で座り込んで（ときには立ったままで）ノートパソコンを使う若者たちのことで、「小池スタイル」と呼ばれる独特のパソコンを使う姿勢とともに話題になった。「発展と衰退」と書いているのは、スマートフォンが発達することでパソコンでなければできない作業が減り、わざわざ無理して外でパソコンを使うことが減るだろうという内容で、実際2013年現在ではほとんど見かけなくなった。

このストリートコンピューティングはのちに流行ったノマドワーキングにある面で通じているのではないかと思う。ノマドという言葉が指すものには生き方から働き方から仕事をする場所を指すものからいろいろなレイヤーがあるので一概にはくくりづらいが、単にカフェなどでノートパソコンを広げて仕事をするというスタイルを指したノマドワーキングについては、ストリートコンピューティングと同じで、「ブロードバンド環境の整備によってどこでもパソコンを使えるようになったので、うれしがっていろんな場所で使ってみている」というだけの要因も結構大きかったのではないかと思っている。それで最初は面白いんだけどだんだん飽きてきて、やっぱりオフィスや自宅で作業するほうが落ち着く、みたいな感じになったりする。

技術的インフラが整えばそれを利用した新しいカルチャーは発生するし、それはたいていの場合、一部の変人たちの悪ふざけから始まる。そして新しい概念や思想や習慣が世の中に定着するには、2年や3年という期間が必要となる。だから新しく流行するものを知りたければ、現在一部の変な人たちがやっているようなアングラなカルチャーに注目するとよいかもしれない。それが2年くらいすると、その気持ち悪さを解毒して、最初の尖った面白さも削られてしまうけどその代わり一般性を獲得したものが、メジャーな場所で大規模に流行する可能性は結構あるんじゃないだろうか。

それでは、こんなところで。僕は生きている限りインターネット上で何かを書き続けていると思うので、ネットのどこかでまたお会いしましょう。御愛読ありがとうございました。SD

ストリートコンピューティングをするpha

小池スタイルでパソコンを使うpha

PRESENT 読者プレゼントのお知らせ

『Software Design』をご愛読いただきありがとうございます。本誌サイト http://sd.gihyo.jp/ の「読者アンケートと資料請求」からアクセスし、アンケートにご協力ください。ご希望のプレゼント番号をご入力いただいた方には抽選でプレゼントを差し上げます。締め切りは2013年3月17日です。プレゼントの発送まで日数がかかる場合がありますが、ご了承ください。

ご記入いただいた個人情報は、プレゼントの抽選および発送以外の目的で使用することはありません。アンケートのご回答については誌面作りのために集計いたしますが、それ以外の目的ではいっさい使用いたしません。ご入力いただいた個人情報は、作業終了後に当社が責任を持って破棄いたします。なお掲載したプレゼントはモニター製品として提供になる場合があり、当選者には簡単なレポートをお願いしております（詳細は当選者に直接ご連絡いたします）。

01 バックライト付キーボード SKB-WAR2

1名

2色（ブルー／ホワイト）のLED色を選択できるバックライト付きのUSBキーボードです。キーストロークが深く確実なキータッチが得られるメンブレンタイプを採用。Windows 8/7/Vista/XPを搭載したPC、およびPlayStation 3、Wiiで使用できます。

提供元 サンワサプライ
URL http://www.sanwa.co.jp

02 ストレッチングレーザーマウス DN-46382

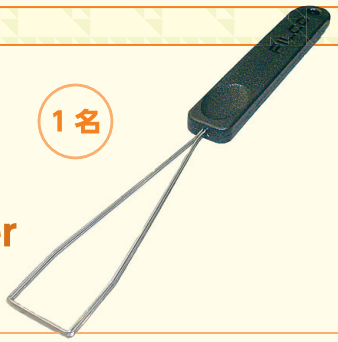
2名

長時間使用しても手首に負担がかからないエルゴノミクスデザインのマウス。2ボタン、2サイドボタン、ホイールボタン、カウント切り替えボタン付き。Windows 8/7/Vista/XP、Mac OS Xで利用可能。

提供元 上海問屋
URL http://www.donya.jp

03 FILCO KeyPuller

1名

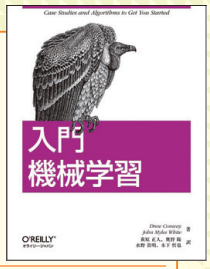


キーボードの製造現場で生まれたプロ仕様のキートップ引抜工具。キーを引き抜く際の傷つきを防止し、キートップの紛失リスクを低減します。メカニカルまたはメンブレンキーボードで使えます。

提供元 ダイヤテック
URL http://www.diatec.co.jp

04 入門 機械学習

2名



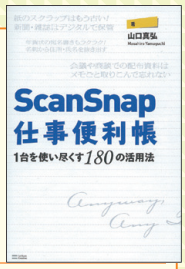
Drew Conway、John Myles White 著、
萩原 正人、奥野 陽、水野 貴明、木下 哲也 訳／
B5変型判、344ページ／
ISBN = 978-4-87311-594-8

プログラマ向けの機械学習の入門書。難しい理論的な解説よりも実際のテクニックを詳述します。大規模データ処理に威力を発揮する機械学習の知識とテクニックを習得したい人に最適の1冊です。

提供元 オライリー・ジャパン URL http://www.oreilly.co.jp

05 ScanSnap 仕事便利帳

2名



山口 真弘 著／
四六判、168ページ／
ISBN = 978-4-7973-7241-0

高い人気を誇るドキュメントスキャナ「ScanSnap」。その活用方法やアイデア180個を掲載。情報整理、会議効率化、人脈作り、スキルアップなどの目的別に紹介。活用シーンがぐっと広がります。

提供元 ソフトバンク クリエイティブ URL http://www.sbcr.jp

06 世界で闘うプログラミング力を鍛える150問

1名

Gayle Laakmann McDowell 著、
Ozy 訳、秋葉 拓哉、岩田 陽一、北川 宜稔 監訳／
B5変型判、456ページ／
ISBN = 978-4-8399-4239-7

米国でベストセラーとなっている書籍の日本語版。トップIT企業のプログラミング面接に合格するための攻略本として、Microsoft、Apple、Google に勤務経験のある著者自らが執筆しました。

提供元 マイナビ URL http://book.mycom.co.jp

07 サウンドプログラミング入門

2名

青木 直史 著／
A5判、288ページ／
ISBN = 978-4-7741-5522-7

C言語を使って、シンセサイザのようにさまざまな音を作り出す方法を紹介します。基本的な音作りのしくみ、音響合成の代表的な方式、具体的な音作りのテクニックなど、プログラムとともに解説します。

提供元 技術評論社 URL http://gihyo.jp

第1特集

OpenStack、CloudStack、Cloud Foundry、Scalr、Eucalyptus

オープン環境でスキルアップ!

もっとクラウドを活用してみませんか?

Amazonが2006年にクラウドサービスを開始して、はやくも7年が経過しました。爆発的に利用者が増えて、2013年は仮想サーバのインスタンス数が物理サーバ数を超えたという発表がされています。現時点で、データセンター企業や通信キャリア各社、SIerまでクラウドサービスを競って展開しています。このようなクラウドサービスはシステムを動かすにあたり、操作性に優れたユーザインターフェースをもち、GUIでどんどんシステムを運用できます。しかし、ITエンジニアは、やっぱりそのクラウドのしくみをちゃんと知って使いこなしたいものです。本特集では、ユーザが増えつつあるオープンソースのクラウド環境(IaaSとPaaS)を紹介し、その使い方を解説します。

Part 1

ユーザ数ナンバー1！

OpenStackでクラウドサービスを始めるには

近年、クラウドコンピューティングは企業や個人、行政、教育、研究の分野など幅広く活用されるようになり、従来のITシステムでは実現できなかったさまざまなサービスが提供されるようになりました。これらを支える重要なクラウド管理基盤技術の1つとして「OpenStack」が注目を集めています。

伊藤忠テクノソリューションズ株式会社ITインフラサービス企画開発部　中島 倫明　NAKAJIMA Tomoaki
株式会社ビットアイル　ビットアイル総合研究所　石川 裕基　ISHIKAWA Hiroki

OpenStackの概要

OpenStackとは

OpenStack はオープンソースで提供されるクラウド基盤ソフトウェアで、Apache License 2.0下で提供されています。IaaS(Infrastructure as a Service)機能に焦点を当てており、ユーザに対してプールされたリソースを仮想サーバ、仮想ストレージ、仮想ネットワークとして提供します。Amazonが提供するEC2/S3/EBSをモデルとしており、現時点でほぼ同等の機能を実現している一方で、最近では仮想リソースだけでなく物理リソースを提供するBareMetalServer as a Serviceや、PaaSを提供するDB as a Serviceの開発も活発に進んでいます。

OpenStackの特徴

OpenStackは大規模なスケールアウトに対応するため、機能ごとにモジュール化され、それぞれをインストールした汎用サーバを並列配置できるようになっています。各コンポーネントはPythonで記述されます。コンポーネントによっては汎用サーバで動作するだけでなく、メーカー固有のハードウェア・ソフトウェアと組み合わせて利用することもできます。

●操作性

ユーザ自身が操作するセフルサービスモデルを採用し、インターフェースにはWebUIとREST APIが利用できます。

●開発スタイル

Ubuntuと同じ方式をとっており、プロジェクト全体はLaunchpadによって管理されます。ソースコードはGitHub上で管理されており、レビューシステムにはGerritを採用しています。開発過程や意思決定のプロセスはメーリングリストやIRCなどを通じてすべて公開されており、

▼表1　OpenStackコミュニティの活動の場

サイト	用途	URL
openstack.org	公式サイト	https://www.openstack.org/
Launchpad	プロジェクト全体の管理	https://launchpad.net/openstack
Wiki	雑多な情報	http://wiki.openstack.org/
EtherPad	議事録等	https://etherpad.openstack.org/
GitHub	ソースコード管理	https://github.com/openstack
Gerrit	レビューシステム	https://review.openstack.org/
メーリングリスト／IRC	質問や提案(Blueprint)	https://www.openstack.org/community/

▼表2　OpenStackリリース履歴

バージョン	リリース名	リリース年月日	特徴
2010.1	Austin	2010年10月21日	Nova/Swift初期リリース。初期は連携できず
2011.1	Bexer	2011年2月3日	Glance追加。Nova/Swiftが連携可能に
2011.2	Cactus	2011年4月15日	VMwareやLXC対応等の機能追加
2011.3	Diablo	2011年9月22日	ZONEスケジューラやレプリケーション等の機能追加
2012.1	Essex	2012年4月5日	Keystone/Horizon追加。統合認証とWebUI操作が可能に
2012.2	Folsom	2012年9月27日	Qunatum/Cinder追加。仮想ネットワーク制御が可能に
2013.1	Grizzly	2013年4月4日	リリース予定

特定ベンダによるクラウドロックインが発生しないように管理されています(**表1**)。そのほか、半年に一度オフラインミーティング(OpenStack Summit)を開催し開発者間のコミュニケーションをとっています。次回Summitは2013年4月15日から4月18日にかけてオレゴン州ポートランドで開催予定です。

●アーキテクチャ

OpenStackの実装にはいくつか特徴があり、巨大なプロジェクトでの連携が円滑に図れるしくみが取り入れられています。詳細は後述する「コンポーネントの基本構造」で解説します。

OpenStackの歴史

OpenStackはNASA(米国航空宇宙局)と、米国のホスティング事業者であるRackspace Hosting社が2010年に立ち上げたプロジェクトとして出発しました。以降、プロジェクトは急速に拡大しましたが、それに伴いコミュニティ活動を行うためのガバナンス体制の弱さや、プロジェクトを創設したRackSpace Hosting社の権限が大き過ぎることが問題となりました。そこで中立の立場から、技術支援や財務支援、知財管理などを行う非営利団体「OpenStack Foundation」が2012年9月19日に設立されました。現在、コミュニティには約80ヵ国から140以上の企業・団体と6,000名以上の参加者を抱えるまでに成長し、日進月歩で今もなお活発な開発が進んでいます(**表2**)。

開発プロセス

OpenStackの開発は次の流れで進められます。

①次期リリース名を投票で決定

OpenStackの主要バージョンには、バージョン番号の他に「リリース名」という名前が付けられ、メーリングリストの議論や質問の中ではこの名前が一般的に使われます。命名方法はリリース順にA→B→C→……とアルファベット順になっており、地名を付けることが慣例です。最新のリリース名はF(olsom)で、現在開発中の次期リリース名はG(rizzly)です。次次期リリースは1月末の投票結果H(avana)に決定しました。

② OpenStack Summit「開発方針／項目を議論」

Summit内では開発者が集まり大枠の開発方針が議論されます。新規機能のアイデアを出し合ったり、現在の問題点と改善案が提示されたりします。提示された案はLaunchpadのBlueprintとして登録されます。

③開発マイルストーンリリース「Blueprintに従った機能追加やバグFIX」

ここから実際にリリースに向けた開発が行われます。開発中は機能を検証しやすくするため、いくつかの実装済みBlueprintをまとめた開発版が3～4回リリースされます。

④ リリース候補版（RC版）リリース「バグFIXのみ」

リリースが近づいてくるとBlueprintの取り込みはストップされ、開発リリースの最終版をベースにしたRC版がリリースされます。ここでは致命的なバグが優先的に修正され、いくつかのバグフィックスをまとめたものをRC1、RC2といった名前でリリースします。

⑤正式リリース

予定されたリリース日になると、実装できなかったBlueprintや細かなバグが残っていても正式版としてリリースされます。これはOpenStackが「リリース日を守る」という点を重要視しているためです。

正式リリースに間に合わなかったBlueprintは次回リリースへと回されます。正式リリースのあとは、すぐに次のSummitが開催され、次期バージョンの議論が開始されます。

⑥メンテナンスアップデート

正式リリース後に見つかったバグの修正や、次期バージョンからバックポートされた項目が反映されたメンテナンスアップデート版がリリースされます。現在の安定版2012.2（Folsom）では、1ヵ月後に［2012.2.1］、2ヵ月後に［2012.2.2］がリリースされています。

OpenStackの構造と機能解説

OpenStackは提供する機能単位で、独立したコンポーネントとして構成されます。開発はコンポーネント単位でそれぞれ別々のプロジェクトで行われ、各プロジェクトの開発には誰でも参加できます。統制をとるためOpenStackコミュニティによって選出されたプロジェクトリーダーとコアチームを中心に運営が行われています。

プロジェクトにはいくつかの種類がありますが、その中心となっているのがOpenStackの主要機能を持つコンポーネント群で、OpenStackプロジェクトに直接管理されており、「コアプロジェクト」と呼ばれています。このコアプロジェクトの周辺に開発・テストをサポートするインフラ関連プロジェクト、マニュアル類を整備するドキュメントプロジェクトなどが存在しています。

その他にOpenStackと連携できる機能を持ったコミュニティベースのプロジェクトがあります。これらのプロジェクトの中には開発が進む

COLUMN 「クラウドって何？」

クラウド、クラウドといわれるようになって久しい昨今ですが、みなさんは「ITインフラのクラウド」という概念をどのようにとらえていますか？

「ITインフラを所有せず、利用する」という認識の人もいれば、「大規模なサービスインフラを効率的に運用する手法」と思っている人もいるかもしれません。また「リソースを専有せず、共有すること」と言う人もいれば、「従来のサーバ仮想化にセルフサービスポータルがついたもの」と考える人もいます。

さまざまな解釈があると思いますが、筆者は「APIを介してアプリケーションのように操作できる、物理的な構成やネットワーク環境に左右されないITインフラ」がクラウドの技術的な側面だと考えています（正確な定義もあります：http://www.nist.gov/itl/cloud/index.cfm）。

クラウド下のITインフラでは、これまで人が手作業で行ってきた作業をプログラムが代行できる、という効率化の側面も持ちます。しかし重要な点として、巨大なリソースをインスタント（即席）で作って・使って・捨てる、という新しい利用方法が可能になることです。これまではコストの問題から手がつけられなかった分野でクラウドが活用され、市場を大きく広げていく可能性につながっていくからです。

現在、クラウドのエコシステムは発展途上の段階にあり、ハードウェア、ソフトウェア、利用方法、コスト概念、様々なものを巻き込み急速に成長しています。次の世代の主流となるか、大きな転換への通過点の一つなのか、その時にならなければ結果はわかりません。

1つの技術要素という枠を超え、業界構造を大きく変化させる可能性を秘めたものがITインフラにおけるクラウドであり、その中心にいるのがOpenStackだと筆者は考えています。

につれてOpenStackコミュニティの承認を受け、インキュベーション段階を経てコアプロジェクトに昇格するものもあります。

コアプロジェクト

現在、7つのコアプロジェクトが登録されています。一般的に「OpenStack」と呼ばれる場合はこれら7つのコンポーネントのことを指します。正式名称は「OpenStack XXX」と命名されていますが、括弧内のコードネームで呼ばれることが一般的なため、本章でもこの解説以降はコードネームを利用します。

● OpenStack Compute（Nova）

ハイパーバイザーを制御し、仮想マシンの管理を担当します。利用可能なハイパーバイザーにはKVM、Xen、LXC、VMware ESXi、Hyper-V、PowerVMなど多岐に渡り、ハイパーバイザーを混在して利用することもできます。Amazon Elastic Compute Cloud[注1]相当の機能を提供します。

● OpenStack Object Storage（Swift）

HTTP REST形式でファイル入出力を行うオブジェクトストレージ機能を提供します。分散環境下でのレプリケーション機能を持ち、ノードの並列配置により性能・容量をリニアにスケールさせることができ、単一障害点を持たない可用性を実現しています。Amazon Simple Storage Service[注2]相当の機能を提供します。

● OpenStack Image Service（Glance）

仮想マシンが利用するイメージテンプレートと、作成された仮想マシンのスナップショットを管理します。ユーザが作成したイメージのアップロードや、ダウンロード機能も提供されます。レプリケーション機能も持ち、別のGlanceサーバへイメージを転送することもできます。

● OpenStack Identity（Keystone）

OpenStack全体の統合認証機構を提供します。テナント、ユーザ、ロールという単位でアクセス権をコントロールします。Keystoneを利用することでコンポーネント自身が独自に認証／認可の機構を備える必要がなくなります。

● OpenStack Dashboard（Horizon）

ユーザ、管理者に対してWeb UIを提供します。セルフサービス形式になっており、ユーザは管理者を介さずに仮想マシンの作成や仮想ネットワークの設定などを行えます。

● OpenStack Networking（Quantum）

仮想ネットワーク機能を提供し、L2/L3ネットワークの管理、利用中のIPアドレス管理、外部公開用のアドレス管理などを行います。さまざまなネットワーク環境をプラグイン形式で制御可能で、Linux Bridge、Open vSwitch、VLAN、SDN、商用ネットワーク装置／ソフトなどを利用できます。

● OpenStack Block Storage（Cinder）

仮想マシンに対してブロックストレージを提供します。仮想マシンは割り当てられたストレージを外付けディスクや、ブート領域として利用できます。Cinderが管理できるストレージは、Linuxが持つiSCSI/LVMをはじめとし、分散ブロックストレージCephやSheepDog、各種商用ストレージと幅広く、用途やサービスレベルによって選択できます。Amazon Elastic Block Store（EBS）[注3]相当の機能を提供します。

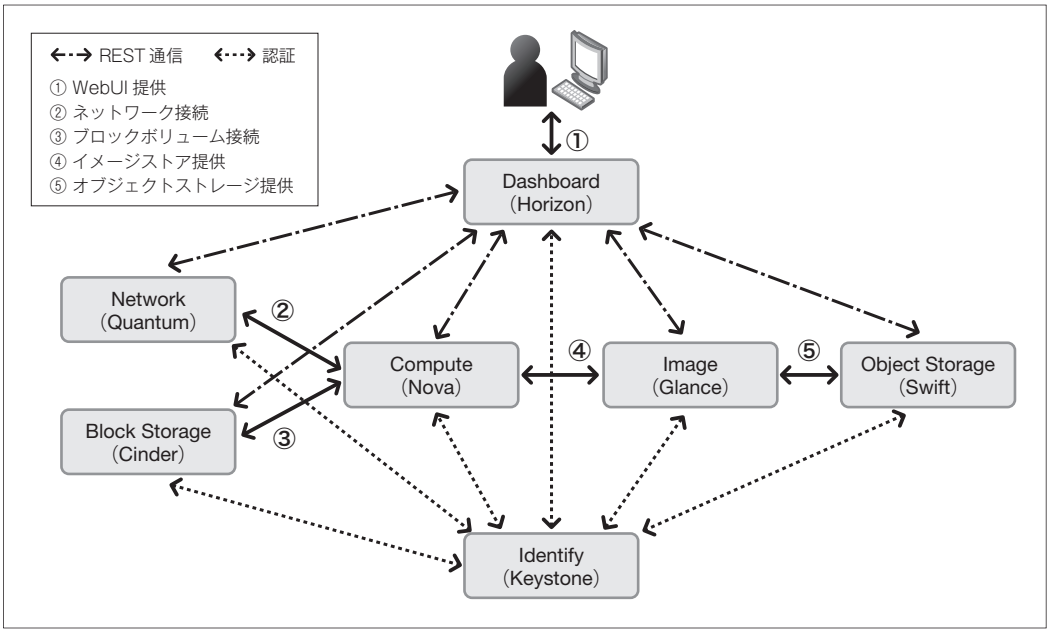
コアコンポーネントは**図1**のように連携をします。

インキュベーションロジェクト

現在インキュベーション状態のプロジェクトは2つ存在します。

注1） Amazon Elastic Compute Cloud（http://aws.amazon.com/ec2/）
注2） Amazon Simple Storage Service（http://aws.amazon.com/s3/）
注3） Amazon Elastic Block Store（http://aws.amazon.com/ebs/）

▼図1　コンポーネント間の連携



● Ceilometer

課金を行うための各種リソースの利用状態を監視します。APIを介して、インスタンスのCPU使用率／メモリ使用率／ディスク使用率、ネットワークパケット流量などの情報を計測／監視をします。

● Heat

Novaインスタンスのオーケストレーション基盤です。OpenStack REST APIとAWS CloudFormation互換APIを介して、複数のコンポジットクラウドアプリケーションを構成するためのサービスです。AWSのCloud Formation相当の機能を提供します。

複数のコンポーネントが存在していますが、利用者はこの中から必要なコンポーネントのみを選択してOpenStack環境を構築できます。ここで紹介した以外にもプロジェクトは多数存在しています。プロジェクトを確認したい場合はOpenStack.org Wiki Projectsページ[注4]をご覧ください。

コンポーネントの基本構造

OpenStackではコンポーネント間やコンポーネント内での連携を円滑に進めるため、実装方法に工夫がされています。ここではその代表的な3つを紹介します。

● WSGIとミドルウェア

OpenStackは各コンポーネントのAPIをWSGI(Web Server Gateway Interface)で実装しています。WSGIはWebサービスの形態の一種で、ミドルウェアと呼ばれるプログラムを複数層重ねて実現されます。

標準で各コンポーネントは複数のミドルウェアを持っており、1つのミドルウェアは1つの機能を担当しています。実現したいサービスに合わせて自由にミドルウェアを選択・組み合わせできます。さらにユーザはWSGIの形式に沿っ

注4）OpenStack.org Wiki Projects(http://wiki.openstack.org/Projects)

てミドルウェアを実装することで、独自の処理を容易に実現できます。

たとえば、各コンポーネントは認証・認可の機能としてリクエストに埋め込まれたトークンの妥当性を検証していますが、これは「authtoken」という名前のミドルウェアとして実装されています。ここを別の認証基盤用に書き換えたミドルウェアに差し替えることで、独自の認証と連携させるといった利用方法もできます。

● Plugin/Driver方式

OpenStackはサービス実現のために各コンポーネントが制御・利用する対象をPlugin/Driver形式で自由に選択できるようにデザインされています。ユーザに対しては実環境を隠蔽した抽象化APIを提供し、その裏側の実動作をPlugin/Driverで定義することで、ユーザサイドからは環境に依存しない操作が可能になり、開発サイドもPlugin/Driverの開発に専念できます。そしてサービス提供者は好みやスキルセットに応じて「ハイパーバイザーにはKVMを利用し、ブロックストレージには商用ディスクを用い、ネットワーク制御にはOpen vSwitchを使う」といった自由度の高い環境構築が可能になります。

● AMQPとRPC

OpenStackはコンポーネント内でのプロセス間通信にAdvanced Message Queuing Protocol（AMQP）を利用しています。AMQPはメッセージキューの規格で、メッセージのFIFO配送や、指定ホストへの配送、キューを参照する全ホストへの配送などさまざまな配送方法をサポートします。AMQP機能を提供する代表的なソフトウェアとして、RabbitMQやApache QPIDなどがあり、どちらもOpenStackから利用できます。

OpenStackではこのAMQPを利用して他のプロセスと通信し、並列配置された分散システムの整合性を保っています。OpenStackの実装では実際のメッセージ操作は「RPC」という名前で抽象化されており、普段はAMQPの存在を意識する必要はありません。

OpenStackの始め方

一とおりOpenStackについて理解したところで、実際にOpenStackを構築してみましょう。OpenStackを始めるにはおもに3つの方法があります。

① GitHubで公開されているソースを利用する

GitHub[注5]で公開されている各種ソースを利用する方法です。最新の機能やバグの修正をいち早く取り込めるなどのメリットがあります。ただし、インストールをするには多くの依存関係を解消する必要があるため構築の難易度は高くなります。

② ディストリビューションが提供するパッケージを利用する

UbuntuやRHEL系ディストリビューションが提供するパッケージを利用する方法です。apt-getコマンドやyumコマンドを実行するだけで依存関係を解決するため、簡単にインストールができます。ただし、パッケージが更新されるまで最新の機能やバグ修正を取り込めません。

Ubuntu 12.04 LTSで提供されるのはEssexバージョンのOpenStackです。Folsomバージョンを利用する場合はCloud Archiveリポジトリを追加します（**図2**）。RHEL系ディストリビューションではEPELリポジトリ[注6]を利用します。

③ 開発者用環境構築スクリプトを利用する

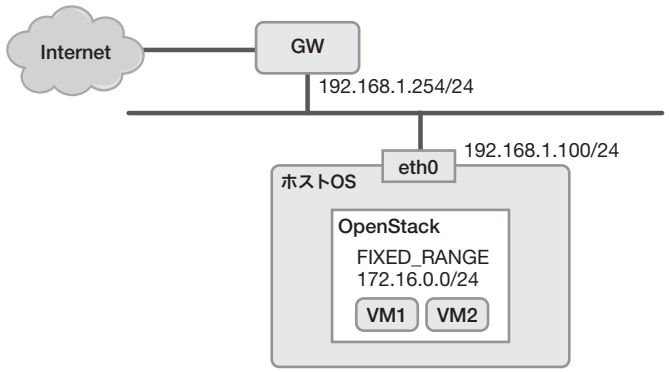
OpenStack開発者が試験環境を構築するために使われるDevStack[注7]を利用する方法です。柔軟な構成をすることはできませんが一とおり

注5） https://github.com/openstack/
注6） http://fedoraproject.org/wiki/EPEL
注7） http://devstack.org

▼図2　Cloud Archiveリポジトリの追加

```
$ sudo apt-get install ubuntu-cloud-keyring
$ echo "deb http://ubuntu-cloud.archive.canonical.com/ubuntu precise-updates/folsom main" | sudo tee /etc/
apt/sources.list.d/folsom.list
$ sudo apt-get -y update
```

▼図3　構築環境イメージ



の機能を試すのに最適です。DevStackはGitHubからOpenStackのソースを取得し、依存関係の解決や設定ファイルの自動生成を行います。現在はUbuntu 12.04以上、Fedora 16以上で動作します。最新バージョンではSUSE Linuxにも対応しました。

DevStackを使用したインストール

DevStackを使用してさっそくOpenStackの最新バージョンFolsomをインストールしてみましょう。今回はNova(仮想マシン制御)、Glance(イメージ管理)、Keystone(認証)、Cinder(ブロックストレージ)、Quantum(仮想ネットワーク)のインストールを行います。Swift(オブジェクトストレージ)のインストールは行いません。

前提条件

今回は**表3**の環境にDevStackをインストールします。ここでは物理サーバ上に構築していますが、Oracle VM VirtualBoxやVMware vSphere Hypervisor上でも動作します。手軽に試す場合はこちらをお勧めします。構築環境のイメージは(**図3**)のとおりです。図中、管理端末は必要に応じて用意します。Ubuntuインストール後は、IPアドレスを固定し、構築の準備を行います(**図4**)。

▼表3　DevStackインストール前提条件

項目	値
CPU	1つ以上(仮想化支援機能を備えたもの)
メモリ	4096MB以上
OS	Ubuntu Server 12.04.01 64bit
NIC(eth0)	192.168.1.100/24
ゲートウェイ	192.168.1.254/24
仮想マシンが利用するネットワーク	172.16.0.0/24

DevStackの入手

DevStackをGitHubから入手します。gitコマンドが必要になるのでインストールされていない場合はgitパッケージをインストールします。今回はFolsomリリース版のOpenStackをインストールするため、DevStackのmasterブランチから最新のソーコードを取得したあとにstable/folsomブランチに切り替えます(**図5**)。

localrcファイルの作成

DevStackは取得したコードに含まれている

devstack/stack.sh を実行すると必要最低限の機能でOpenStackが起動するように設定します。

インストール内容をカスタマイズする場合はdevstackディレクトリ内にlocalrcファイルを作成してその中に設定項目を記述します。localrcファイルの詳しい説明はDevStackのサイトで確認できます[注8]。**リスト1**では各種パスワード設定、サービスの有効化／無効化、Novaの設定、ネットワークの設定を行います。

▼図4　IPアドレスの設定

```
$ sudo vim /etc/network/interfaces
auto eth0
iface eth0 inet static
    address 192.168.1.100
    netmask 255.255.255.0
    network 192.168.1.0
    broadcast 192.168.1.255
    gateway 192.168.1.254
    dns-nameservers 192.168.1.254

$ sudo /etc/init.d/networking restart
```

DevStackを使用したOpenStackインストール

localrcを作成したらstack.shを実行します[注9]。localrcの設定に誤りがなければ、**図6**のような出力がされます。通常OpenStackのプロセスはデーモンとして起動しますが、DevStackを使用した場合は、GNU Screenが起動し仮想コンソール上でOpenStackの各プロセスが起動しています。各プロセスのログは仮想コンソールの標準出力とSCREEN_LOGDIRに設定したディレクトリに表示されます。GNU Screenの使い方に

▼図5　DevStackソースの取得

```
$ sudo apt-get -y install git
$ cd ~ && git clone https://github.com/openstack-dev/devstack.git
$ cd ~/devstack
$ git checkout stable/folsom ←stable/folsom ブランチに変更
$ git branch ←ブランチを確認
  master
* stable/folsom
```

▼リスト1　各種設定

```
$ vim ~/devstack/localrc
HOST_IP=192.168.1.100 ←自ホストIPアドレスを指定
ADMIN_PASSWORD=openstack ←パスワードをopenstackに設定
MYSQL_PASSWORD=$ADMIN_PASSWORD ←変数も利用可能
RABBIT_PASSWORD=$ADMIN_PASSWORD
SERVICE_PASSWORD=$ADMIN_PASSWORD
SERVICE_TOKEN=admintoken ←サービス管理用トークンを設定
disable_service n-net ←Quantum 利用のため、Nova Networkサービスを無効化
disable_service n-obj ←Nova Object サービスを無効化
enable_service q-svc ←Quantum サービスを有効化
enable_service q-agt ←Quantum エージェントを有効化
enable_service q-dhcp ←Quantum DHCPエージェントを有効化
enable_service q-l3 ←Quantum L3 エージェントを有効化
ENABLE_TENANT_TUNNELS=True ←GREのトンネリングを有効化
FIXED_RANGE=172.16.0.0/24 ←仮想マシン用プライベートネットワークを設定
NETWORK_GATEWAY=172.16.0.254 ←仮想マシンが使用するゲートウェイを設定
FLOATING_RANGE=10.0.0.0/24 ←外部から直接仮想マシンに接続するためのIPアドレスレンジを指定
LOGFILE=stack.sh.log ←インストールログ出力ファイル名を指定
SCREEN_LOGDIR=/opt/stack/logs ←screen ログの保存先を指定
```

注8）http://devstack.org/localrc.html
注9）DevStackがインストールするソフトウェアの中にはGitHubにコミットされた最新のソースを利用するものがあるため、取得タイミングによっては正常に動作しない可能性があります。経験則ですがDevStackの問題はコミュニティ側ですぐに修正されるので、後日インストールを再実行すると成功する場合が多いです。またある程度Python、シェルスクリプトに精通している方はインストールログ（stack.sh.log）から問題を特定して修正することもできます。

▼図6　DevStackインストール正常終了時出力

```
$ cd ~/devstack && ./stack.sh
…… (中略) ……
Horizon is now available at http://192.168.1.100/
Keystone is serving at http://192.168.1.100:5000/v2.0/
Examples on using novaclient command line is in exercise.sh
The default users are: admin and demo
The password: openstack
This is your host ip: 192.168.1.100
stack.sh completed in 749 seconds.
```

ついては、man screenをご確認ください。

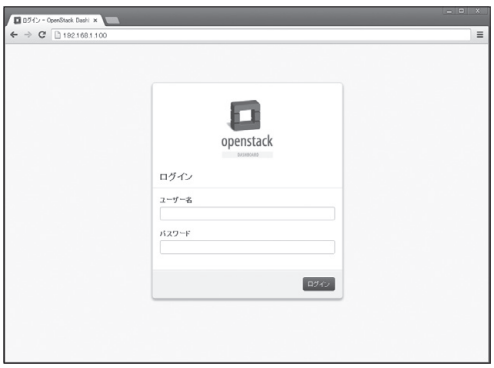
Horizon（DashBoard）を使用したOpenStackの操作

図6に表示されたURLにアクセスするとHorizonにアクセスできます（**図7**）。一般ユーザでログインする場合はID「demo」、管理ユーザでログインする場合はID「admin」を使用します。パスワードはいずれもlocalrcで設定したパスワード「openstack」を使用します。

コマンドを使用したOpenStackの操作

各コンポーネントを操作するコマンドはNovaならnova、Quantumならquantumというようにプロジェクト名がそのままコマンドになっています。どのコマンドもオプションが豊富ですので詳細はオンラインヘルプを参照してください。コマンドラインから操作を行う場合は事前にいくつかの環境変数設定しておく必要があります。DevStackを利用した場合、これらの環境変数を一括設定してくれるdevstack/openrcファイルが作成されます。このファイルを読み込むことで、テナント「demo」、ユーザ「demo」でのコマンド実行が可能になります。管理者操作を行う場合は、環境変数（OS_USERNAME）にadminを設定します。操作したいテナントを変更したい場合は環境変数（OS_TENANT_NAME）を対象テナント名に変更します（**リスト2**）。

▼図7　Horizonログイン画面



インスタンスを起動する

DevStackで構築した環境で実際にコマンドラインからインスタンスを起動してみましょう。ここではテナント「demo」に対してユーザ「demo」を使用してインスタンスを作成します。実行例を**図8**に記載します。インスタンスの起動には、最低限「元となるイメージのID」「起動するインスタンスサイズ（Flavor）のID」「使用するネットワークID」「セキュリティグループ名」が必要です。その他、公開鍵認証を行うためのキーペ

▼リスト2　環境変数の設定

```
$ source ~/devstack/openrc
$ export OS_USERNAME=admin ←管理操作が必要な場合
$ export OS_TENANT_NAME=admin ←テナントの切り替えが必要な場合

以下のように引数を渡して実行することも可能
$ source ~/devstack/openrc admin admin ←ユーザ名（OS_USERNAME）、テナント名（OS_TENANT_NAME）の順で指定
```

▼図8　コマンドラインからインスタンスを起動する

```
$ source ~/devstack/openrc demo demo ←環境変数設定(ユーザ: demo、テナント: demo)
$ glance image-list ←登録済みイメージの確認
+--------------------------------------+----------------------------------+-------------+------------------+----------+--------+
| ID                                   | Name                             | Disk Format | Container Format | Size     | Status |
+--------------------------------------+----------------------------------+-------------+------------------+----------+--------+
| daacfe62-578a-4d01-ac63-de024e6835a0 | cirros-0.3.0-x86_64-uec          | ami         | ami              | 25165824 | active |
| 35785ba9-ab55-4a2e-9faa-a2644459691b | cirros-0.3.0-x86_64-uec-kernel   | aki         | aki              | 4731440  | active |
| 0b2e2d8a-1e9e-43ec-88bd-4eecf4306a7c | cirros-0.3.0-x86_64-uec-ramdisk  | ari         | ari              | 2254249  | active |
+--------------------------------------+----------------------------------+-------------+------------------+----------+--------+
$ nova flavor-list ←Flavor の確認
+----+-----------+-----------+------+-----------+------+-------+-------------+-----------+-------------+
| ID | Name      | Memory_MB | Disk | Ephemeral | Swap | VCPUs | RXTX_Factor | Is_Public | extra_specs |
+----+-----------+-----------+------+-----------+------+-------+-------------+-----------+-------------+
| 1  | m1.tiny   | 512       | 0    | 0         |      | 1     | 1.0         | True      | {}          |
| 2  | m1.small  | 2048      | 20   | 0         |      | 1     | 1.0         | True      | {}          |
| 3  | m1.medium | 4096      | 40   | 0         |      | 2     | 1.0         | True      | {}          |
| 4  | m1.large  | 8192      | 80   | 0         |      | 4     | 1.0         | True      | {}          |
| 5  | m1.xlarge | 16384     | 160  | 0         |      | 8     | 1.0         | True      | {}          |
+----+-----------+-----------+------+-----------+------+-------+-------------+-----------+-------------+
$ quantum net-list ←利用可能なネットワークを確認
+--------------------------------------+---------+--------------------------------------+
| id                                   | name    | subnets                              |
+--------------------------------------+---------+--------------------------------------+
| 7a57c06b-2a70-4452-8403-7d660bf54585 | ext_net | 3fd9a7d5-0f88-41ed-b27f-002ce39d91bd |
| a30eb4c4-9ad9-48b5-81e2-a6c4f1497dd2 | net1    | 00eca97f-08af-4a75-90b4-883e6cb192c2 |
+--------------------------------------+---------+--------------------------------------+
$ nova boot --flavor m1.tiny --image daacfe62-578a-4d01-ac63-de024e6835a0 --security-groups default --nic net-id=
a30eb4c4-9ad9-48b5-81e2-a6c4f1497dd2 demoVM ←インスタンスを起動
$ nova list ←インスタンスリストを取得
+--------------------------------------+--------+--------+-----------------+
| ID                                   | Name   | Status | Networks        |
+--------------------------------------+--------+--------+-----------------+
| d519dc0b-6801-4c7d-8b72-e8b20baed471 | demoVM | ACTIVE | net1=172.16.0.2 |
+--------------------------------------+--------+--------+-----------------+
$ nova secgroup-add-rule default tcp 22 22 0.0.0.0/0  ←SSH通信を許可
$ nova secgroup-add-rule default icmp -1 -1 0.0.0.0/0 ←ICMP通信を許可
$ nova secgroup-list-rules default ←セキュリティグループに登録したルールを確認
+-------------+-----------+---------+-----------+--------------+
| IP Protocol | From Port | To Port | IP Range  | Source Group |
+-------------+-----------+---------+-----------+--------------+
| icmp        | -1        | -1      | 0.0.0.0/0 |              |
| tcp         | 22        | 22      | 0.0.0.0/0 |              |
+-------------+-----------+---------+-----------+--------------+
$ ping -c 3 172.16.0.2 ←疎通確認
$ ssh cirros@172.16.0.2 ←SSH 接続 (パスワード「cubswin:)」)
$ exit ←インスタンスからログアウト
$ nova delete d519dc0b-6801-4c7d-8b72-e8b20baed471 ←不要なインスタンスを削除
```

ア登録や、アタッチするブロックストレージの指定などができます。

●① 環境変数設定

コマンド操作実行の前に環境変数を読み込みます。環境変数設定個別設定ではなく~/devstack/openrcをsourceコマンドで読み込みます。ここでは同時にユーザとテナントを指定します。

●② 起動イメージの確認

Glanceに登録されてた起動イメージをコマ

ンド「glance image-list」で確認します。ここでは「Disk Format」が「ami」となっているイメージ「cirros-0.3.0-x86_64-uec」を指定します。

●③ Flavorの確認

インスタンスのスペックを指定するFlavorをコマンド「nova flavor-list」で確認します。

●④ ネットワーク

テナント「demo」が権限を持つネットワークをコマンド「quantum net-list」で確認します。「ext_net」はFloatingIPを使用するための特殊なネットワークで、ここでは利用しません(「参考 FloatingIPを利用する」参照)。

●⑤ インスタンスの起動

インスタンス起動に必要なパラメータを指定し、コマンド「nova boot」でインスタンスを起動します。セキュリティグループ名は「default」を使用します。起動イメージのIDはイメージ名を指定することもできます。

・Flavor のIDまたは名前：m1.tiny
・起動イメージのID：daacfe62-578a-4d01-ac63-de024e6835a0(※cirros-0.3.0-x86_64-uec)
・セキュリティグループ名：default
・利用するネットワークのID：a30eb4c4-9ad9-48b5-81e2-a6c4f1497dd2(※net1)
・インスタンスの表示名：demoVM

起動したインスタンスをコマンド「nova list」で確認します。「Status」が「ACTIVE」になっていれば正常に起動しています。ただし、インスタンスとしてアクティブになるだけで実際にOSの起動が完了するまで時間がかかります。「BUILD」の場合はインスタンス生成中となっています。「ERROR」の場合はインスタンスの生成に失敗しています。GNU Screenの仮想コンソールに出力されるログから原因を確認し、再実行してください。

●⑥ セキュリティグループの設定

初期状態では外部からインスタンスに対してネットワークアクセスが行えません。確認のため、ICMPおよびSSH通信を許可するルールをコマンド「nova secgroup-add-rule」でセキュリティグループ「default」に追加します。追加したルールは「nova secgroup-list-rules」コマンドで確認できます。

●⑦ 接続確認

「nova list」で表示されたインスタンスのIPアドレスに対して「ICMP ECHO_REQUEST」の発行とSSH接続を行います。インスタンスのユーザIDは「cirros」、パスワードは「cubswin:)」です。VNC接続を行う場合、キーマップの関係で「:(コロン)」が入力できない場合があります。

●⑧ インスタンスの削除

不要なインスタンスはコマンド「nova delete」で削除できます。

参考「FloatingIPを利用する」

AWS EC2でいうElastic IPアドレスに相当する機能です。外部ネットワークから直接仮想マシンにアクセスするための機能を提供します。DevStackが標準で作るFloatingIPは、自ホストからしかアクセスできません。FloatingIPを利用して外部からインスタンスに直接アクセスする場合はOpen vSwitchの設定なども必要になります。FloatingIPの利用例を**図9**に記載します。

再起動時の対応

DevStackにはいくつか揮発性の設定が含まれているため、Ubuntuを再起動した場合は次の設定を手動で行う必要があります。また、DevStackで作った環境はOSの再起動を行っても、自動的に各OpenStackプロセスが起動しません。起動するには、rejoin-stack.shスク

▼図9　コマンドラインからFloatingIPを利用する（※2013年1月1日時点のHorizonからFloatingIPを直接操作することはできません）

```
$ source ~/devstack/openrc admin demo ←環境変数設定（ユーザ：admin、テナント：demo
$ quantum quantum net-external-list -c name -c subnets ←demoテナントが利用可能なFloatingIPとそのサブネットIDを確認
$ quantum subnet-show SUBNETS ←FloatingIPの状態を確認（利用可能なIPアドレスの範囲、ゲートウェイなど）
$ quantum floatingip-create ext_net ←FloatingIPを作成
$ quantum floatingip-list ←作成した FloatingIPの確認
$ quantum port-list ←FloatingIPを紐付けるインスタンスのポートID確認
$ quantum floatingip-associate FLOATINGIP_ID PORT_ID ←FloatingIPとPORT_IDをマッピング
$ quantum floatingip-list ←紐付けたFloatingIPの確認
```

▼図10　OS再起動後にDevStackを再起動する（※IPアドレスは環境によって変わる可能性があります。正確な情報は~/devstack/stack.sh.logを参照してください）

```
$ sudo ip addr add 10.0.0.1/24 dev br-ex ←br-ex インターフェースの活性化
$ sudo ip link set br-ex up

$ sudo route add -net 172.16.0.0/24 gw 10.0.0.2 ←ルーティング情報の追加
$ cd ~/devstack && ./rejoin-stack.sh
```

リプトを使用します(**図10**)。

本番環境の構築

DevStackを利用することで簡単にOpenStackを体験できます。しかし、OpenStack本番環境を構築をする場合にDevStackは柔軟性が乏しいためお勧めしません。代わりにソースコードからのインストールや各OSディストリビューションから提供されるパッケージの利用をお勧めします。その他にも各社が提供するOpenStackディストリビューションを利用する方法や、OpenStackが組み込まれたアプライアンス製品を利用する方法もあります。利用する目的や要件に合わせてインストールの形式、コンフィグ設定やコンポーネント冗長化、ストレージの選択など、自身の環境にあった構成を見つけ出してください。何か困ったことがあったときはOpenStackコミュニティや後述する日本OpenStackユーザ会に気軽に質問してみるのも良い選択です。質問のひとつひとつがOpenStackを成長させる重要な鍵になります。

OpenStackと付き合っていく上で、1つだけコツを挙げるとすれば、構築する際、『欲張らない』ことです。初めから「機能ごとにサーバを分ける」や「特殊なストレージと連携する」などと欲を出すとうまくいかなかったときにどこで問題が発生しているのかが特定できず、貴重な時間を浪費してしまいます。DevStackで勘所をつかみ、小さくはじめて、どういうしくみか理解したうえで発展した構成での構築することが近道です。

OpenStackを使ったサービス

ここではOpenStackが使われているサービス事例を紹介しながら、それぞれのサービスでどのようにOpenStackが利用されているか紹介します。

OpenStackをフルスタックで利用している事例

Rackspace®Cloud Servers™/CLOUD Files™（http://www.rackspace.com/cloud/files/）

OpenStackの創設者である米Rackspace Hosting社が提供するサービスです。Cloud ServersはNova/Glance/Quantumを中心に構築されており仮想サーバサービスを提供しており、CLOUD FilesはSwiftを利用したオブジェクトストレージサービスを提供しています。そのほかにも、MySQLを利用できるCloud Databaseや、OpenStackを自社データセンター内で利用可能にする「Private Cloud」サービスなどが提供

されています。

HP Cloud Services（https://www.hpcloud.com/）

米Hewlett-Packard社が提供するIaaSサービスで、OpenStackの全機能を利用した仮想マシンサービス、オブジェクトストレージサービスなどを提供します。2つのリージョン（米国西海岸、米国東海岸）を持ち、それぞれのリージョンには3つのデータセンターが所属しています。データセンターごとにAvailability Zone（他のZoneと物理的な設備を共有しない単位）が分けられており、大規模なサービスとして展開されています。OpenStackが提供する機能以外にもデータベース、アプリケーション実行環境など豊富なサービスを備えています（**図11**）。

OpenStackの一部を利用している事例

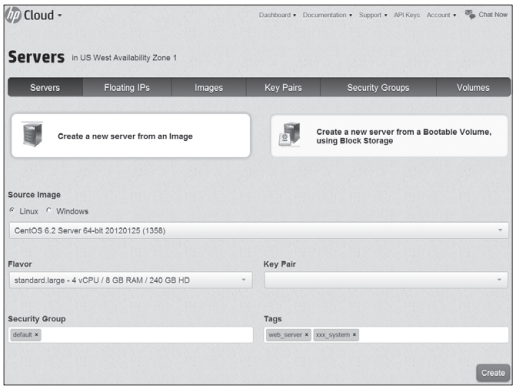
GMOインターネット お名前.com VPS（http://www.onamae-server.com/vps/）

GMOインターネット社による「お名前.com VPS」のサービス基盤にOpenStackが採用されています。このサービスではNovaの仮想マシン管理の機能の一部を使い大規模環境を管理しており、周辺を自社開発のシステムと連携させることで効率よくサービス開発をしています。内部の技術情報が公開されており、サービス開発する上で有益な情報源となります（http://www.onamae-server.com/blog/）。

San Diego Supercomputer Center（https://cloud.sdsc.edu/hp/index.php）

サンディエゴ・スーパーコンピューターセンターではシミュレーションやHPCで扱う大量データのストア先として5.5PBの容量を持つSwiftを利用しています。Swiftは安全・安価に大容量が確保できるだけでなく、汎用サーバを並列に配置することでリニアに性能向上ができるため、高価なストレージシステムを用いることなく高速なデータ入出力にも対応できます。本センターのSwift環境では8〜10GB/secのファイル読み込み性能が報告されています。

▼図11　HP Cloud Servicesインスタンス作成画面



OpenStackを基盤として、PaaS/SaaSを提供している事例

Cisco WebEx（http://www.webex.com/）

WebExはCisco社のサービスとして提供されるWeb会議システムのSaaSで、本サービスはOpenStack上で稼働しています。

このようにWeb系システムやそのほかにもSNS系のようなサービスをOpenStack上で稼働している例は公開されていませんが数多く存在しています[注10]。その理由として、OpenStackを利用することでオープンかつスケーラブルな環境を構築できるだけでなく、物理環境の違いに影響されない標準化されたクラウドインフラストラクチャを利用できるようになります。これにより開発者と運用者はアプリケーションの開発・テスト・ディプロイ・エンハンス・スケールなどの作業において良好なDevOpsを実現できるためです。

その他の例

紹介した事例のようにサービスとして利用され

注10） 公開事例が少ないのは、アプリケーションサービスで勝負する企業は基盤にOpenStackを採用しているという事実を公表するメリットがないためです。

るケースのほか、社内プライベートクラウド基盤として採用されるケースや、研究機関や企業内の開発・検証・実験環境で利用されるケースも多々あります。OpenStackはフルスタックで利用することも、使いたい機能だけに絞って特定コンポーネントのみを利用することもできます。利用用途もOpenStackそのものの機能をサービスとして提供したり、OpenStackを基盤としてアプリケーションサービスを提供することもできます。こうした点がクラウド分野においてOpenStackが支持される理由の1つになっています。紹介した事例以外にも多数の事例が公式サイトのUser Stories（https://www.openstack.org/user-stories/）にて公開されています。

コミュニティ紹介

ユーザ会の概略

最後に日本のOpenStackコミュニティについて紹介します。日本語によるOpenStackに関する情報発信、情報共有を行い、国内でのOpenStackの普及、人材育成に貢献を目的として日本OpenStackユーザ会（以下JOSUG）が2010年10月22日に設立されました。OpenStackプロジェクトの拡大に合わせて国内のコミュニティも急速に拡大し、2013年1月現在でJOSUGは17の企業・団体、800名以上が参加しています（**表4**）。

活動紹介

現在はオープンソースカンファレンスへの参加や、1ヵ月に一度を目標に首都圏で開催している勉強会が主な活動となっています。勉強会は、普及を目的とした入門レベルの勉強会と、OpenStack利用者・開発者を対象にしたハッカソンを交互に実施しています（**表5**）。他にもメーリングリストでのQ&A対応によるナレッジの蓄積や、他団体との連携した活動（オープンクラウド実証実験タスクフォース http://www.ocdet.org/）など、その他Advent Calendarの開催なども行っています。今後もこれまで同様に活動を続けていきます。直近では2013年3月12日に「OpenStack Day Tokyo 2013」という大規模なイベントを開催します。

このイベントでは最新のOpenStack状況とそのエコシステムを紹介することで、クラウドを作りたい人・クラウドを利用したい人すべてにOpenStackを知っていただき、OpenStackの普及と国内のクラウド事業を促進することを目的としています。OpenStack初心者の方から、すでに利用している方まで興味のある方であれば誰でも参加できます。本イベントの今後の予定についてはイベント告知ページ（http://openstackdays.com/）をご覧ください。

まとめ

さまざまな調査会社の資料から今後のIT市場はクラウドを中心に発展していくと予想されています。これはクラウドというものが「所有から利用へ」という話ではなく、「インターネッ

▼表4　OpenStackコミュニティ活動の場

用途	URL
公式サイト	http://openstack.jp
メーリングリスト	http://groups.google.com/group/openstack-ja

▼表5　JOSUGの活動

JOSUGの活動	URL
これからのイベント告知	http://openstack.jp/events.html
これまでのイベント報告	http://openstack.jp/news.html
Advent Calendar 2012JP	http://openstack.jp/documents/adventcalendar-2012.html

COLUMN
「地雷原を行く」

筆者がはじめてOpenStackに触れたのはDiablo（OpenStackリリースから1年後4回目のリリース）が出てすぐのころでした。当時のコンプロジェクトはNova、Glance、Swift、Horizon、そしてDiabloリリース1週間前に突如組み込まれたKeystoneしかありませんでした。

Novaなどはコンポーネント独自の認証をKeystoneに変更したことで起こる問題などで、OpenStacker達を苦しめました（JOSUGのメンバの1人は、あまりに動かないKeystoneを漬物石と揶揄したほどでした）。

また、コンポーネントにも問題があり「ことごとく動かない」を経験し、OpenStackを触っている人たちは「変態」だと思い知らされました。新機能を試すと高確率で何かのバグを踏み抜く様はまるで「地雷原で前転を繰り返す」ような行為でした。

しかし、これらの問題に心折られなかったOpenStackerたちが根拠よく改善に向けた議論をしたり、バグを報告・修正したりすることでOpenStackは機能を追加しながらも徐々に安定を手にしてきました。これはOpenStackがリリース直後から注目されていたこともそうですが、コミュニティがコミュニティとして機能していたからこそなせた業ではないかと考えています。もし、OpenStackが誰からも注目されず、非協力的なコミュニティであったのなら1年持たず廃れてしまったかもしれません。OpenStackがリリースされて約2年、OpenStackはシステム的にもコミュニティ的にも急成長しました。FolsomバージョンではIaaSが必要としている機能は粗方出そろい、これからの開発では新機能追加はもちろん、安定性向上や運用面にも目が向けられて行きます。

まだまだOpenStackは成長段階です。機能の追加やさらなる利便性や安定性の向上、もしかすると大きな機能を根本から考え直さないといけないような問題にぶち当たるかもしれません。しかし、今触れれば多くのユーザと一緒にOpenStackを育てることができます。これまで使うだけだったクラウドを育てるチャンスは多くはありません。開発だけでなく、質問することも財産になります。門は常に開かれています。本章を通して興味を持たれた方がいらっしゃいましたら一緒にOpenStackを育てていきましょう！（地雷もまだ埋まっています）。

ト上に分散したリソースを束ねて巨大なリソースとして利用する」という現在のアーキテクチャを大きく変えていく概念だと考えられており、世の中が徐々にこの方向へシフトしていくことを予想したものです。グリッドコンピューティングやHPC（High Performance Computing）分野において長年取り組まれてきた領域ですが、その実現にもっとも近い場所にいるのがOpenStackです。

OpenStackのAPIに仮想マシン作成やストレージ領域作成のリクエストを出すと、その仮想リソースが確保されると同時にリソースを操作するための「アドレス」を含んだ情報が返されます。この挙動は1台のパソコンの上でOSのシステムコールを使ってファイルオープンやメモリ確保の命令を出した場合の動きととてもよく似ています。「OpenStackはクラウドのLinuxを目指している」と言われることがありますが、これはエコシステムだけの話ではなく、技術的にもOpenStackのAPIは「クラウドのシステムコール」を実装しようとしていると言えます。

現在のOpenStackでは作成された仮想リソースを、従来どおり単一の仮想マシンやディスク領域として使う方法が一般的です。しかし今後はデータベースやアプリケーションサーバなどのさまざまなソフトウェアがOpenStackのAPIを直接操作し、自分自身が動作する環境を自律的に操作していくことで、人手を介さずに性能や可用性がコントロールされるようになっていくと考えられます。

近い将来を見たときには「現在普及しているサーバ・ストレージの仮想化を効率的に実現する現実的なシステム」として、さらにその先の未来を見たときには「分散コンピューティングを実現する野心的なシステム」として、このように現在と未来をカバーする技術的な背景があるからこそOpenStackは全世界で注目を集めています。

本章を読んでOpenStackに興味を持たれた方がいらっしゃいましたら、ぜひともOpenStackの世界へ踏み出していただければと思います。SD

第1特集
もっとクラウドを活用してみませんか？

Part 2

VirtualBoxとDevCloudを使って15分でIaaS構築!

CloudStackを試してみませんか?

IaaS型クラウドを構築できるオープンソースソフトウェアとして、Part1で紹介したOpenStackと人気を二分するのが「CloudStack」です。Part2では、このCloudStackの特徴や構造を知り、VirtualBoxに取り込むだけで始められるDevCloudを使った簡単なIaaS構築手順を解説します。

寺門 典昭　TERAKADO Noriaki　(株)IDCフロンティア

CloudStackとは?

CloudStackは今とても勢いのあるオープンソースのクラウド基盤構築ソフトウェアです。日本の事業者でも、IDCフロンティア、SCSK、NTTコミュニケーションズ、KDDIなどが採用し、IaaS(Infrastructure as a Service)の提供を開始しています。2010年に登場して間もないオープンソースを日本の主要なサービスプロバイダが続々と採用を決めている事実が、CloudStackに対する期待を表しています。

早速、実際にどんなところが評価されているのかCloudStackの特徴を見ていきましょう。

CloudStackの特徴を学ぼう

リッチなユーザインターフェース

オープンソースといえば、テキストベースのコンフィグレーションを思い浮かべる方も多いのではないでしょうか。CloudStackではすべての設定がグラフィカルな画面で行えるのです。さらに、仮想化しただけでは把握しづらかったサーバのCPU、メモリ、ストレージやIPアドレスのリソースなどの利用率をとてもわかりやすく表示してくれます(**図1**)。

インフラを管理する人にとって視覚的に表現されていることは重要ですよね。このようなリッチで実用性のあるインターフェースが人気を集めた1つの要因ではないでしょうか。

複数のハイパーバイザに対応

CloudStack は VMware vSphere、Citrix XenServer、KVM、Oracle VMをサポートしています(2013年1月現在)。さらにMicrosoft Hyper-V(2012)や LXC(Linux Containers)などの追加検討が進められており、今後もっと多くのハイパーバイザのサポートが期待できます。

IaaSの提供を生業とする筆者には、複数のハイパーバイザに対応していることがとても重要です。選択肢が1つだけとか少ないものでは、多様化する要求に応えられない場合も出てきます。各ハイパーバイザにはそれぞれ特徴があり、必要に応じて使い分けるのが一般的になってきていますし、今後もその傾向が強くなるでしょう。

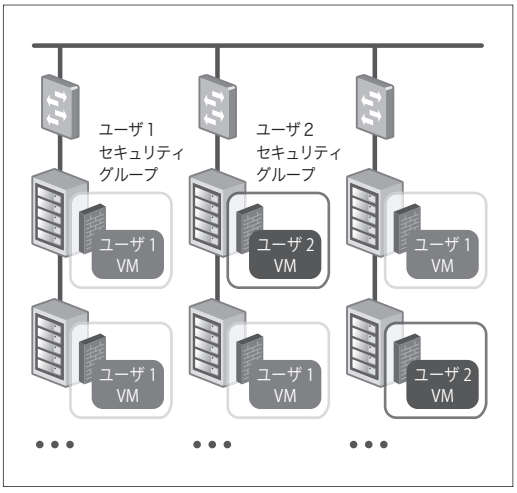
▼図1　管理者ダッシュボード

豊富なネットワークパターンと機能

CloudStackでは大きく分けて2つのネットワークタイプが存在します。ユーザアカウント(以降、アカウント)の仮想マシンが同じネットワークをシェアする「Basicゾーン」(**図2**)と、アカウントごとにVLANでネットワークを隔離する「Advancedゾーン」(**図3**)があります。

この2つのネットワークタイプでは、セキュリティの考え方が異なります。Basicゾーンでは、仮想マシン自身のファイアウォール(iptablesやWindowsファイアウォールなど)で行うか、またはCloudStackのセキュリティグループを使って仮想マシンごとに制御します。一方、Advancedゾーンでは、仮想ルータを使ってネットワーク単位でセキュリティの確保を行います。

▼図2　Basicゾーン



Advancedゾーンのメリットの1つとして、各アカウントごとの専用仮想ルータを使ってVPNやロードバランシングを利用できることが挙げられます。サーバ側の知識はあるのにネットワークには弱いという人が実は結構多いのではと感じているのですが、そんな人たちにとってネットワーク機能が簡単に利用できるのは、とてもうれしいのではないでしょうか。仮想ルータだけでなく、物理ファイアウォールや物理ロードバランサも同様にCloudStackで管理することができるため、より拡張性のあるネットワークを構築できます。

実用的なAPI

API(Application Program Interface)は、IaaSに必須な機能といえるでしょう。Webサーバのオートスケールなど人を経由しない運用管

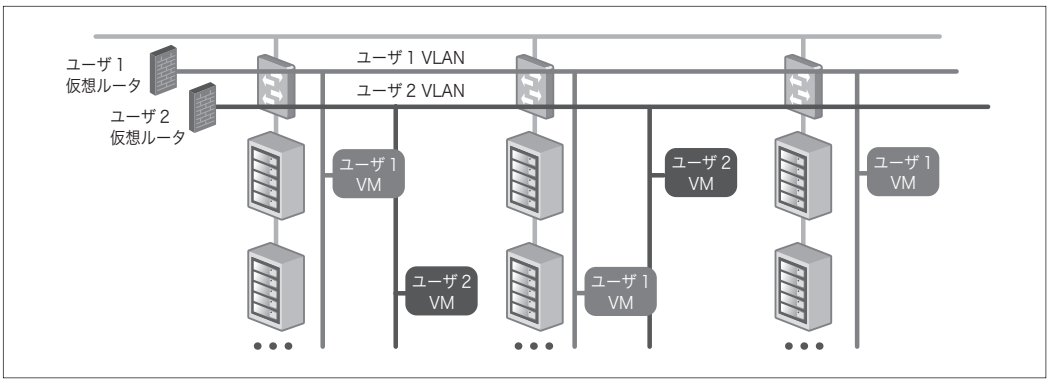
COLUMN
セキュリティグループとは

仮想マシンが稼働するハイパーバイザ側でフィルタリングを行う機能で、ファイアウォールルールセットのこと。一部のハイパーバイザのみ対応しています。

COLUMN
サポートされる物理アプライアンス

ファイアウォールにはJuniper SRXが、ロードバランサにはCitrix NetScalerとF5 Big-IPがサポートされています(2013年1月現在)。

▼図3　Advancedゾーン



理を実現できたり、外部システムとの連携にも密にかかわっています。たとえば、RightScaleやScalrなどのクラウド管理サービスでは、APIだけを利用してCloudStackのクラウド基盤を操作しています。

APIについては、後半で実際に試してみますのでぜひご覧ください。

他製品との連携もすごい

CloudStackはプラグインの開発やAPIを公開することで、さまざまな製品との連携を可能にしています(**表1**)。筆者は単独のソフトウェアが大きくなるよりも、他製品を含めたエコシステムとして広がっていくソフトウェアを好んで使います。エコシステムが大きくなるソフトウェアは周りからも評価され、魅力のある証拠だと考えるからです。

少し話は逸れますが、筆者がCloudStackを知ったのは2010年の秋頃でした。初めてCloudStackの管理者ガイドを読んだとき、とても感動したことを覚えています。インフラに関するアーキテクチャがしっかりしていて、周辺(ネットワーク装置やストレージなど)のコンフィグレーションまで具体的に書かれていてました。とくにネットワーク部分のインテグレーションの完成度の高さに驚きました。こんな情報を無償で公開して良いのかと思ったほどです。未だクラウドがバズワードとも言われていた時期でしたが、それを読んで本当のクラウド(IaaS)を実現できる時代になると確信しました。その直後に、知人の伝でCloudStackの生みの親であるSheng Liang氏(当時の開発元Cloud.comのCEO)に会うことができ、CloudStackのロードマップなどを聞かせてもらいました。そのとき聞いた考え方や目指している方向などに大いに共感し、これがきっかけでIDCフロンティアはCloudStackを採用することを決めています。

なお、このCloudStackの管理者ガイド(Cloud Stack Administration Guide)[注1]は現在も公開されていますので、ぜひ読んでみてください。一見の価値があります。

▼表1　連携可能な製品／サービス(予定含む)

カテゴリ	製品／サービス名
分散ストレージ	Swift、Caringo、Riak CS(Basho)、Cloudian、Ceph RBD
クラウド統合管理ツール	RightScale、Scalr、UForge
サーバ運用管理ツール	Puppet、Chef
監視ツール	VMTurbo、Zabbix、Munin
ネットワーク	Nicira、Big Switch、Nexus 1000v

CloudStackの主なコンポーネントを理解しよう

図4はCloudStackのネットワーク構成例です。ここでは、CloudStackのコンポーネントがどんな役割をもっているか1つずつ確認していきましょう。

マネージメントサーバ

CloudStackの中枢がこのマネージメントサーバです。すべてのリソース(物理リソースや仮想マシンなど)を管理し、ユーザインターフェースやAPIを提供します。データベースには広く使われているMySQLを利用しています。

ホスト

ハイパーバイザが稼働するサーバで、仮想マシンに必要なリソース(CPU、メモリ)を提供します。

プライマリストレージ

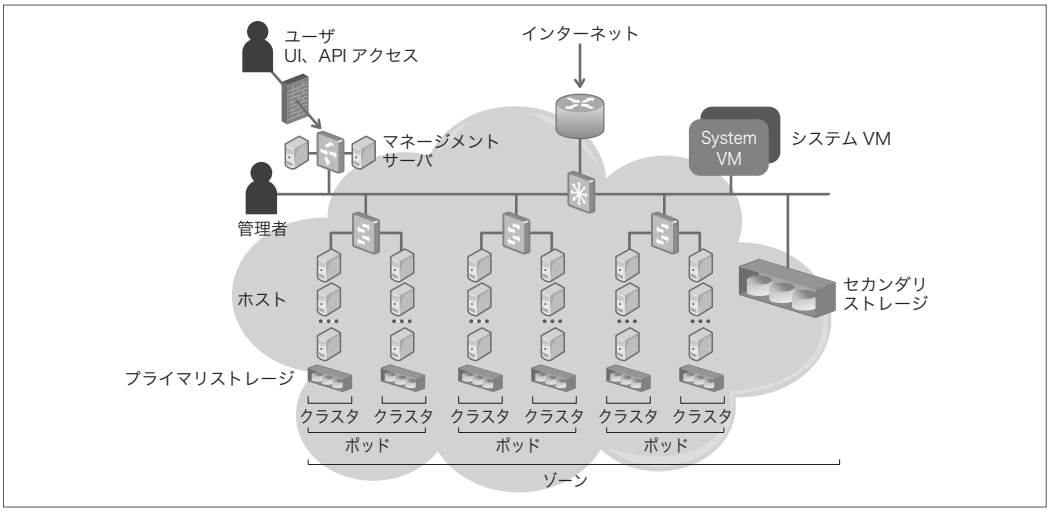
仮想マシンのディスク領域を提供します。ホストのローカルディスクか、NFS、iSCSI、FiberChannelが利用できます。最近では分散ブロックデバイスのCeph RBDもサポートされました。

セカンダリストレージ

OSテンプレート、仮想マシンのディスクの

注1) http://docs.cloudstack.org/CloudStack_Documentation/Administration_Guide%3A_CloudStack

▼図4　CloudStackのネットワーク図（例）



COLUMN
マネージメントサーバの拡張性

複数台のマネージメントサーバをクラスタ化することができます。可用性の向上はもちろん、各ノード間でのジョブシェアリング（負荷分散）にも対応していて、大規模な管理が行えるようなアーキテクチャになっています。

COLUMN
ゾーンの選択

仮想マシン作成時にゾーンを選ぶことができます。たとえば、バックアップ用途の仮想マシンなどを別のゾーンへ作成しておけば、万が一の障害時に影響を少なくすることができます。

スナップショットを保存する領域です。プライマリストレージとは別ハードウェアにすることにより耐障害性が向上します。

クラスタ

複数のホストとプライマリストレージで構成され、おもにノード障害時に仮想マシンがフェイルオーバーするグループとなります。

ポッド

複数のクラスタ（1つでも可）を収容する単位。ポッドごとにスイッチやサーバ、ストレージなどを物理的に分けることで全体の可用性を向上させることができます。

ゾーン

ポッドよりも大きなグループで、地理的に離れたデータセンターに配置すれば、システムの災害対策として活用することもできます。

肝心要のシステムVM

もう1つ忘れてはいけない重要なコンポーネントがあります。システムVMと呼ばれ、2種類の仮想マシンが存在します。

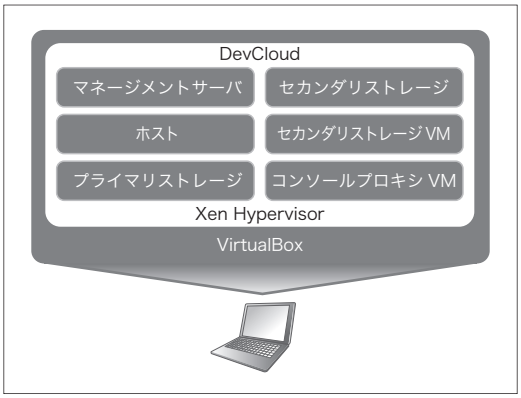
1つ目がセカンダリストレージVMです。セカンダリストレージ領域に保存されているテンプレートやスナップショットを管理します。テンプレートのアップロードやスナップショットの取得などはこの仮想マシンが処理しています。

2つ目がコンソールプロキシVMです。仮想マシンのコンソール画面を提供します。このVMがハイパーバイザとVNC（Virtual Network Computing、リモートデスクトップソフト）を

COLUMN
システムVMはオートスケール対応

CloudStackは、常にシステムVMの状態を監視していて、処理数が多くなると自動的にシステムVMを追加して負荷を分散します。

▼図5　DevCloud概念図



▼図6　DevCloudインポート画面

使って通信し、ユーザのWebブラウザへコンソール画面を描写しています。

システムVMは重要なコンポーネントなので、ぜひ覚えておきましょう。

◆ ◆ ◆

ここまでを読んでCloudStackの魅力を感じていただけたでしょうか？　次からは、実際にCloudStackを触って理解度を深めていきたいと思います。

DevCloudを使ってCloudStackを15分で構築してみよう!

DevCloudとは?

DevCloudは、無償で公開されているCloudStackの各コンポーネントがプレインストールされたOVA(Open Virtual Appliance)形式の仮想アプライアンスです(**図5**)。

DevCloudのインストール

DevCloudインストール要件

DevCloudをインストールするには、以下のインストール要件を満たしている必要があります。

・VirtualBoxのインストール(Windows、Mac(64bit)、Linux、Solaris)
・1GBの空きメモリ
・20GBの空きディスク容量
・仮想化支援機能のCPU(推奨)

始めにVirtualBoxのダウンロードサイト[注2]からVirtualBoxをダウンロードしてPCへインストールしましょう。数分でインストールできます。

DevCloudのインポート

VirtualBoxのインストールができたら、次のURLからDevCloudのOVAファイルをダウンロードします(約1.6GB)。

●DevCloud.ova
http://download.cloud.com/templates/devcloud/DevCloud.ova

ダウンロードしたOVAファイルをVirtualBoxの[ファイル]→[仮想アプライアンスのインポート]から選択して、DevCloudの作成を開始します。「すべてのネットワークカードのMACアドレスを再初期化」がチェックされていないことを確認してインポートをクリックします(**図6**)。

その後はデフォルトのまま進んでください。完了したら、いつでも初期状態からやり直せるようにスナップショットを取得しておくと便利です。スナップショットの取得は、[スナップ

注2) https://www.virtualbox.org/wiki/Downloads

ショット]→[スナップショット作成]から実行できます(**図7**)。

続いてCloudStackを実際に動かしてみましょう。

CloudStackを動かしてみよう

DevCloudをVirtualBoxの「起動ボタン」で起動します。起動が完了(「devcloud login:」が表示)したらVirtualBoxのほうはそのままにしておき、ローカルPCのWebブラウザで「http://localhost:8080/client/」へアクセスします。

図8のログイン画面が出てきたら、ユーザ名に"admin"、パスワードに"password"と入力しましょう。ログインが成功するとCloudStackのダッシュボード(図1)が現れます。

仮想マシンを作成してみよう

初期状態ではゾーンが無効になっているので、仮想マシンを作成することはできません。作成できるようにゾーンを有効にします。[インフラストラクチャ](**図9-①**)→[ゾーン]の[すべて表示](図9-②)をクリックして表示された画面の「操作」欄にある有効化ボタン(図9-③)をクリックします。確認ダイアログが表示されたら[はい]をクリック(図9-④)して完了です。

ゾーンが有効になると、システムVMが起動を開始します。システムVMの状態を確認するには、[インフラストラクチャ]→[システムVM]の[すべて表示]をクリックします。s-1-VMがセカンダリストレージVM、v-2-VMがコンソールプロキシVMです。2つのシステムVMの状態が"Running"になったら、仮想マシンを作成してみましょう。[インスタンス]→[インスタンスの追加]をクリックして仮想マシン作成画面へ移ります(**図10**)。

① セットアップ画面で「テンプレート」を選択
② テンプレートの選択画面で「tiny Linux」を選択
③ コンピューティングオファリング画面で「tinyOffering」を選択(**図11**)
④ データディスクオファリング画面で「設定しない」を選択
⑤ ネットワーク画面でそのまま[次へ]
⑥ 確認画面では任意で名前やグループ名を入力
⑦ [VMの起動]ボタンで完了

▼図7　DevCloudのスナップショット取得

▼図8　CloudStackログイン画面

▼図9　ゾーンの有効化

▼図10　インスタンス追加

COLUMN
Localhostの8080番ポートでDevCloudにアクセスできるのはなぜ？

VirtualBoxでDevCloudへポートフォワーディングされるように設定されています。VirtualBoxの[設定]→[ネットワーク]→[アダプター1]→[高度]→[ポートフォワーディング]で確認することができます。

▼図11　コンピューティングオファリング選択画面

▼図12　コンソール画面のHTMLソース

▼図13　マネージメントサーバへのログイン(Mac OS X)

```
My Mac:~ $ ssh -p 2222 root@localhost
root@localhost's password:
..省略..
root@devcloud:~#
```

▼図14　コンソールプロキシの設定変更

```
root@devcloud:~# mysql -t cloud -e "insert into configuration (name, value) VALUES('consoleproxy.static.publicip', 'localhost')"
root@devcloud:~# mysql -t cloud -e "insert into configuration (name, value) VALUES('consoleproxy.static.port', '8443')"
```

▼図15　マネージメントサーバのサービス再起動

```
root@devcloud:~# service cloud-management restart
* Stopping CloudStack-specific Tomcat servlet engine cloud-management   [ OK ]
* Starting CloudStack-specific Tomcat servlet engine cloud-management   [ OK ]
root@devcloud:~#
```

Creating状態の仮想マシンが表示されていると思います。なかなかRunningにならない場合には、左の[インスタンス]をクリックして更新してみると良いでしょう。

仮想マシンにアクセスしてみよう

[インスタンス]→[仮想マシンの表示名]→[コンソールの表示]ボタン >_ をクリックすると、本来であれば仮想マシンへコンソールアクセスができるのですが、初期状態だとコンソールプロキシの設定がローカル環境(DevCloudの環境)になっていないため接続ができません(**図12**)。コンソールの表示ができるように、マネージメントサーバにログインしてコンソールプロキシの設定を変更します。

マネージメントサーバへログインするには、自分のPCからlocalhostの2222番ポートへsshします。ユーザ名は"root"で、パスワードは"password"です(Macをお使いであればターミナルから**図13**のように、$ ssh -p 2222 root@localhostで接続します[注3])。

マネージメントサーバへログインができたら、**図14**のmysqlコマンドを入力してコンソールプロキシの設定を変更します。変更した設定を有効にするためにマネージメントサーバのサービス再起動を行います(**図15**)。

サービス再起動後は再度CloudStackへログ

COLUMN　DevCloudで複数の仮想マシンを作る

DevCloud上で稼働しているホストはリソースが多くありません。そのため、一番小さいtinyOffering(1CPU、100MBメモリ)でも1台しか起動できません。さらに小さいオファリングを作成すれば2台以上の仮想マシンを起動することができますので、ぜひトライしてみてください。コンピューティングオファリングは、サービスオファリング画面で作成することができます。

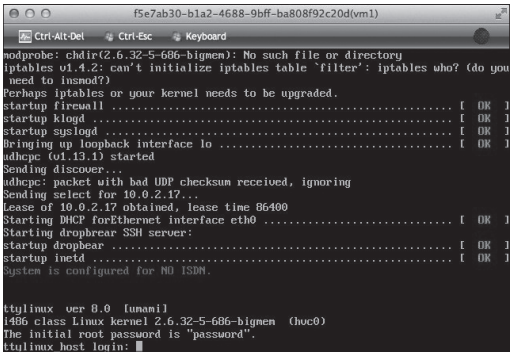
注3) VirtualBoxのコンソールから直接ログインすることもできます。

インが必要です。ログインした後に再度コンソールを表示してみましょう([インスタンス]→[仮想マシンの表示名]→コンソールの表示 >_)。すると、今度はちゃんと仮想マシンのコンソールが表示できるようになったと思います(**図16**)。

このコンソール画面から、ユーザ名"root"、パスワード"password"でログインすることもできます。ログイン後に仮想マシンから外部への接続を試す場合には、/etc/resolv.confから10.0.2.3だけを残してすべて削除します(**図17**)。

編集が終わったら、**図18**のように外部へ通信してみましょう。

▼図16　仮想マシンのコンソール画面



▼図17　/etc/resolv.confの内容

```
% cat /etc/resolv.conf
nameserver 10.0.2.3
%
```

▼図18　外部(Google)への通信テスト

```
% ping www.google.co.jp
PING www.google.co.jp (74.125.235.88): 56 data bytes
64 bytes from 74.125.235.88: seq=0 ttl=63 time=21.525 ms
```

APIを試そう

通常、APIを利用するにはAPIキーとシークレットキーが必要です([アカウント]→[admin]→[表示-ユーザー]→[admin]で確認することができます。もし発行されていなければ、上にある[キー生成]で発行されます)。ですが、今回は認証が不要な統合APIを利用してアクセスしてみたいと思います。

デフォルトでは統合APIは無効になっています。これを有効にするにはマネージメントサーバへsshでログインしたあと、**図19**のコマンドを実行します。今回も変更を有効にするためにマネージメントサーバのサービス再起動を行います(図15)。

サービス再起動ができたら、マネージメントサーバ上からAPIを実行して、起動している仮想マシンの停止をしてみます。停止する仮想マシンのIDが必要なので、インスタンスの詳細画面から確認します。確認できたら**図20**のようにAPIを実行してみましょう。UIでイン

▼図19　統合APIを有効化

```
root@devcloud:~# mysql -t cloud -e "update configuration set value='8096' where name='integration.api.port'"
```

▼図20　仮想マシンの停止コマンドを実行

```
root@devcloud:~# curl 'http://localhost:8096/api?command=stopVirtualMachine&id=1fcbfefb-a85e-413e-b79e-491f1e5e20b2'
<?xml version="1.0" encoding="ISO-8859-1"?><stopvirtualmachineresponse cloud-stack-version="3.0.3.2012-07-04T06:31:57Z"><jobid>0132210d-6a5a-4043-8fbc-83dea1674855</jobid></stopvirtualmachineresponse>root@devcloud:~#
```

COLUMN

統合APIとは

他のシステムと連携するためのAPIエンドポイントです。認証がないため、必要ない場合は無効にしておくか、セキュリティに十分注意して利用してください。

COLUMN

APIのレスポンス形式をXMLからJSONにしたい

APIの実行オプションにresponse=jsonを追加すると、レスポンスをJSON形式にすることができます。デフォルトはXML形式です。
例)http://localhost:8096/api?command=listZones&response=json

COLUMN

CloudStackのグローバル設定

ダッシュボード左下に[グローバル設定]があります。グローバル設定にはCloudStackの動作を決定する設定がいくつもあります。そのうちの一部の設定を紹介したいと思います。

alert.email.addresses	仮想マシンの作成失敗やリソース超過などのアラートメールを送る宛先を指定します
cpu.overprovisioning.factor	CPUの割当倍率を設定します。値を2にするとホストに搭載された倍のCPUを割り当てることができます。その他、メモリとストレージの倍率設定があります
storage.max.volume.size	1つのディスクの最大サイズを指定します
expunge.interval、expunge.delay	削除したVMをどの程度残しておくかを設定します。デフォルトは1日(86400秒)残しておく設定になっています。この期間内であれば仮想マシンを復旧することができます
vm.allocation.algorithm	VMやディスクの配置するリソースを決定するアルゴリズムを設定します。アルゴリズムには、random、firstfit、userdispersingなどがあります。userdispersingは筆者が要求して実装してもらった方式で、アカウントごとのVMやディスクを可能な限り別リソースに配置するアルゴリズムです

その他、たくさんのグローバル設定がありますので一度眺めてみてください。CloudStackがクラウドインフラに必要な動作をいかに考慮しているかがわかってくると思います。

スタンス画面を見ながら実行すると楽しいかもしれません。

停止ができたら、今度はstopVirtualMachineをstartVirtualMachineに変えて、仮想マシンの起動にチャレンジしてみてください。

正しくAPIの実行はできたでしょうか。その他のAPIコマンドについても、次のURLを参照してぜひトライしてみてください。

●Apache CloudStack(incubating)API Docs
http://incubator.apache.org/cloudstack/docs/api/index.html

さらにCloudStackを学ぶために

ここまでCloudStackを学んでみていかがでしたでしょうか? さらにCloudStackを学びたいという方のために参考サイトを挙げておきます。Citrixのサイトには、管理者ガイドの日本語版がありますで、ぜひ読んで理解を深めてほしいと思います。SD

●参考サイト

・Apache CloudStack	URL	http://incubator.apache.org/cloudstack/docs/
・Citrix CloudPlatform	URL	http://support.citrix.com/product/cs/v3.0/doc/
・日本CloudStackユーザー会	URL	http://cloudstack.jp/

Part 3

Open PaaSの本命

Cloud Foundryの簡単デプロイがお勧め!

近年、ソフトウェアの開発／実行環境をサービスとして提供するPaaSの人気が高まりつつあります。オープンソースのPaaSであるCloud Foundryならば、手元にPaaSそのものを構築してしまうことができます。Cloud Foundryの構築を通じて、PaaSの世界に飛び込んでみましょう。

日本Cloud Foundryグループ　草間 一人　KUSAMA Kazuto　Twitter:ID@jacopen

Open PaaSのプライム選択肢「Cloud Foundry」

本節ではPaaSとIaaSの違い、PaaSの動向、そしてCloud Foundryについて解説します。

PaaSとIaaS、SaaSの違い

PaaSとはPlatform as a Serviceの略で、一言で説明すると、「ソフトウェアの開発・実行環境をサービスとして提供するもの」です。ソフトウェアの実行環境を提供するためには、ソフトウェアを動作させるためのコンピューティングリソース、OS、ミドルウェアが必要です。また、動作しているソフトウェアへのアクセスをエンドユーザに提供するためのネットワークも必要ですね。PaaSはこれら一式をまとめて、サービスとして提供します。

Amazon EC2に代表されるIaaS(Infrastructure as a Service)と決定的に異なる点は、PaaSはあくまでもソフトウェアの実行環境を提供するものだというところです。ハードウェアおよびOSを提供するIaaSでソフトウェアを動作させる場合、言語やフレームワークごとのミドルウェアは自らインストール、および設定をしなくてはいけません。

PaaSの場合そういった作業は不要で、書いたソフトウェアをデプロイするだけで動作します。OSへのアクセス権限自体が用意されていないサービスも多く、ソフトウェア開発に集中できるよう、シンプルに作られているのが特徴です。

また、ソフトウェアそのものをサービスとして提供するSaaS(Software as a Service)とも異なります。SaaSはあらかじめ用意されたソフトウェアをサービスとして利用できるものですが、PaaSの場合、そこで動作させるソフトウェアは自ら用意しなくてはなりません。IaaS、PaaS、SaaSで自動化される部分を図にしたものが**図1**です。

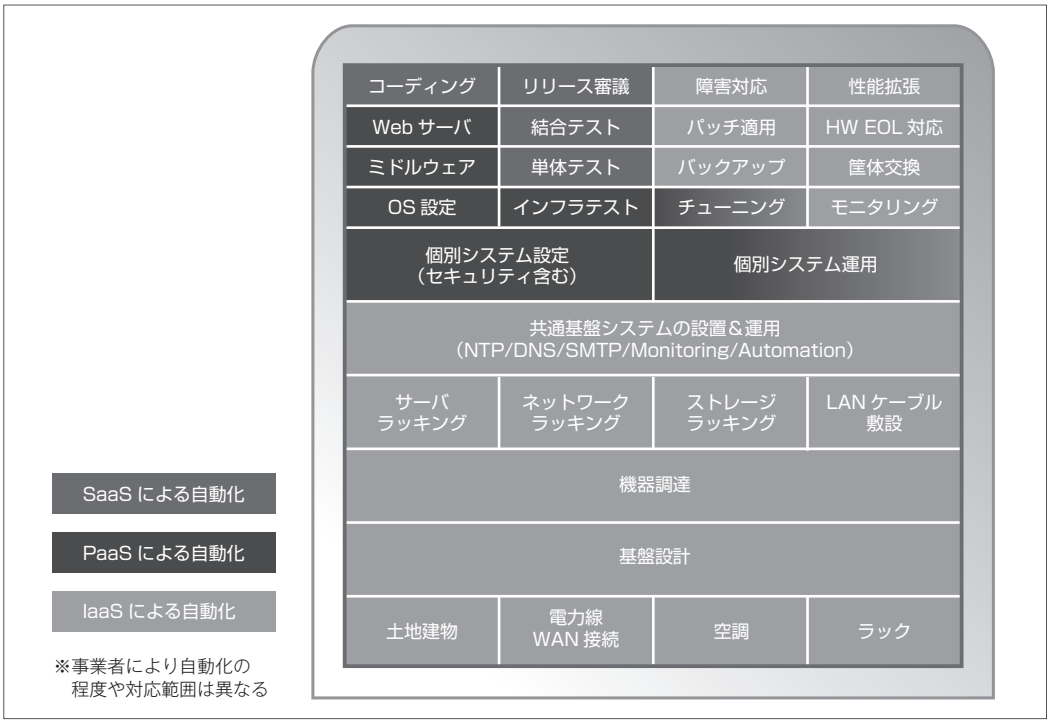
PaaSの動向

もともとPaaSは、2007年にセールスフォース・ドットコムによって提唱されました。そのセールスフォース・ドットコムは、「Force.com」というPaaSを提供しています。Force.com以外では、同じくセールスフォース・ドットコムによる「Heroku」、Microsoftの「Windows Azure」、Google の「Google App Engine」、Engine Yardの「Engine Yard」などのサービスが、世界的にも有名です。

また2012年には、国内でもさまざまなサービスが登場しました。IIJの「Mogok」、paperboy&co.の「Squale」、TIS の「eXcale」、ニフティの「C4SA」などがあります。PaaSの普及という観点では、選択肢が豊富にあることはとても好ましいことです。

いずれのPaaSにも共通する点として、はじ

▼図1　IaaS/PaaS/SaaSによる構築運用の自動化カバー範囲



めは利用できる言語やフレームワークを絞っておき、徐々に対応を広げていくというアプローチが挙げられます。たとえばHerokuの場合では、もともとはRubyのPaaSとして始まりましたが、現在ではJavaやPython、Nodeなどを利用できるようになりました。このような多言語対応PaaSは一般的になりつつあり、国内サービスにおいても多言語対応が増えてくるものと思われます。

Open PaaSとしての実績

これまで紹介したPaaSは、それぞれのベンダが独自のしくみでPaaSを実現し、提供しているものです。このしくみ自体は公開されていないため、「プロプライエタリ(Proprietary) PaaS」と呼べるでしょう。

これに対するアプローチとして、「Open PaaS」と呼べるしくみが登場しています。名前からも想像できるように、PaaSレイヤをOSSとして公開し、開発が進められているものです。そのうちの1つが、「Cloud Foundry」です。Cloud Foundryは、SpringSourceの「SpringSource Cloud Foundry」がベースになっており、同社を買収したVMwareが2011年4月にOSSとして発表しました。Apache2ライセンスで、GitHub上で公開されています。

Cloud Foundryは、マルチ言語対応(Ruby、Java、Node、PHP、Python)、マルチサービス対　応(MySQL、PostgreSQL、MongoDB、Redis、RabbitMQ)となっており、開発者が慣れ親しんだ環境をそのまま利用できます。

Cloud Foundryは登場以来さまざまな企業の興味を引いており、以下の企業からCloud Foundryを採用したPaaSの提供が行われています。

・AppFog、HP、DELL、ActiveState、Tire3、Uhuru Software

国内ではNTT Communicationsが「Clod^{n} PaaS」というサービスを、企業ユーザ向けに提

供を開始しています。また、VMware自らCloud Foundryを採用したサービスをcloudfoundry.comとして提供しています。2013年1月現在、cloudfoundry.comはベータ版を提供中で、無料で利用できます。

Cloud Foundryを試すには、上記ベンダのサービスを利用する方法もありますが、OSSの強みとして、自らの環境にCloud Foundryを構築してしまうこともできます。次章では、Cloud Foundryを実際に利用してアプリケーションのデプロイを解説します。

デプロイが簡単! Cloud Foundry体験!

本節では、実際にCloud Foundryにアプリケーションをデプロイします。Cloud Foundryを試すには、次の3つの方法があります。

1. Cloud Foundryを採用したPaaSを利用する（cloudfoundry.com、appfogなど）
2. Micro Cloud Foundryをダウンロードし、実行する
3. Cloud Foundryのソースをダウンロードし、インストールする

とにかく早くCloud Foundryを試したい、というのであれば、1.のcloudfoundry.comを利用する方法が一番良いでしょう。この場合、Cloud Foundryのクライアントさえ用意すれば良く、あとはアプリケーションをデプロイするだけです。

2.のMicro Cloud Foundryは、Cloud FoundryがあらかじめインストールされたVMのイメージで、VMware Playerなどでこのイメージを実行するだけで、Cloud Foundry環境を手元で動かすことができます。無料配布されており、https://micro.cloudfoundry.com/よりダウンロードできます。

3.は、自らの環境にCloud Foundryをインストールする方法で、1.や2.に比べると少々手間はかかります。ですが、せっかくのOSSのPaaSですから、今回はこの方法を解説します。

作業の大まかな流れは次のとおりです。

① Cloud Foundryの構築
② クライアントPCに、Cloud Foundryのクライアントツールのインストール
③ クライアントPCから、Cloud Foundryにアプリケーションのデプロイ

cloudfoundry.comやMicro Cloud Foundryを利用する場合は、②以降を参考にデプロイを試してみてください。

Cloud Foundryの構築

Cloud Foundryを構築する環境の確認

Cloud Foundryを構築するには、次のスペックの環境が必要です。

・OS：Ubuntu Server 10.04（64bit）
・メモリ：最低1GB、推奨2GB
・ディスク：最低1GB、推奨100GB
・その他条件：インターネットに接続されていること

Cloud Foundryを試す用途であれば、上記の条件さえ満たしていれば、古いPCやVMで十分です。メモリに余裕のあるPCを使われている方の場合、VirtualBoxやVMware Playerを利用するのが手軽で良いでしょう。

また、OSがUbuntu Server 10.04と、少々古いOSとなっていますが、これは今回利用するインストーラがUbuntu Server 10.04を前提としているためです。筆者が試した限りでは、Ubuntu Server 12.04や12.10の場合、インストール途中にエラーが発生し、止まってしまいます。

VMの作成やOSのインストールについては、本稿では深く触れません。インストール作業はSSHを経由して行います。OSの準備ができたら、次のコマンドでopenssh-serverをインストールしておきましょう。また、インストールの際にcurlも必要となりますので、併せてインストールしておきます。

▼図2　Cloud Foundryのインストール確認

```
Deployment Info
***************
* Status: Success
* Config files: /home/cf/cloudfoundry/.deployments/devbox/config
* Deployment name: devbox
* Note:
  * If you want to run ruby/vmc please source the profile /home/cf/.cloudfoundry_deployment_profile
  * If you want to run cloudfoundry components by hand please source the profile /home/cf/.cloudfoundry_
deployment_local
* Command to run cloudfoundry: /home/cf/cloudfoundry/vcap/dev_setup/bin/vcap_dev start
```

```
sudo apt-get updatesudo apt-get install openssh-
serversudo apt-get install curl
```

Cloud Foundryのインストール

それでは、Cloud Foundryをインストールしていきましょう。まず、構築用環境にSSHでログインします。この際のユーザは、sudo権限を持っている必要があります。続いて、次のコマンドを実行します。

```
bash < <(curl -s -k -B https://raw.github.com/
cloudfoundry/vcap/master/dev_setup/bin/vcap_dev_
setup)
```

おおよそ30分から1時間ほど時間がかかります。**図2**のような表示がでれば完了です。まれにですが、インストール途中、ライブラリのダウンロード等でタイムアウトが起きてしまい、インストールが失敗してしまうことがあります。その際は、再度このコマンドを実行すると良いでしょう。

あまりに簡単で拍子抜けするようですが、Cloud Foundryのインストールはこれで完了です。

Cloud Foundryの起動

それでは、インストールしたCloud Foundryを起動してみましょう。次のコマンドを実行することで、Cloud Foundryが起動します(**図3**)。

```
~/cloudfoundry/vcap/dev_setup/bin/vcap_dev start
```

クライアントツールのインストール

Cloud Foundryへ接続し、アプリケーションをデプロイするにはいくつかの方法があります。そのうちの1つが、vmc(VMware Cloud CLI)というコマンドラインツールです。今回は、このvmcを利用してCloud Foundryを試してみましょう。

vmcを利用するには、Rubyが動作する環境が必要です。本記事では、Ubuntu10.04およびRuby 1.8.7-p0環境を前提として解説を進めていきますが、Ruby Installer for Rubyをインストールしたwindowsや Mac OS XのRubyでも利用できます。Rubyのバージョンは1.8.7、1.9.2、1.9.3で確認しています。Cloud Foundryの公式ページ[注1]に詳細なドキュメントがありますので、ご覧ください。

vmcをインストールするには、次のgemコマンドを実行します。これでCloud Foundryを利用する準備が整いました。

```
$ ruby -v ←バージョン確認
ruby 1.9.3p0 (2011-10-30 revision 33570) [x86_64-linux]
$ sudo gem install vmc -v 0.3.23
↑vmcコマンドのインストール
…… (中略) ……
$ vmc -v ←インストールの確認
vmc 0.3.23
```

vmcを利用してCloud Foundryへ接続

それでは、さっそくvmcコマンドを利用してCloud Foundryへの接続を試していきたいところですが、その前に、クライアントPCから

注1)　http://docs.cloudfoundry.com/tools/vmc/installing-vmc.html)

▼図3　起動時に表示されるログ

```
Targeting deployment "devbox" with cloudfoundry home "/home/cf/cloudfoundry"
Setting up cloud controller environment
Setting up the uaa environment
Using cloudfoundry config from /home/cf/cloudfoundry/.deployments/devbox/config
Executing /home/cf/cloudfoundry/.deployments/devbox/deploy/rubies/ruby-1.9.2-p180/bin/ruby /home/cf/
cloudfoundry/vcap/dev_setup/bin/vcap start cloud_controller uaa postgresql_node dea redis_node mysql_node
mongodb_gateway stager mysql_gateway vblob_gateway rabbitmq_gateway router postgresql_gateway health_manager
vblob_node redis_gateway rabbitmq_node mongodb_node -c /home/cf/cloudfoundry/.deployments/devbox/config -v /
home/cf/cloudfoundry/vcap -l /home/cf/cloudfoundry/.deployments/devbox/log
cloud_controller              :  RUNNING
uaa                           :  RUNNING
postgresql_node               :  RUNNING
dea                           :  RUNNING
redis_node                    :  RUNNING
mysql_node                    :  RUNNING
mongodb_gateway               :  RUNNING
stager                        :  RUNNING
mysql_gateway                 :  RUNNING
vblob_gateway                 :  RUNNING
rabbitmq_gateway              :  RUNNING
router                        :  RUNNING
postgresql_gateway            :  RUNNING
health_manager                :  RUNNING
vblob_node                    :  RUNNING
redis_gateway                 :  RUNNING
rabbitmq_node                 :  RUNNING
mongodb_node                  :  RUNNING
```

Cloud Foundryへの経路を確保しておく必要があります。

vmcからCloud Foundryへの接続、およびデプロイしたアプリケーションへの接続には、HTTP(HTTPS)を利用するため、クライアントPCから構築したCloud Foundry環境の80番ポートへ接続できる環境が必要です。また、デフォルトではvcap.meというドメインを利用しますので、vcap.meの名前解決を行った際に、Cloud Foundry環境のIPアドレスが返ってくる必要があります。

仮にMacやLinuxをお使いであれば、もっとも簡易な方法として、ssh tunnelを張ってしまう方法があります。

```
sudo ssh -L 80:<Cloud Foundryを構築したマシンのIPアドレス>:80 <Ubuntuのユーザ名>@<Cloud Foundryを構築したマシンのIPアドレス> -N
```

またこの他に、Hostsファイルに記述してしまう方法もあります。この場合、アプリケーションごとにHostsファイルの記述を増やしていく必要があります。

```
<Cloud Foundryを構築したマシンのIPアドレス>  api.vcap.me ←API用
<Cloud Foundryを構築したマシンのIPアドレス>  hello-sd.vcap.me ←アプリ用
```

準備ができたら、さっそく**図4**のコマンドを実行してみましょう。vmc targetコマンドで接続するCloud Foundryの設定、vmc loginコマンドでログイン、vmc infoコマンドで接続先の情報取得を行います。

Cloud Foundryにアプリケーションをデプロイ

準備も整ったので、さっそくCloud Foundryにアプリケーションをデプロイしてみましょう。今回は、Rubyの軽量フレームワーク「Sinatra」を利用して、アクセスするとHello worldを表示する簡単なアプリケーションを作ります。

利用するコードは**リスト1**です。このコードを、app.rbという名前にして保存してください。保存場所は、他のファイルが存在しないディレ

▼リスト1　サンプルコード(app.rb)

```
require "sinatra"

get "/" do
  "Hello World"
end
```

クトリが良いでしょう。

次に、ファイルを保存したディレクトリへ移動し、**図5**のようにvmc pushコマンドを実行します。

「Starting Application　:OK」まで表示されていれば、アプリケーションのデプロイは成功です。さっそくブラウザでURLへアクセスし、Hello Worldが表示されていることを確認してみましょう。

このように、ちょっとしたアプリケーションであれば簡単に、Cloud Foundry上にデプロイすることができます。

アプリケーションのアップデート

次に、アプリケーションのアップデート方法を解説します。先ほどのリスト1を**リスト2**のように変更してみましょう。

この修正を反映するには、再度vmc updateを行います。

```
vmc update hello-sd ←同じアプリケーション名を指定
Uploading Application:
  Checking for available resources: OK
  Packing application: OK
  Uploading (0K): OK
Push Status: OK
Stopping Application 'hello-sd': OK
Staging Application 'hello-sd': OK
Starting Application 'hello-sd': OK
```

これだけで、アプリケーションが更新されます。ブラウザで、表示されている文字列が変わっていることを確認してみましょう。

▼図4　Cloud Foundryの接続

```
$ vmc target http://api.vcap.me ←vmcの接続先を、http://api.vcap.me/へ切り替えます
Successfully targeted to [http://api.vcap.me]

$ vmc register ←最初に接続するユーザを作成
Email: ←任意のメールアドレスを入力
Password: ←任意のパスワードを入力
Verify Password: ←任意のパスワードを入力
Creating New User: OK
Attempting login to [http://api.vcap.me]
Successfully logged into [http://api.vcap.me]

$ vmc login
Attempting login to [http://api.vcap.me]
Email: ←登録したメールアドレスを入力
Password: ←登録したパスワードを入力
Successfully logged into [http://api.vcap.me]

$ vmc info ←target先の情報を取得

VMware's Cloud Application Platform
For support visit http://support.cloudfoundry.com

Target:   http://api.vcap.me (v0.999)
Client:   v0.3.23

User:     kusama@example.com
Usage:    Memory   (0B of 2.0G total)
          Services (0 of 16 total)
          Apps     (0 of 20 total)

$ vmc runtimes ←利用可能言語を表示

+-------------+--------------+----------+
| Name        | Description  | Version  |
+-------------+--------------+----------+
| ruby18      | Ruby 1.8.7   | 1.8.7p357 |
| ruby19      | Ruby 1.9.2   | 1.9.2p180 |
| ruby193     | Ruby 1.9.3   | 1.9.3p194 |
| java        | Java 6       | 1.6      |
| java7       | Java 7       | 1.7      |
| python2     | Python 2.6.5 | 2.6.5    |
| node        | Node.js      | 0.4.12   |
| node06      | Node.js      | 0.6.8    |
| node08      | Node.js      | 0.8.2    |
| erlangR14B01 | Erlang R14B01 | R14B01  |
| php         | PHP 5        | 5.3      |
+-------------+--------------+----------+

$ vmc services ←利用可能サービスを表示

============= System Services =============

+------------+---------+-------------------------------+
| Service    | Version | Description                   |
+------------+---------+-------------------------------+
| blob       | 0.51    | Blob service                  |
| mongodb    | 1.8     | MongoDB NoSQL store           |
| mysql      | 5.1     | MySQL database service        |
| postgresql | 9.0     | PostgreSQL database service   |
| rabbitmq   | 2.4     | RabbitMQ message queue        |
| redis      | 2.6     | Redis key-value store service |
+------------+---------+-------------------------------+

========== Provisioned Services ===========
```

▼図5　vmc pushコマンドを実行してアプリケーションをデプロイ

```
~$ cd hello ←保存先ディレクトリに移動
~/hello$ ls ←app.rbが存在することを確認
app.rb
~/hello$ vmc push
Would you like to deploy from the current directory? [Yn]:
Application Name: hello-sd ←アプリケーション名を入力（任意）
Detected a Sinatra Application, is this correct? [Yn]: ←Sinatraが自動検出されているので、そのままEnter
Application Deployed URL [hello-sd.vcap.me]:←アプリケーションのURLを指定。ここはデフォルトのままEnter
Memory reservation (128M, 256M, 512M, 1G, 2G) [128M]:  ←デフォルトは128MBなので、そのままEnter
How many instances? [1]: ←インスタンス数を入力。デフォルトは1なので、そのままEnter
Create services to bind to 'hello-sd'? [yN]: ←サービスの作成をするかどうか。ここではデフォルトのままEnter
Would you like to save this configuration? [yN]: ←これまでの設定を保存するかどうか。ここではデフォルトのままEnter
Creating Application: OK
Uploading Application:
  Checking for available resources: OK
  Packing application: OK
  Uploading (0K): OK
Push Status: OK
Staging Application 'hello-sd': OK
Starting Application 'hello-sd': OK
```

▼リスト2　app.rbを変更

```
require "sinatra"

get "/" do
  "Hello World from Software Design!"
end
```

Cloud Foundryのこれまでとこれから

開発主体の変更

2012年12月、VMware、および親会社のEMCは、Pivotal Initiativeという新組織を設立し、クラウドプラットフォームやビッグデータに関するプロダクトを移管すると発表しました。これまでCloud Foundryは、VMwareが中心となって開発を進めてきましたが、今後はPivotal Initiativeへと移ることとなります。なお、Pivotal Initiativeの最高戦略責任者として、VMwareのCEOを勤めたこともある、ポール・マリッツ氏が就くことが決定しています。

VMwareは仮想化インフラストラクチャにおけるトップベンダですし、EMCもストレージにおけるトップベンダですが、いずれもCloud Foundryのようなミドルウェアを扱うには、すこしイメージが異なります。そこで、このグループが持っていたvFabricやSpringSource、Cloud Foundryといったミドルウェアを Pivotal Initiativeに集約し、ミドルウェアにおけるトップベンダを目指すことになったようです。

この体制変更により、Cloud Foundryの開発が大きく変わるということは無いようですが、今後Cloud Foundryに関わる人達には、Pivotal Initiativeという名前が馴染み深いものとなっていくでしょう。

PaaSとしての機能向上

Cloud Foundryは日々アップデートが続けられており、この半年だけでもさまざまな機能が追加されました。筆者もすべてのアップデートを追えているわけではありませんが、特徴的なものを紹介します。

Rutime/Framework対応強化

Cloud Foundryで使えるRuntimeやFrameworkが強化されました。ここ最近では次のようなものが追加されています。

・Runtimes
　Node 0.8
　Java7
・Frameworks
　Play 2.0

また、Standaloneという形でのデプロイもサポートされました。これにより、Webサーバーを必要としないアプリケーションを動作させることができるようになりました。他のPaaSでいう、「Worker」に相当するものが実現可能になります。

ServiceのVersions/Plans対応

これまでServicesでは、MySQLならMySQL 5.1のみ、PostgreSQLならPostgreSQL 9.0のみと、1種類のServiceにつき1つのバージョンしか指定できませんでした。

このVersions対応により、バージョンを指定してServiceを作成できるようになりました。ベンダが対応していれば、MySQL5.0、5.1、5.5で好きなバージョンを選ぶ、ということが可能になります。また、Plansというしくみも導入されました。これまでは、1つのServiceにつき1種類のPlanしか提供できなかったため、すべてのユーザに同じデータベース容量のPlanを割り振ることができませんでした。Plansにより、データベースの容量、最大接続数、キャッシュサイズなど、Planごとに異なる設定を行い、ユーザに提供できるようになりました。

V2(ng)シリーズ登場

Cloud Foundryはコンポーネントごとにリポジトリが分けられ、開発が進められています。それに平行し、アーキテクチャの変更を伴う大きなアップデートが、次期バージョンとしてさらに別リポジトリとして開発が行われています。次期バージョンはv2、もしくはng(next generation)という名前が付けられています。

・Router v2
・DEA ng
・Cloud Controller ng
・VMC ng
・Health Manager 2.0

この中でもとくに注目すべきは、DEA ngとCloud Controller ngです。DEA(Droplet Exection Agent)とは、Cloud Foundryにデプロイされたユーザのアプリケーションを動作させるコンポーネントですが、このDEAが「Warden」というしくみに対応しました。

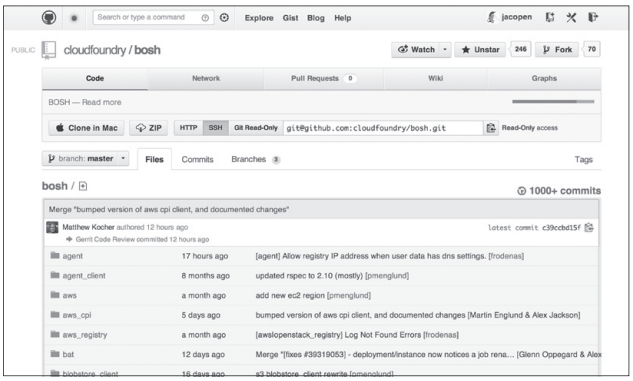
Wardenは軽量なLinuxコンテナで、OSレベルでの仮想化を行います。LXCに近いしくみと言えばわかりやすいでしょうか。これまでは、マルチテナントモード[注2]での動作の場合、1つのDEA上で複数のユーザアプリケーションが同居していました。DEAngでは、アプリケーションをそれぞれWardenコンテナで仮想化することで、公平なリソース分配や、セキュリティの向上を実現しています。

また、Cloud Controllerはその名のとおり、Cloud Foundryにおいてユーザアプリケーションサービスの配置や管理、APIの提供などの中心機能を担うコンポーネントですが、これがCloud Controller ngとして大きく変わります。

まず、もともとRailsで書かれていたものがSinatraに書き直されました。また新機能として、App Space/Organizationといった新しい概念を導入しました[注3]。これにより、チーム開発機能が利用可能になります。なお、Router v2については、すでに現行バージョンのCloud Foundryに取り込まれています。前節でセットアップしたCloud Foundryも、Router v2がセットアップされています。また、VMC-ngと呼ば

注2) Cloud Foundryは、1DEA 1アプリケーションのシングルテナントモードと、1DEA複数アプリケーションのマルチテナントモードが選択できます。
注3) http://blog.cloudfoundry.org/2012/06/20/heads-up-on-some-new-cloud-controller-features/

▼図6　Bosh（https://github.com/cloudfoundry/bosh）



れていたものも、vmcの0.4系としてリリースされています。Cloud Foundryは、すべてのコンポーネントのバージョンを一気に上げるのではなく、安定したコンポーネントから逐次置き換えていく方針のようです。残りのv2/ngについても、そう遠くないうちに取り込まれるのではないかと推測しています。

BOSHの登場

今後Cloud FoundryでPaaSを構築する場合、非常に重要なツールがあります。それが、BOSHです（**図6**）。

BOSHは2012年4月に発表された、大規模サービスのリリースエンジニアリング、デプロイ、ライフサイクルマネジメントを行うためのツールチェーンです。Cloud Foundryチームによって開発されており、ライセンスはCloud Foundryと同じくApache 2ライセンスです。GitHubからソースコードのダウンロードができます。

Cloud FoundryはIaaSレイヤに依存しないアーキテクチャが特徴です。これは多くの場合メリットをもたらすのですが、裏を返せば、Cloud FoundryにはIaaSをコントロールするための機能がいっさい用意されていません。

大規模サービスを運用する場合、VMの作成からCloud Foundryのデプロイ、デプロイしたコンポーネントのライフサイクルマネジメントなどが必要不可欠ですが、これまでは、独自にしくみを作り込んでいく必要がありました。しかし、BOSHの登場により、IaaSのコントロールまでまとめて任せられるようになりました。もともとBOSHはVMwareがcloudfoundry.comを運用するために開発したツールのため、実績も十分です。現在BOSHは、AWS、vSphere、vCloud、OpenStack環境で利用できます。

最後に

Cloud FoundryがOSSとして公開されてから1年半経ちました。2012年にはCloud Foundryを採用して有料サービスを提供するベンダが登場したほか、DELLやHPといった大企業でも採用されるケースが出てきています。2013年には国内でもCloud Foundry採用サービスが登場する見込みで、Cloud Foundry普及のための足回りが徐々に固まってきたように思います。

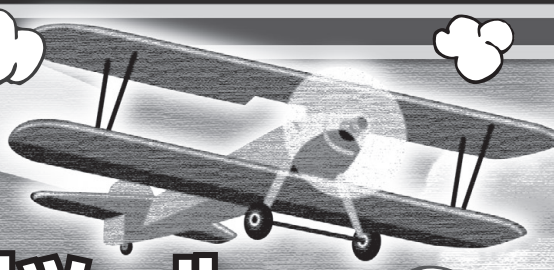
本記事ではCloud Foundryの一部しか紹介できませんでしたが、Cloud Foundryは洗練された自律分散アーキテクチャで構築されており、クラウドに限らずアーキテクチャ面に興味のある方にも、とても参考になる作りになっています。ぜひみなさんの環境でもCloud Foundryを構築していただき、洗練されたクラウドアーキテクチャが手元で動く楽しさを感じていただきたいと思います。本記事がCloud Fondryの世界へと踏み出す第一歩となれば幸いです。SD

第1特集
もっとクラウドを活用してみませんか？

Part 4

複数クラウドの構築が15分でできる!

マルチクラウド管理ツール「Scalr」入門



オープンソース系クラウドの構築方法を説明してきましたが、いろいろなクラウドを試していると今度は管理するのが大変なことに！　本章では、そんな複数のクラウド環境を併用するのに超便利な管理ツール「Scalr」について解説します。

本文　：Scalrユーザ会　梶川 治彦　KAJIKAWA Haruhiko（株）IDCフロンティア
コラム：Scalrユーザ会　成田 真樹　NARITA Masaki（株）IDCフロンティア

マルチクラウド管理ツールでもっと楽しよう!

Amazon EC2以外のクラウドサービスもいろいろと出てきましたし、パブリッククラウドに加えプライベートクラウドも花盛りです。これらクラウドをまとめて管理しちゃいましょう！

それが本稿で紹介するマルチクラウド管理ツールです。

インフラ構築が15分で完了!

さて、クラウド上に1サイト立ち上げるのに、どれぐらいの時間が必要でしょう？　よくある構成としてロードバランサ、アプリケーションサーバ、データベースサーバを利用した3階層構造のサイトがあると思いますが、これらをクラウド上で1から構築したとすると1日では終わらないかもしれません。また、冗長化したり、監視、バックアップ、DNSなど、これらも含めるとさらに日数がかかるかと思います。

15分と言うと大げさに思うかもしれませんが、クラウド管理ツールでは上記のような機能を包括しているので、本当に15分程度で冗長化なども含めた3階層構造のサイトを簡単に構築できます。

インフラ構築はクラウド管理ツールに任せて、最も重要な「サービス企画」や「コンテンツ開発」へ力を注ぐことが可能です。

クラウド管理ツールって何?

マルチクラウド管理ツールの説明に入る前に、そもそものクラウド管理ツールが登場した背景から見てみましょう。クラウドといえば、やはりAmazon EC2（以下、EC2）を連想される方も多いと思います。クラウド管理ツールも当初はおもにEC2を管理する目的で作られました。

今でこそEC2も便利なGUIのAWS Management Consoleを持っていますが、登場当初はAPIを操作するためのコマンドラインツールぐらいしかありませんでした（FirefoxのExtensionなどもありましたね。あれは便利でした）。また、当初はEBSブートなどなかったので、インスタンスを停止すると、せっかく設定したサーバ設定やデータもすべて消えてしまうのがあたりまえ。EC2は時間課金なので、コストを抑えるためにも必要なときだけ立ち上げて、不要なときは停止するといった運用を行いたいわけですが、そうするとデータが消えてしまう等々いろいろと面倒な起動／停止を伴う運用が重要でした。

こういった不便を補うための手段はいろいろありましたが、その中の1つとしてクラウド管理ツールが登場しました。GUIでの操作に加え、インスタンス停止で全部消えてしまう問題への対策として、「サーバ設定を別で管理し、インスタンス起動後に適用する」といった形をとる

ことができるようになりました。

クラウド管理ツールから マルチクラウド管理ツールへ

単なるEC2を管理する管理ツールから、いつマルチクラウド管理ツールとなっていったのでしょうか？

いろいろなIaaSの登場

EC2の成功と共に、Amazon Web Services（以下、AWS）以外のクラウドが登場してきます。ホスティング業者やデータセンター事業者、通信キャリアなどによるパブリッククラウドもそうですが、プライベートクラウド等で多く利用されるオープンソースのEucalyptusやCloudStack、OpenStackなどです。

クラウドを選んで使う時代

それぞれのクラウドはそれぞれの特色があります。たとえば、コスト重視、パフォーマンス重視、安定性重視、サポート重視などなど特徴はさまざまです。また、場合によってはプライベートクラウドもあるでしょう。ユーザは利用目的に合わせて、クラウドを選んで使う時代になってきています。

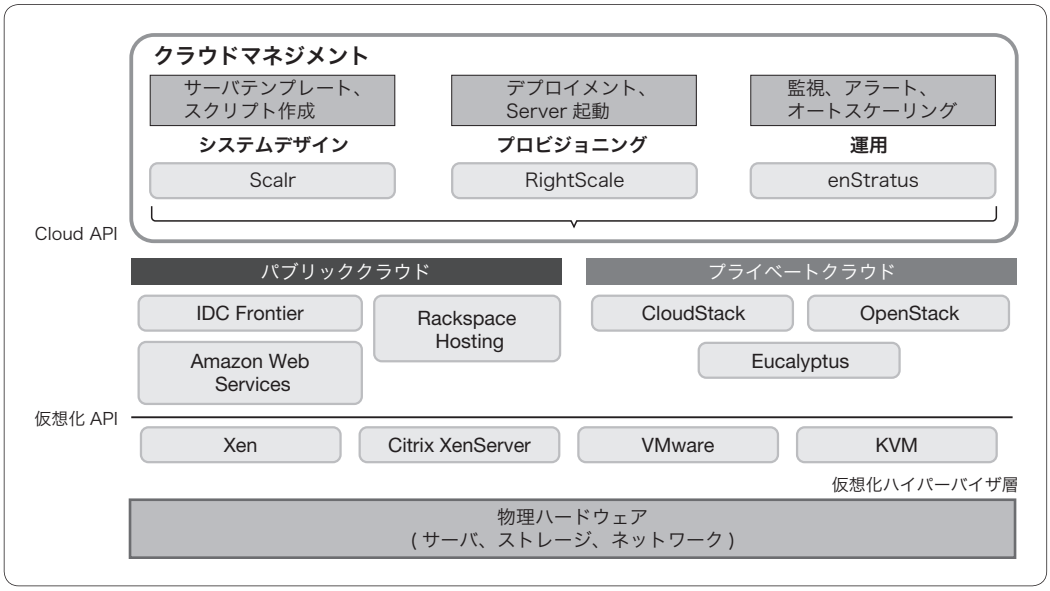
また、最近の傾向として、クラウド環境におけるベンダーロックインを避ける動きもあり、あえて一社に絞らず複数のクラウドを選択する、といった動きも出てきています。

マルチクラウド環境を統合管理

そうなってくると、いろいろ使い分けているクラウドを一緒に管理したくなります。EC2で構築した環境を別のクラウドで動かしたくなったりもします。サーバの設定は同じだけど、別々のクラウドへ同一の環境を作る必要も出てくるでしょう。

それに応える形で、管理ツールもEC2の管理ツールからマルチクラウドを管理できるツールへと変貌を遂げていきます（**図1**）。これは偶然なのかもしれませんが、管理ツールでは前述のとおり、「EC2はサーバを止めるとデータが消えるため、サーバ設定は別管理している」ことが幸いして、他のクラウドをサポートするにあたっても、インフラ部分さえサポートできてしまえば、後はEC2用に管理していたサーバ設定をそのまま適用できた、といったところも

▼図1　マルチクラウド環境を支える主要技術



マルチクラウドのサポートが早くできた理由ではないでしょうか。

マルチクラウド管理ツールを使う理由

実際にとても便利と思われる特徴をいくつか挙げてみます。

●複数のクラウドを同一操作で管理！

何度か出てきていますが、やはり1ヵ所から複数のクラウドを管理できるのは楽です。とくに管理者と運用者が別々の場合、運用者へクラウド別の説明がいらなくなるので、管理者も運用者もどちらも楽ができます

●同一設定のサイトを簡単に構築！

同じような構成／設定のサイトをいくつも立てる場合は、構成／設定をクローン(もしくは同じものを適用して)してサラリと同じものが作れます。管理ツールによっては、クラウドを超えてクローンを作成できます

●モニタリングや監視も勝手に開始！

たいていの管理ツールは、立ち上げたインスタンスのモニタリングや監視をしてくれます。リソースグラフも勝手に作ってくれます(**図2**)

●オートスケールで負荷対応！

たいていの管理ツールは、オートスケールに対応しています。モニタリングしているぐらいなので、サーバの負荷は見えています。だったらこのデータを元にオートスケールできるのは、当然のことでしょう

▼図2　モニタリング画面サンプル

構成管理も管理ツール上で！

ドキュメントを書くのは苦手ですか？　筆者は大嫌いです **:-)**

クラウド管理ツールでは、サーバの構成／設定情報を別途ツール上で管理しているため、わざわざドキュメントにしなくても使い方がわかります。

細かい設定を行う場合にはスクリプト実行で対応することになりますが、これもスクリプトを見れば、誰でもどのようなことがされているのかが理解できます。しかも、たいていのツールはスクリプトのバージョン管理までできます。

加えて、運用のためのスクリプトを作成しておくと、ボタン1つで実行できるので、ここもドキュメント要らずにできます。

マルチクラウド管理ツール「Scalr」

マルチクラウド管理ツールはいくつか存在しますが、今回は「Scalr(スケーラー)」をご紹介します。

Scalrとは

Scalrはオープンソースのマルチクラウド管理ツールです。オープンソースソフトウェア(OSS)版の展開に加え、Scalr社が提供するHosted版、Enterprise版が存在します。Hosted版はScalr社がScalrの機能を提供してくれるホスティング型のサービスです。Enterprise版は、オンプレミス環境へScalrをインストールし利用するタイプのサービスです。どちらもScalr社によるサポートを得ることができます。

Scalrの特徴は、OSS版の存在、わかりやすいGUI、簡単に設定できるオー

トスケール、インテグレートされたDNSなどが挙げられます。

Scalr上での管理はインフラ的な要素(クラウド部分)とDNS設定やConfigなどOS以上の設定部分が分離されており、どのクラウドを利用する場合でも同一の設定を適用することで、同じ環境を構築できるようになっています。

サポートしているクラウド

マルチクラウドを管理できますが、実際には**表1**のクラウドをサポートしています。サポートクラウドはどんどん広がっていて、最近ではIDC Frontierがサポートされました。Google、KT、OpenStackに関しては、本誌が出るころにはサポート対象になっているかもしれません。また、プライベートクラウドとしてCloud Stack、OpenStackがサポートされているので、これらを採用するクラウドベンダー(パブリッククラウド)の取り込みが進むかもしれません。

僕らをラクにする機能

Scalrはクラウドベンダーが用意する管理ポータルなどに比べて、より便利(僕らを楽)にしてくれる機能をたくさん持っています。

●数ステップでサイトを作成

①Farm作成、②Role追加／設定、③起動、の3ステップでサイトの立ち上げが可能です(Farm、Roleは後ほど説明します)

●冗長化やオートスケールが簡単

MySQLなどは、2台以上の起動数を指定するだけでMaster/Slave構成の冗長化が可能です。オートスケールも何を元に(Loadaverageなど)スケールするかを指定するだけです

●DNSも同時に管理

DNSも同時に管理できます。オートスケールで増減する分は自動的に登録／削除してくれます

▼表1　Scalrサポートクラウド一覧

Cloud Provider	Type
Amazon Web Services	Public
Rackspace(First-genのみ)	Public
Rackspace Next-gen	Public(サポート予定)
IDC Frontier(セルフタイプのみ)	Public
CloudStack	Private
OpenStack	Private(Private Beta)
Eucalyptus	Private
Nimbula	Private
KT ucloud	Public(Private Beta)
Google Compute Engine	Public(サポート予定)

●各デーモン設定のプリセット管理

各デーモン(ApacheやMySQLなど)の設定情報をプリセットとして管理可能することが可能です。これらを割り当てるだけで設定できます

●各ポイントでスクリプト実行

サーバ起動時、シャットダウン時などさまざまなポイントでスクリプトの実行ができます。Scalrに足りない機能を補うことにも使えますし、運用を目的としたスクリプトも良いでしょう

もちろん「マルチクラウド管理ツールを使う理由」でも挙げた部分はScalrでもサポートされていて、便利(僕らを楽)にしてくれます。

Scalrを使ってみよう!

Scalrが簡単／便利そうだということが伝わりましたでしょうか？　まだよくわかりませんか？　百聞は一見に如かずということで、Scalrを実際に使ってみて簡単便利を体験してみましょう！

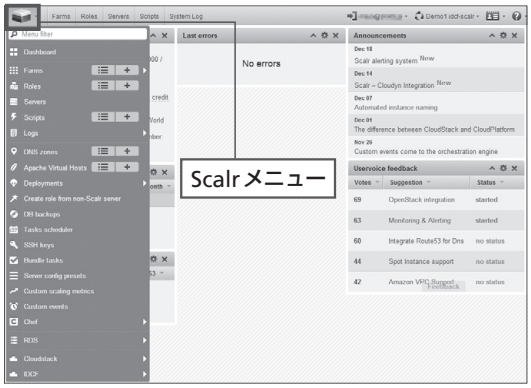
Hosted Scalrへ登録(30-Days Free)

Scalrには3種類のバージョン(Hosted、Enterprise、OSS)が存在することを説明しましたが、本稿ではHosted Scalrを利用してみ

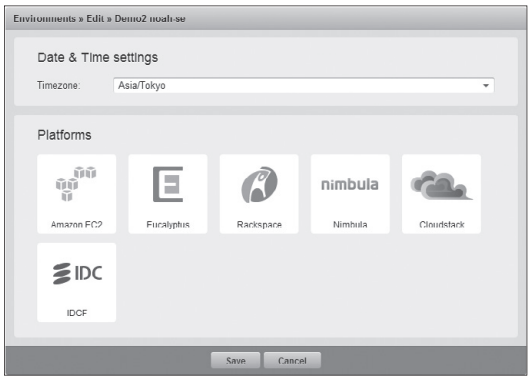
たいと思います。Hosted ScalrはScalr社が提供するScalrのホスティングサービスのため、利用するためには費用が発生しますが、30日間フリーで利用できるお試し版がありますので、こちらで試していきたいと思います(**図3**)。

それではさっそく登録しましょう。ブラウザでScalrの日本語公式サイト[注1]へ行き、右上の「無料お試し体験」へ進みましょう。必要事項を記入しアカウント作成を行ってください。しばらくするとScalrからパスワードが記載されたメールが届きますので、ログイン画面[注2]から登録したメールアドレスとメールで届いたパスワードでScalrへログインしてください。

▼図3　Scalrログイン後のメイン画面(Scalrメニューを開いた状態)



▼図4　クラウド登録画面



Scalrへクラウド登録

Scalrへログイン後、最初に行うことはクラウドの登録です。パブリッククラウドとしてはAmazon EC2、Rackspace、IDC Frontierを登録できますが、ここではEC2とIDC Frontier(以下IDCF) Cloudの説明を記載します。

ログイン直後はクラウド登録画面(**図4**)になっており、かつEC2を利用するかどうかを設定する画面になっていると思います(もしなっていない場合は、画面右上、右から3番目をクリックし、「Manage」を選択してください。画面右の[Actions]→[Configure]とすることでクラウド登録画面を呼び出すことができます)。各クラウドのアイコンをクリックすることで、クラウドの登録を行います。当然ですが、マルチクラウド管理ツールなので複数のクラウドを登録して利用することができます。

また、画面右上、右から3番目をクリックして表示されるManageの画面で、右上の緑色の[+]をクリックすることでクラウド登録環境を

COLUMN オープンソース版でさらに自由な管理を!

お試しアカウントでは制限が気になりますか?
それならOSS版のScalrを使ってみましょう。

OSS版はHosted版の無制限アカウントと基本的に同じ環境で、台数制限や使用期限にとらわれずに使うことができます。使用するには**https://github.com/Scalr/scalr**にアクセスし、ダウンロード、インストール、各種設定という手順を踏む必要がありますが、IDCフロンティアではOSS版Scalrのテンプレート(IDCF Cloud用)を公開しているので、試しに使ってみたいという方はこちらのテンプレートを利用するとより手軽に始められるでしょう。また、Scalrのrpmも公開しています(**http://repo.cloud.idc.jp/Linux/CentOS/6/idcf/x86_64/**)。こちらを利用すれば、ある程度インストールの手間を省くことが可能です。サポートはありませんが、さまざまな制限から解放されるため「もっと自由に、低コストで!」というのであれば試してみる価値はあります。

注1) http://scalr.jp/
注2) https://my.scalr.net/

COLUMN
クラウドサービス登録時の警告

使用するクラウドサービスを登録すると「roles builder supports only EC2 and Rackspace platforms.」という警告が出ることがあります。これはRole Builder（本文「Roleとは」参照）に対応していないクラウドサービスに対して表示される警告ですが、すでにクラウドの登録自体は完了しているので、既存のRoleのみを使う（Role Builderを使わない）のであれば気にしなくても問題ありません。

複数作成し、切り替えながら利用することもできます。余談ですが、User/Teamの機能で、Teamごとに作成した環境を割り当てて利用するといったこともできます。

Amazon EC2の場合

「I want use Amazon EC2」のチェックを入れた後、EC2側の「Account Number」「Access Key」「Secret Key」「X.509 Certificate file」「X.509 Private Key file」を入力する必要がありますので、EC2側からこれらの情報を取得したうえでEC2の登録を行ってください。EC2側の情報はAWS公式日本語サイト[注3]のアカウント情報、セキュリティ証明書からすべて入手することができます。

IDCF Cloudの場合

「I want use IDCF」のチェックを入れた後、IDCF Cloud側の「API Key」「Secret key」を入力する必要がありますので、IDCF Cloud側からこれらの情報を取得したうえで登録を行ってください。IDCF Cloud側の情報はクラウドへログイン[注4]後、右上の［アカウント名］→［マイプロファイル］で確認できます。なお、API URLについては変更する必要はありません。

Farm作成とRole追加／設定

それではさっそくサイトを立ち上げてみましょう。

Scalrでは①Farm（グルーピング）の作成、②FarmへRole（機能）の追加と設定、という流れでサイトを作り上げていきます。基本的にはこれだけでサイトの立ち上げができますが、付随してDNSの管理やVirtualHostの管理等も行うことができるので、まとめてやってしまうと楽です。

基本的にインフラ要素（実際のインスタンス、EC2とIDCF Cloudなど）とコンフィグ（DNSやVirtualHost）は分離されており、別管理する形になっています。これはクラウド環境がEC2でもIDCF Cloudでも、作成したコンフィグを割り当てるだけで同一の環境を作りあることができることを意味しています。これはマルチクラウド管理に優れた方法です（**図5**）。

Farmとは

Farmとは各Server（正確にはRole）の振る舞いや設定を、論理的にグループ化する機能です。基本的にはどのようなグルーピングにしても問題ないですが、わかりやすくA-SiteのFarm、B-SiteのFarmといったようにサイト単位で分けたり、MySQLのFarmやApacheのFarmなど機能単位で分けるのがわかりやすいでしょう。

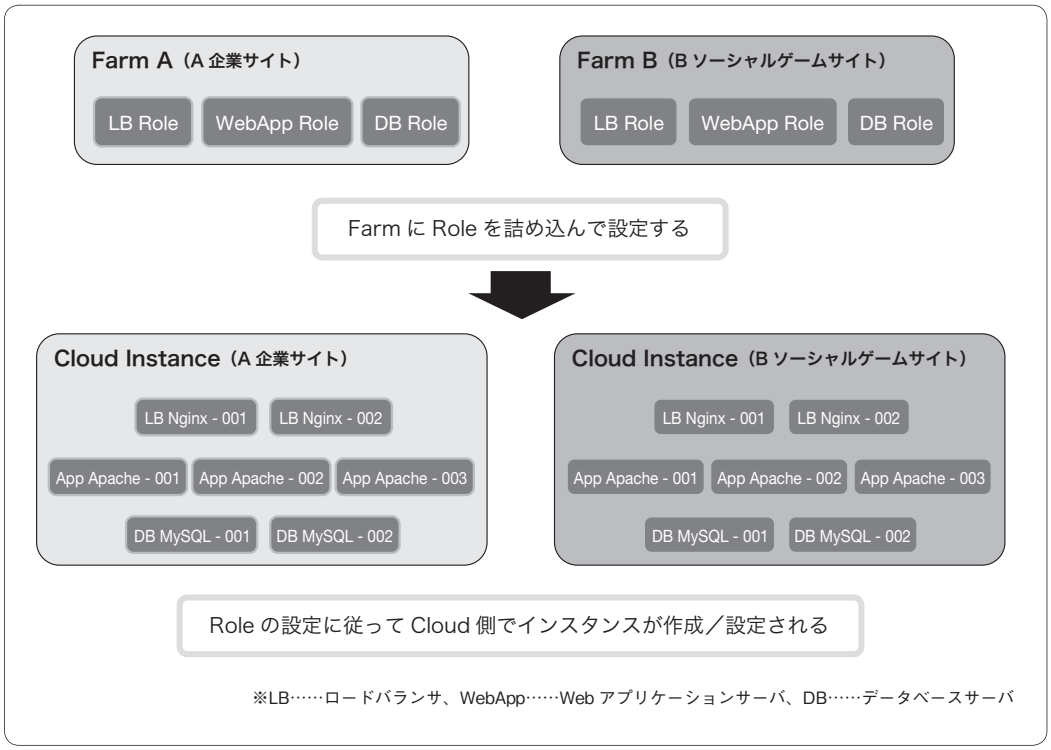
作成したFarmの一覧は、左上の［Scalrメニュー］→［Farms］から確認できます。

Roleとは

RoleとはApacheやMySQLなどの機能（役割）のことです。Roleの種類は複数ありますが、これらRole一つ一つが各クラウド上ではOSのイメージファイル（AMIやTemplate）として存在します。RoleはScalrが用意しているものに加えて、自分で作成することもできます。「Role

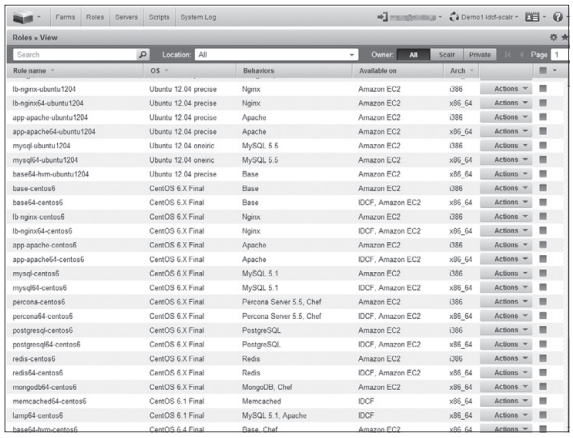
注3） https://aws.amazon.com/jp/
注4） https://noahcloud.jp/portal/portal/login

▼図5　FarmとRoleのイメージ



Builder」といったGUIベースのものに加え、既存サーバを取り込む「Create role from non-Scalr server」や「Create role from managed server」といくつかの手法があります。とくにScalr管理外のサーバを取り込む機能は、便利かつ特徴的でしょう。

▼図6　Roleの一覧



Roleの一覧は、左上の[Scalrメニュー]→[Roles]で確認できます(**図6**)。

Farm作成とRoleの追加／設定

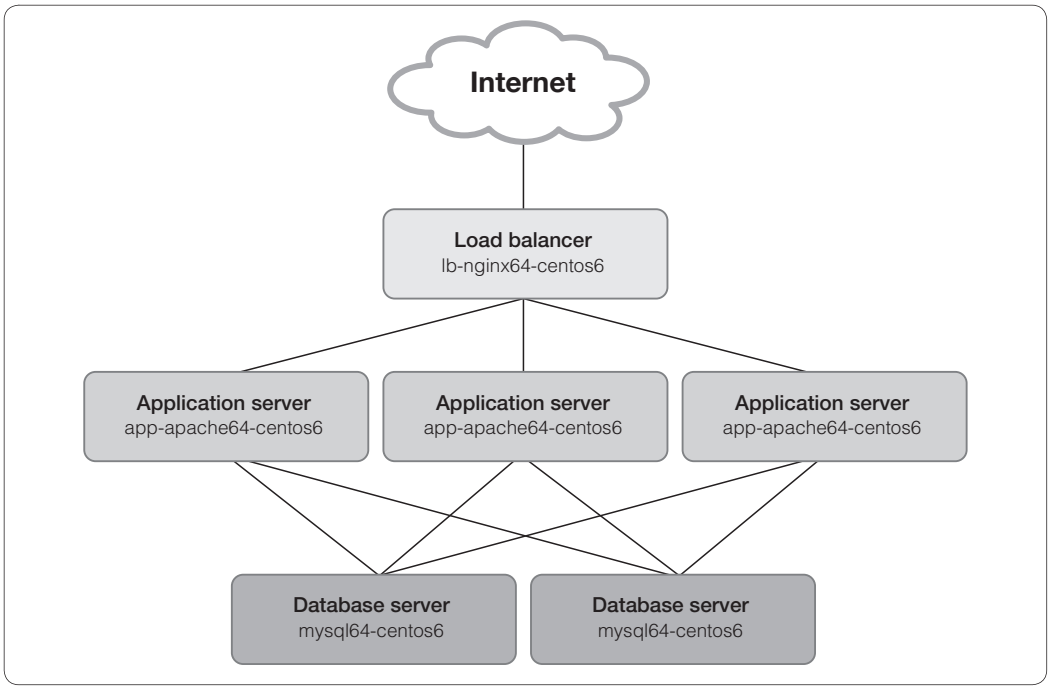
左上Scalrメニューから[Farms]→[+]でFarmを新規追加します。

Farmタブの[General info]の[Name]に適当な名前を設定してください。その後、Rolesタブをクリックします。

[Roles library]をクリックするとカテゴライズされたRoleの一覧が表示されますので、必要なRoleを選択し[+]をクリックで追加していきます。

ここでは試みにLoad balancerとして「lb-nginx64-centos6」、Application serverとして「app-apache64-centos6」、Database serverとして「mysql64-centos6」を選択し、LB-APP-DBの3階層構造のサイトを作成してみましょう(**図7**)。

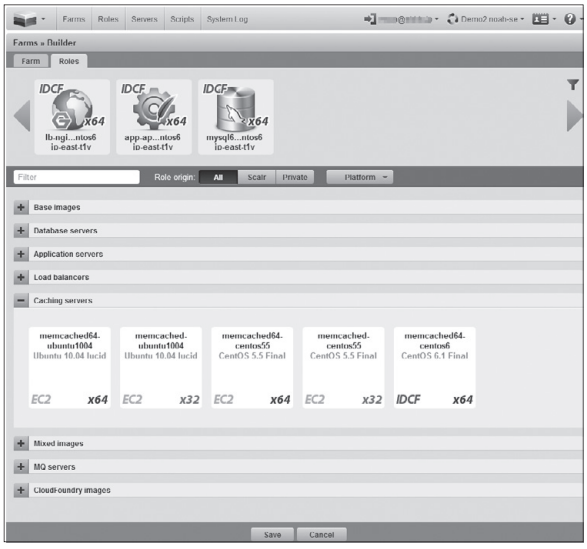
▼図7　構築する3階層構造のサイト



なお、複数のクラウドを登録している場合、Role追加時にクラウドを選択して登録することができます。

次に各Roleの設定を行います。画面上部に並んだ各Roleのアイコンをクリックしてください。画面下部へ選択したRoleの設定画面が出てきます(**図8**)。最低限、次の設定を行ってください。

▼図8　FarmへRoleを追加する設定画面



● Amazon EC2

「Placement and type」から「Availability zone」と「Instance type」を選択します

● IDCF Cloud

「Cloudstack settings」から「Service offering」「Network」「Shared IP」の3つを選択します。「Shared IP」の「Use system defaults」はうまく動かない場合があるようなので、必ずIPアドレスを選択してください。ここで指定するIPアドレスはすべてのRoleで同一のIPアドレスで問題ありません

● EC2、IDCF共通

「MySQL settings」の「EBS Type」と「EBS size」については後から変更することができませんので、最初の段階で適切な設定を行ってください。MySQLにかかわらず、Database

のRoleには最初にしか設定できない同様の設定項目「Database settings」がありますので、ご注意ください。

最後に[Save]ボタンでFarmを保存してください。

Farmの起動!

[Save]ボタンを押すと、Farmの一覧画面になっていると思います(なっていない場合は、Scalrのメニューから[Farm]→[一覧]を選択してください)。

出来上がったFarmの[Actions]→[Launch]でFarmを起動しましょう!!

Farmを起動すると、StatusがTerminatedからRunningへ変わります。

Farm一覧の[Servers]の[View]をクリックすることで、クラウド側で起動するServer(インスタンス)の一覧を確認できます。

Server一覧では、各インスタンスのStatusを確認することができ、FarmをLaunchした直後であれば、Pendingになっているはずです。この後、Initializingになり、Runningに変われば、起動完了です。

DNS Zoneの管理

ScalrではDNSの管理も同時にできます。自分のドメインを管理することもできますが、Scalrが提供してくれる、「[*複雑な英数字(自動生成)*].scalr.ws」といったドメインも利用できます(scalr.wsはおもにテスト用になると思います)。

先ほど起動したFarmへDNS Zoneを割り当ててみましょう。

Scalrメニューから[DNS zones]の右側にある[+]を実行してください(**図9**)。

「Domain name」で自分のドメインか、[*複雑な英数字*].scalr.wsを選択しますが、今回はテストなので「Use domain automatically generated and provided by Scalr」を選択し、「[*複雑な英数字*].scalr.ws」を利用します。

次の「Automatically create A records for」で、作成したZoneのFarmへの割り当てと、Zoneに対してAレコードを割り当てるRole(登録したZoneを名前解決すると、ここで指定したRoleに関連づいたIPアドレスが返ってくるようになります)を指定します。今回はFarmへ「作成したFarm」を割り当て、Roleには「lb-nginx64-centos6」を指定します。

「SOA settings」はそのままで問題ないでしょう。

「DNS Records」は必要なレコードがあれば追加します。今回は「www 0 CNAME %hostname%.」を指定して、「www.[*複雑な英数字*].scalr.ws」でもアクセスできるようにします。

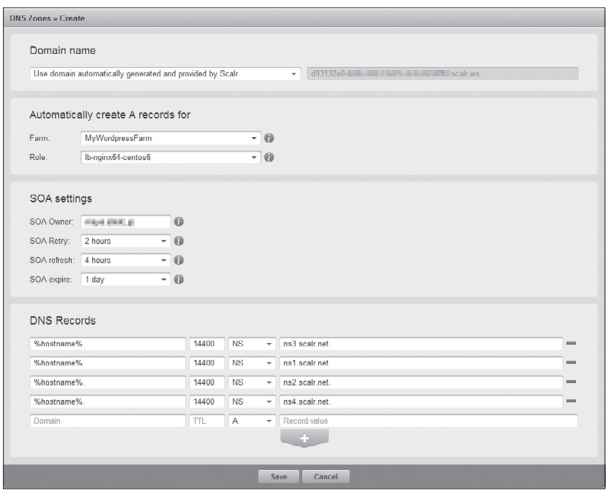
設定が終わったら[Save]を押して保存してください。DNS Zoneが有効になるまで、5分から10分程度かかります。

Apache VirtualHostの管理

VirtualHostの管理も別で行えます。

Scalrメニューの[Apache Virtual Hosts]の右側にある[+]を実行します(**図10**)。

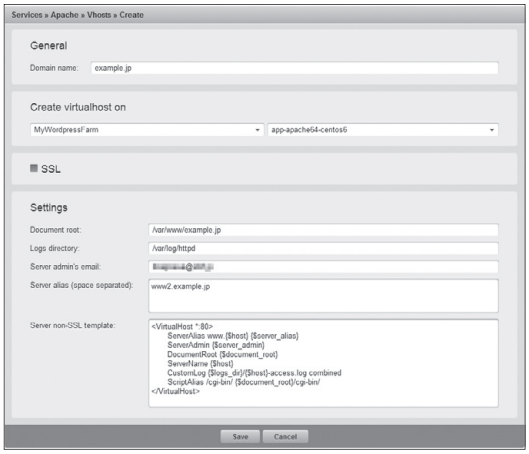
▼図9　DNS Zoneの設定画面



「General」の「Domain name」欄にはVirtual Hostとして設定するドメイン名を指定します。今回は先ほど作成したDNS Zoneの「[*複雑な英数字*].scalr.ws」を指定します。

「Create virtualhost on」でVirtualHostを作成したいFarmとFarm内のApache Roleを指定します。今回は「作成したFarm」「app-apache64-centos6」を指定します。

▼図10　VirtualHostの設定画面



▼図11　ポートフォワードの設定（IDCF Cloud）

▼図12　ファイアウォールの設定（IDCF Cloud）

SSLを利用する場合は「SSL」へチェックを入れ「Certificate」「Private key」「Certificate chain」をそれぞれ指定しますが、今回は指定しません。

「Settings」については、サイトに合わせて設定してください。

アクセスしてみよう！

アクセスする前に、IDCF Cloudで利用している場合はIDCF Cloud側で追加の設定が必要です。ScalrのRole設定で指定したIPアドレスに対して、ポートフォワードとファイアウォールの設定が必要になります。

今回の設定では、Role設定の「Cloudstack settings」で設定したIPアドレスへのポートフォワードとして、パブリックポートのTCP-80（httpサービス用）を「lb-nginx64-centos6」Roleで起動した仮想マシンのプライベートポートのTCP-80へフォワードする設定を追加します（**図11**）。

同様にファイアウォールの設定として、TCP-80をオープンする設定を追加します（**図12**）。

なお、EC2の場合はセキュリティグループの設定もScalrが自動的に行うため、これらの設定は必要ありません。

いよいよアクセスしてみましょう。せっかくDNS Zoneも登録したので、作成したZoneに対してブラウザでアクセスしてみてください。

アクセスURL http://[*複雑な英数字*].scalr.ws

「If you can see this page, it means that your Scalr farm configured succesfully.」と表示されましたか？

どうでしょうか。Farm作成からLaunchまで15分とかかっていないのではないでしょうか。かなり簡単に3階層構造のサイトを構築できることがおわかりいただけたかと思います。

もっとも、実際にはこの後にコンテンツを乗せたりしなければなりませんが、インフラ構築

としてはかなり楽ができそうだと感じていただけたでしょうか。

その他便利機能

基本はFarm作成→Role追加／設定→Launchで完了なのですが、DNS ZoneやVirtualHostの設定のように、便利な機能がいろいろあります。とくに便利な機能のいくつかをご紹介します。

コンテンツのデプロイ機能

ここまでに行ったインフラ構築の後はコンテンツのデプロイだけだと思いますが、Scalrはデプロイの機能も持っています。

デプロイ機能はコンテンツのソース定義と、定義したソースを使ってアプリケーションの定義を行う必要があります。

Scalrメニューから[Deployments]→[Sources]でソースの定義を行います(**図13**)。ソースの「Type」として、svn、http、gitから選択できます。

次にScalrメニューから[Deployments]→[Applications]でアプリケーションの定義を行います(**図14**)。

定義したソースの指定と共に「Pre-deploy」「Post-deploy」欄でアプリケーション展開前、展開後に実行するスクリプトを定義できます。スクリプトはどんな内容でもかまわないのですが、主な利用方法として、Preのほうで環境整備(たとえば、動作に必要なrpmのインストールなど)、Postでアプリの設定(アプリのコンフィグファイルの準備など)などを行うのが一般的でしょう。

実際のデプロイは、Scalrメニューから[Deployments]→[Applications]の画面で、[Actions]→[Deploy]で実行できます。また、Roleの設定であらかじめ指定しておくことも可能です。

スクリプトの実行

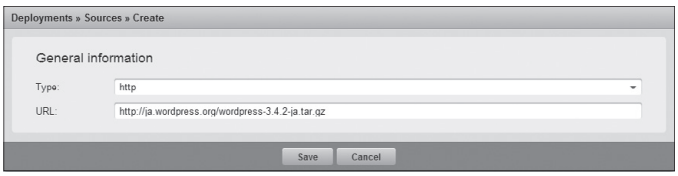
Scalrに付属のDeploy機能は複雑なことを行うのにはあまり向いていません。デプロイ時にもっと複雑なこと(ソースからアプリケーションのインストールなど)を行いたいのであれば、別途用意されている、スクリプト実行の機能が便利です。

スクリプトはScalrメニューの[Scripts]から確認や追加ができます(Scalr社が最初から準備しているスクリプトもあります。**図15**)。

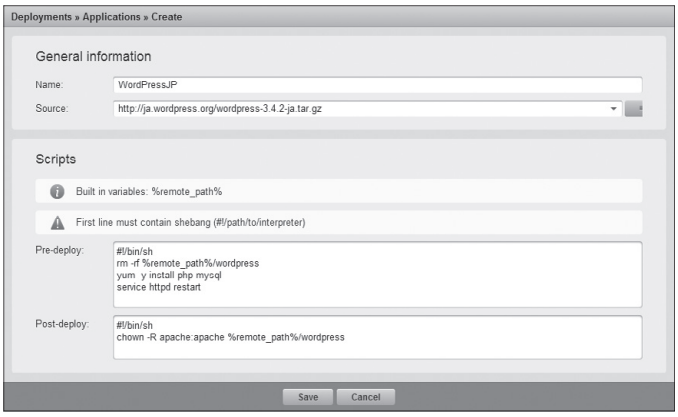
詳細は書きませんが、新規追加に加え、既存のスクリプトからForkしたり、バージョン管理をすることもできます。

実行方法は複数用意されており、スクリプト一覧やサーバ一覧からの即時実行に加え、[Scalrメニュー]→[Tasks scheduler]によるスケジュール実行などもサポートされます。また、Role設定であらかじめ設定することもでき、この場合はさまざまなタイミング(ホスト起動時、ダウン時など)を指定して、実行させるこ

▼図13　デプロイのためのソース定義画面



▼図14　デプロイのためのアプリケーション定義画面



とができます。

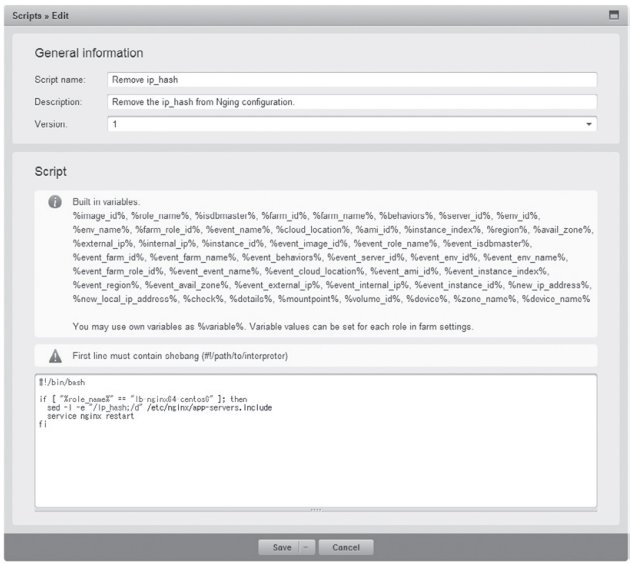
基本的には、Scalrが持ち合わせていない機能については、スクリプト実行でカバーしていく形になります。

また、運用でも利用できる面が多くあります。たとえば、バックアップスクリプトを用意しておき、オペレータにScalrから実行してもらうなどが考えられます。

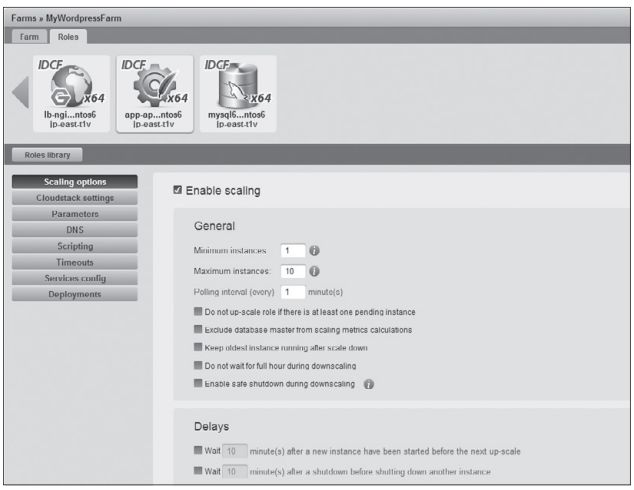
オートスケールへの対応

Roleの設定に「Scaling options」の項目がある

▼図15　スクリプト編集画面



▼図16　オートスケールの設定画面



ことに気が付かれた方も多いと思います。

Scalrでは、「Minimum/Maximum instances」による最少／最大インスタンス数の指定と、「Enable scaling based on」で、何を計算のベースにスケーリングを行うかを指定するだけで、オートスケーリングに対応できます(**図16**)。スケーリング計算のベースとしては「LoadAverages」「FreeRam」「URLResponseTime」「SQLQueue Size」「DateAndTime」「BandWidh」があり、これらから複数の指定が可能です。この中でも「Date AndTime」は日時指定によるオートスケールとなるので、あらかじめわかっているイベントに備えて指定するなど使いでがあると思います。

また、スケーリング計算のベースは自分で定義([Scalrメニュー]→[Custom scaling metrics])することもでき、サーバ上のファイルを読み込んだ結果や、スクリプト実行の結果を元にしたオリジナルのスケーリング計算ベースを定義できます。

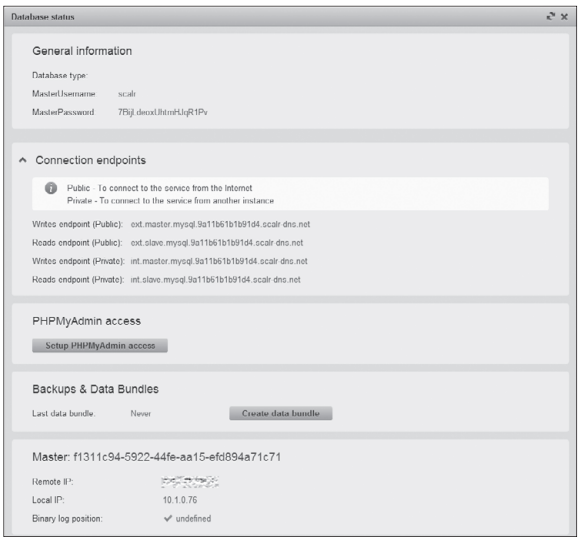
オートスケールして増減したサーバはRoleの種類によってさまざまなことが自動的に処理されます。たとえば、増減したサーバは自動的にDNSへ登録／削除されます。Webサーバが増減した場合、自動的にLoad Balancerへ追加／削除されます。MySQLの場合は1台目がMasterとなり、2台目以降はSlaveとして構成されます。Masterが停止した場合などは、自動的にSlaveをMasterへ昇格してくれます。また、アプリケーションがMySQLへアクセスするためのConnection endpointsと呼ばれる専用のアクセス先DNSが提供されます。これはFarm一覧から[Actions]→[MySQL status]で確認できます(**図17**)。当然ですが、MySQLの増減や

COLUMN Slave-DBが作れない?

少し前の話になりますが、CloudStack ver2.2.14とScalr ver3.5で動かしていたときに、Slave-DBの作成に失敗する現象がよく起きました。原因は「Master-DBのSnapShotから別のクラスタに作成したSlave-DBのボリュームをアタッチできない」というCloudStackの不具合によるものです。そのため、Slaveが同クラスタに配置されることを祈って何度も作り直すという、非常に不安定な状況でした。その後、CloudStack ver2.2.15に上がることで仕様が変更され、上記の問題は落ち着きました。

Scalr上で同じ構成のFarmを作っても、使用するクラウドサービスの仕様によってはうまく動作しないことがあります。使っていて疑問点や予想外のエラーが出た場合は、公式サイトから問い合わせてみるとよいでしょう。

▼図17　Connection endpointsの一覧画面



Master/Slaveの切り替えなどがあると、Connection endpointsの中身も自動的に書き換えてくれますので、アプリケーションは提供されるConnection endpointsへアクセスしていれば、増減やMaster/Slaveの切り替わりを意識することなく、対応することができます(DNSなので、書き換わりのタイムラグは存在します)。

Scalrユーザ会への案内

Googleグループで「Scalrユーザ会[注5]」が立ち上がっていますので、興味のある方はぜひ参加してみてください。

Scalrユーザ会では、「国内における普及促進」「Scalrに関する情報収集と公開、技術促進」「ユーザ間および外部との技術的・交流促進」「Scalr関連技術に関する情報収集・配布」「Scalr講習会、ワークショップ等の開催」など、Scalrを盛り上げていきたいと考えています。なお、ユーザ会ではHosted Scalr、OSS Scalrどちらの話題も取り扱っていく予定です。

すで開催済みの「第1回目の勉強会」の際に利用したスライドをslideshareで公開しています[注6]。こちらのスライドはIDCF Cloudでの例となっていますが、クラウド固有部分以外はEC2でも参考になると思います。内容的には今回の記事とかなりかぶっていますが、実際にWordpressのサイトを立ち上げるところまで書かれていますので、参考にしてみてください。

Scalrユーザ会は「オープンソースカンファレンス2013 Tokyo/Spring[注7]」へ参加いたします。ブースに加え、『マルチクラウド可能なクラウド構成ツール「Scalr」で楽々インフラをデプロイしよう』といった内容でセミナーも行う予定です。また、1つ前の時間枠では「クラウド管理システム"Scalr"をテンプレートでラクラク構築してみよう!」といった内容でIDCフロンティアによるセミナーも行う予定です。こちらはIDCF Cloudのトライアルも配布されますので、Scalrを試してみる絶好の機会となっています。ご興味/質問等ありましたら、ぜひお立ち寄りください。SD

注5) https://groups.google.com/forum/#!forum/scalr_user_group　本誌サポートサイトにもリンクを貼っておきます。
注6) http://www.slideshare.net/hal-k/scalr-hands-on-2012120701
注7) http://www.ospn.jp/osc2013-spring/

第1特集
もっとクラウドを活用してみませんか？

Part 5

Amazon EC2との親和性には一日の長あり!

それでもEucalyptusをお勧めしたい理由

IaaS環境を構築できるオープンソースソフトウェアの草分け的存在「Eucalyptus」。国内ではOpenStackやCloudStackの台頭によりやや押され気味ではありますが、開発は活発に進み、AWSとの親和性も健在です。あらためて現状を知り、選択の幅を広げておきましょう。

羽深 修　HABUKA Osamu

Eucalyptusを使ってみませんか?

皆さんはEucalyptusを知っていますか？　もしくは覚えていますか？　Eucalyptusは OpenStackやCloudStackよりも前に一世を風靡(ふうび)したIaaS環境を構築するオープンソースソフトウェアです。「風靡した」と過去形で書いたのは、日本国内においてはここ1〜2年でEucalyptusの人気が下火になり、OpenStackとCloudStackが二強状態となっているためです。

Part5ではそんなEucalyptusを紹介し、OpenStackやCloudStack以外にもIaaS環境を構築するソフトウェアがあることを知っていただき、もしくは使っていただこうと思います。

Eucalyptusおさらい

Eucalyptusは米国カリフォルニア大学サンタバーバラ校の研究プロジェクトで開発され、現在はEucalyptus Systems社が開発および管理を行っています。バージョン2.0系まではオープンコアライセンスモデル[注1]を採択しており、オープンソース版とエンタープライズ版がありましたが、バージョン3.0としてリリースされたエンタープライズ版を最後に2つのソースコードはオープンソース版に統合されました。オープンコアライセンスモデルを採択していたことにより、よく「真のオープンソースではない[注2]」と揶揄(やゆ)されていましたが、今では開発に関する情報は積極的に公開され、メーリングリストやIRCで自由に議論することができます。当然ながら議論の結果によっては提案した機能や要望やパッチが取り込まれます[注3]。

前述したようにEucalyptusはOpenStackやCloudStack同様にIaaS環境を作るためのソフトウェアですが、Eucalyptusの最大の特徴はAmazon EC2/S3/IAM/STS API互換であることと、EC2のマシンイメージが流用できるという点です。つまり、Amazon EC2環境とのハイブリッドクラウド環境を作る場合、OpenStackやCloudStackに比べるとEucalyptusのほうが親和性が高いと言えます。事実、2012年3月23日（米国時間で22日）にEucalyptus社とAWSが提携し、EucalyptusにおけるAWSに関するAPIの互換性に対して支援していくことを表明[注4]しています。

Eucalyptusの最新バージョンは2012年12月20日（米国時間で19日）にリリースされた3.2.0です。Eucalyptus 3.2.0の最大の特徴は利用者用のWebUI（**図1**）が付属し、CUIでの操作に慣

注1） コアの部分をオープンにし、拡張部分は商用として提供するライセンス形態
注2） コミット要求に対する対応があまり良くなかったのも一因です。
注3） 個人的には現在のEucalyptusのほうがOpenStackよりも要望が通りやすいです。
注4） http://www.eucalyptus.com/news/amazon-web-services-and-eucalyptus-partner

れないユーザでも簡単に直感的に操作できるようになったことです。

Eucalyptusを使ってみようと思ったら、Eucalyptus社が提供しているEucalyptus Community Cloud[注5]を利用するのが一番手っ取り早い方法です。ただし、Eucalyptus Community Cloudは「利用する」ことができるだけで、プライベートクラウドの醍醐味の1つである「構築して楽しむ」ということができません。自分自身のマシンにEucalyptus環境を構築したい場合はEucalyptus FastStart[注6]を利用して、CentOS + KVMという構成でEucalyptus環境を構築できます。もしさらに本格的にEucalyptus環境を構築したい場合には、Eucalyptus社が配布しているバイナリを利用して構築する必要があります。

Eucalyptusを支えるしくみとは

Eucalyptusは次に示すいくつかのコンポーネントで構成されます。

- Cloud Controller(CLC)……EC2 APIを受け、要求に応じて以下のコンポーネントに命令を投げるコンポーネント
- Walrus……S3 APIを受けたり、NCやSCからの要求に応えるコンポーネント
- EUARE……IAM APIやSTS APIを受けたり、CLCからの要求に応えるコンポーネント
- Cluster Controller(CC)……ネットワークに関する機能を提供し、CLCからの要求を受けたり、配下のNCを管理し命令するコンポーネント
- Storage Controller(SC)……EBSに関する機能を提供し、CLCやNCからの要求に応えるコンポーネント
- Node Controller(NC)……CCからの要求を受け、ハイパーバイザに命令するコンポーネント

これらのコンポーネントはFront End、Middle End、Back Endという3つの層で表現されます(**図2**)。

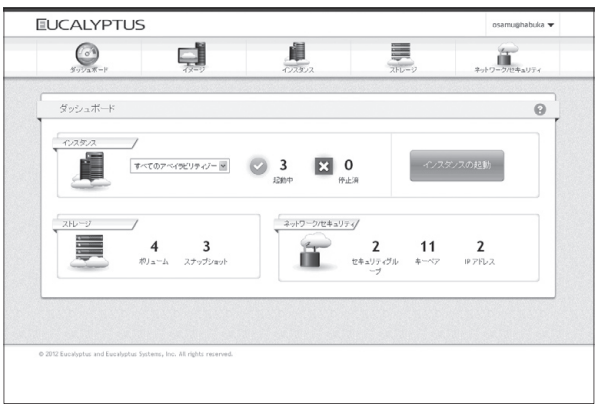
これらのうち、EUAREは明確に1つのコンポーネントとして独立しておらず、現状はCLCの内部に実装されています。ちなみにハイパーバイザとしてXenやKVMではなく、VMware(ESX/ESXi/vCenter)を使用する場合はVMware Brokerというコンポーネントが必要ですが、説明がややこしくなるため割愛します。

Front Endの各コンポーネントは1つのEucalyptus環境にそれぞれ1つしか配置することができません。Middle EndとBack Endをまとめて Zone もしくは Available Zone と呼び、Middle Endの各コンポーネントは1つのZoneにそれぞれ1つしか配置できません。このZoneを複数用意することでMulti Clusterという構成を作ることができます(**図3**)。

Eucalyptusができること、できないこと

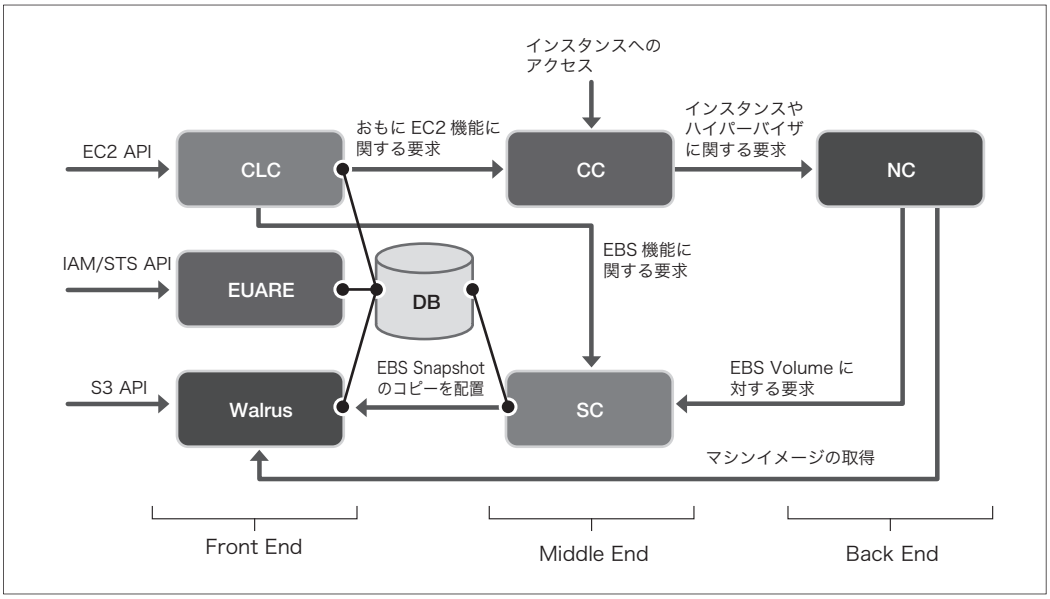
Eucalyptusはプライベートクラウドである

▼図1　ユーザコンソール

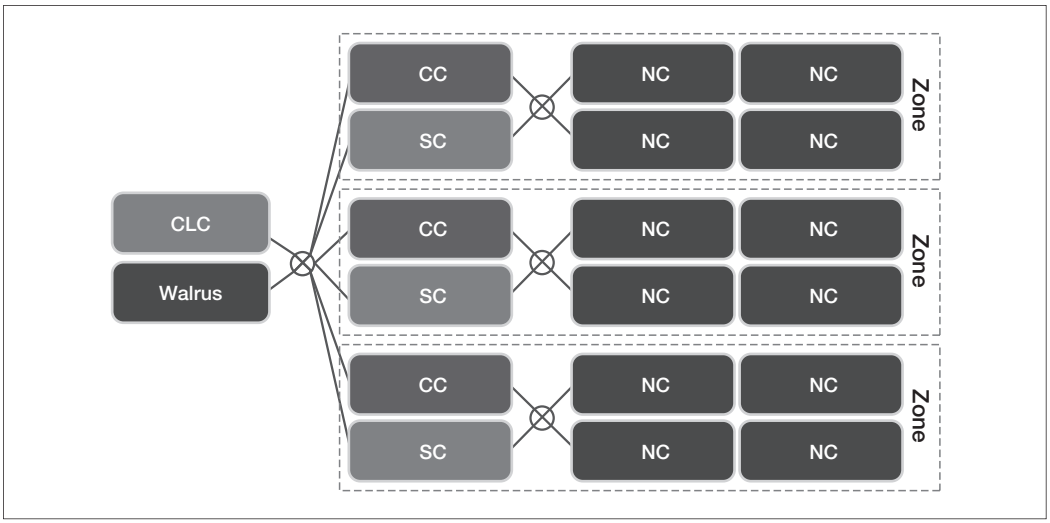


注5) http://www.eucalyptus.com/eucalyptus-cloud/community-cloud
注6) http://www.eucalyptus.com/download/faststart

▼図2　Eucalyptusの各コンポーネント



▼図3　Multi Clusterの構成例



ために課金機能は(必須ではないために)持っていませんが、レポート機能は備えているためユーザごとの利用状況を集計することは可能です。

また、EucalyptusはAWS互換を目指して開発されていますが、AWSの進化のスピードはとても早く、そのためEC2/S3/IAMのすべての機能をカバーすることは難しいのが実情です。たとえばAmazon EC2の互換性としてはAPIやいくつかの特徴的な機能を実装していますが、すべてをカバーしているわけではありません(**表1**)。ただし次のバージョン3.3ではEC2の特徴的な機能のうち3分の2がカバーされる予定です。

AWSとの互換性以外でEucalyptusがOpenStackやCloudStackと異なる点としては、Eucalyptus社が配布しているイメージを2つの

コマンド[注7]を使って簡単に自身のEucalyptus環境に登録できる機能が挙げられます。

また、冒頭で「EC2のマシンイメージが流用できる」と書きました。実際にEC2のマシンイメージを手作業でEucalyptus環境に取り込もうとすると若干の手間が発生しますが、それを簡単に行ってくれるami2emiというスクリプトが公開[注8]されています。

Eucalyptusを巡る冒険

冒頭にも述べましたが日本ではEucalyptusの人気は下火になり、実際に筆者はよく周りの人たちから「Eucalyptusってまだあるの？」とか「Eucalyptusって誰も使っていないんじゃない？」とか言われますが、今回これを読んでくださった皆さんは、Eucalyptusがまだ現役で盛んに活動しているオープンソースソフトウェアであることを再認識していただけたかと思います。

米国ではバージョン3系がリリースされてからコミュニティの活動がさらに活発になり、たびたび勉強会が催されているようですので、日本Eucalyptusユーザ会も国内においてEucalyptusに関する勉強会を開催していきたいと考えております。勉強会への要望などがありましたら、気軽に日本Eucalyptusユーザ会のメーリングリスト[注9]に投稿していただけると嬉しいです。

なお、日本Eucalyptusユーザ会の最近の活動ですが、昨年(2012年)はAdvent Calendarを開催したり[注10]、前述のユーザコンソールを翻訳しました。現在はおもにドキュメントなどの翻訳を行っています[注11]。ただし筆者は英語が得意ではないため、翻訳作業の進捗はあまり芳しくない状態です。加えて、日本Eucalyptusユーザ会のWebサイトも頻繁に更新できておらず、もう少し皆さんに情報を配信していきたいと考えておりますが、翻訳同様に人的リソースが不足しており、なかなか「下火になった人気」を回復するのが容易ではありません。

もし今回本稿を読んで興味を持ってくださった方や再びEucalyptusに触れてみようと思ってくださった方は、気軽に日本Eucalyptusユーザ会に参加していただければと思います。ちなみに最近の潮流に乗り、女子会もあるそうですので「日本Eucalyptusユーザ会のあのコアラかわぇえ」(**図4**)と思った方も気軽に参加ください。SD

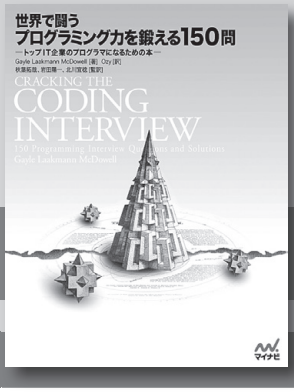
▼表1　EC2の主な機能の実装状況

Amazon EC2の特徴的な機能	Eucalyptusの実装状況
Amazon Elastic Block Store	○
EBS最適化インスタンス	○
複数のロケーション	○
Elastic IP アドレス	○
Amazon Virtual Private Cloud	-
Amazon CloudWatch	3.3に向けて検討中
Auto Scaling	3.3に向けて検討中
Elastic Load Balancing	3.3に向けて検討中
高性能コンピューティング(HPC)クラスタ	-
ハイI/Oインスタンス	3.3に向けて検討中
VM Import/Export	-
AWS Marketplace	-

◀図4
日本Eucalyptusユーザ会のマスコットキャラクタ

注7）eustore-describe-imagesコマンドとeustore-install-imageコマンド。
注8）https://github.com/eucalyptus/ami2emi
注9）http://ml.eucalyptus-users.jp/mailman/listinfo/eucalyptus-users
注10）http://036.habuka.jp/hiki/EucalyptusAdventCalendar2012JP.html
注11）http://eucatrans.habuka.jp/

BOOK no.1 世界で闘うプログラミング力を鍛える150問
トップIT企業のプログラマになるための本



Gayle Laakmann McDowell 【著】／Ozy 【訳】／秋葉 拓哉、 岩田 陽一、 北川 宜稔 【監訳】
B5変型判、 456ページ／価格＝3,759円 （税込） ／発行＝マイナビ
ISBN＝978-4-8399-4239-7

一見、プログラミングの筆記テスト対策の問題集のようだが、じつはトップIT企業の面接対策のノウハウを詰め込んだ本。どんな質問をされるか、面接までにどんな準備をしておけばよいか、MicrosoftやGoogleなどの面接はどんなものか、といったことがまとめられている。

本書の後半は、技術的な問題と解答の例がぎっしり掲載されている。これは面接だけでなく、筆記試験でも役立ちそうだ。解答例は一例だけでなく、アルゴリズム、言語、技術の異なる数パターンが紹介されているものもあり、解答例だけで本書の半分以上を占める充実ぶり。

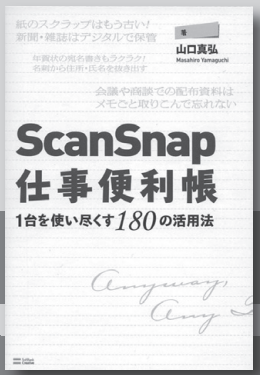
これが面接で出されるというのだから、米国IT企業のレベルの高さを感じずにはいられない。

BOOK no.2 入門 機械学習

Drew Conway、 John Myles White 【著】／萩原 正人、 奥野 陽、 水野 貴明、 木下 哲也 【訳】
B5変形判、 344ページ／価格＝3,360円 （税込） ／発行＝オライリー・ジャパン
ISBN＝978-4-87311-594-8

機械学習を使いこなすには統計や数学の理解が必須だが、本書では理論はひとまず脇に置いて、機械学習技術の使い方や実装方法を解説する。たとえば、最初に「メールをスパムとそうでないものに分類する」「Webのページビューを予測する」といった具体的な用件を挙げ、それを実現するのに必要な理論（二値分類、線形回帰など）を簡単に紹介する。そしてすぐにR言語でどう実装するかを解説する。理論の詳細よりRの関数や機能をどう組み合わせて実現するかに重きが置かれている。すぐにも実践的な機械学習技術を習得したいなら、まずは本書で実装方法の大枠をつかみ、細かい理論は必要なときにほかの専門書にあたるのが良さそうだ。

BOOK no.3 ScanSnap仕事便利帳
1台を使い尽くす180の活用法



山口 真弘 【著】
四六判、 168ページ／価格＝1,300円 （税込） ／発行＝ソフトバンク クリエイティブ
ISBN＝978-4-7973-7241-0

ScanSnapと聞くと、名刺管理や自炊を思い浮かべる人が多いのではないだろうか。しかし、それだけでは宝の持ち腐れというものだ。本書を読むにつれて、印刷物をデータ化するメリットは想像以上だと感じる。“大きさが違う書面を一括管理できる”“場所をとらない”“持ち運びが容易”“劣化しない”“複製が容易”“検索しやすい”など、キリがない。EvernoteやDropBoxを使えば、外出先でもスマホやタブレットからデータを呼び出せる。すごいなと思ったのは、読み込む書類にマーカーしておけば、部分的に抜き出して保存する機能があること。OCRしておけばテキストも抽出できる。そんな目から鱗の利用法が多く紹介されている。

BOOK no.4 サウンドプログラミング入門
音響合成の基本とC言語による実装

青木 直史 【著】
A5判、 288ページ／価格＝3,129円 （税込） ／発行＝技術評論社
ISBN＝978-4-7741-5522-7

サウンドプログラミングに興味があっても、高度な理論を理解する過程で諦めてしまった方は多いことだろう。本書では、理論はともかく、実際に動作するサンプルプログラムが紹介されているのでパラメータなどを変えながら動作させられるのがうれしい。本書の後半では、音響合成の代表的な方式として“ピコピコ”と電子音を鳴らす「PSG音源」、電子楽器の1つである「アナログシンセサイザ」、金属的な音を作り出すことができる「FM音源」、音響合成の方式として現在の主流となっている「PCM音源」が取り上げられており、それぞれの方式による音作りの方法やテクニック、加工技術が紹介されている。

光、ギガビット、高速ネットワークを体験！

実践！ワイヤリングの教科書

クラウド時代の昨今、ネットワークも高速化してきています。以前は10BASE-Tや100BASE-TXだったものが、GbE（Gigabit Ethernet）を使い、FTTH（Fiber To The Home）も広く普及してきています。本特集では、LANケーブルから光ファイバまで含めた「ワイヤリング」にスポットを当て、その技術から実際の接続まで総合的な解説を行います。

第1章

ネットワーク高速化とワイヤリングの技術解説

株式会社ブロードバンドタワー
佐伯 尊子　SAEKI Takako

15年前のネットワーク環境

ネットワークことはじめ

コンピュータ同士を接続するしくみを利用し始めてから、すでに20年以上経ちます。とくにこの15年強は、インターネットに代表される、あらゆる端末がネットワークに接続する環境に浸っている状況です。そしてこの状況は、ますます加速していくことになりそうです。

それでは、15年前の環境を確認しながら、現在～近い将来について考えてみることにしましょう。

15年前は、長野オリンピックが開催された年(1998年)です。本格的にインターネットによるリアルタイム情報更新が行われたオリンピックでもありました。配信側のIBMだけでなく、受信側の国内ISP(Internet Service Provider)、IX(Internet eXchange point)も当時の技術を総結集して当時の大規模トラフィックに対応しました。まだUstreamやYouTube、Twitterなどはありませんでしたが、テキストや静止画がリアルタイムで全世界に配信される状況は、当時としては圧巻でした。

またこの15年前は、インターネットだけでなく企業ネットワークや官公庁がオフコンなどのシステムからネットワークを介したクライアント／サーバシステムに積極的に移行していた時期でした。

さて、そのような時代のオフィスのネットワーク環境を思い出してみましょう。

社内IT管理部門に専用サーバ(室)があり、そこが社内ネットワークの中心だったのではないでしょうか？　さらに、利用するプロトコルも、「サーバ／クライアント」機は専用プロトコル、インターネット閲覧、電子メールはEthernet(およびTCP/IP)と、いくつかのプロトコルを使い分けていました。これらのデータ通信以外に、電話やファックスなどのレガシー通信もありました。

現在のネットワーク環境

概要

現在はインターネットに代表されるTCP/IPのネットワークが至るところに存在しています。データ通信だけでなく、電話やテレビ放送も、データ通信として扱うことができる時代になりました。私たちは、有線／無線を使いこなし、あらゆる場所からインターネットに接続することができます。それでは現状の課題を考えていきましょう。

LAN

まずは、ネットワークの代表格であるLANについて確認します。私たちが普段利用するPC(ノート／デスクトップ共)も、すでにGbEが標準かと思います。

100M Ethernetまでは、「100mを越えて敷設

してはいけない」「リピータやHUBの段数は2段まで」の2つを覚えておけば、それ以外あまり規格の制限を気にしなくても問題なく利用できていたかと思います。しかしGbE以降規格に紐付く留意点が増えてきたように感じます。

現在UTPを用いて一般的に利用されているGbEですが、製品が発売され始めた当初は、一部の高価なネットワーク機器向けの仕様でした。その理由の1つは光ファイバを用いたGbEが、UTPのGbEより先に立ち上がったことです。それまでのほとんどがUTPのネットワークだったにも係らず、トラフィックの急速な伸びに伴い、あわてて「光ファイバを用いてルータやスイッチをつなぐ」ネットワークを覚えなければなりませんでした。

もちろん、当時すでにFDDIを利用したネットワークを構築していたエンジニアは、光ファイバを用いることになっても、コネクタ形状の違いを確認する程度で、それなりに知識を有していました。しかし、Ethernetから光を覚え始めたエンジニアは、光ファイバのコネクタの形状1つ取っても、名称がわからないため、業者に発注指示するのに苦労していました。

さらに、利用する光ファイバの種類によって接続長が変わるため、機器の設置位置を気を付ける必要がありました。このようなことから、Ethernetの規格を深く理解する必要がありました。そんな試行錯誤から光LANの経験を積まれた方も少なくないと思います。**図1**に各規格の制定年と、利用速度について示します。

そして、ここ1～2年、10GbEは未来のことかと思っていたら、あっという間に手元で運用する時代に入ってきました。

サーバ

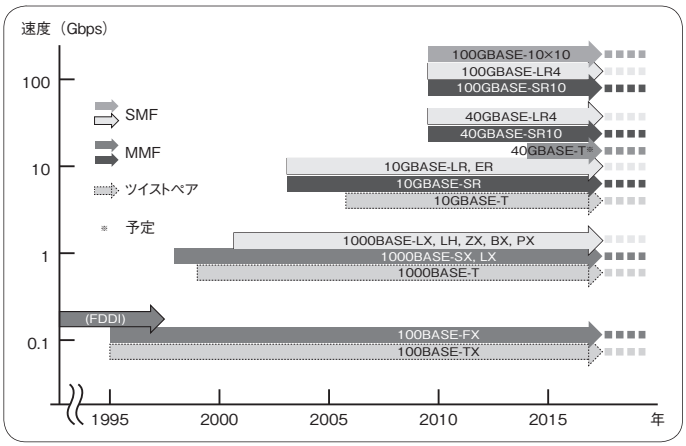
15年前との一番の違いは、仮想化技術の導入です。それまで1つの筐体に縛られていたサーバの機能を、筐体にとらわれることなく自由度高く機能を構築できるようになりました。仮想化を導入することで、サーバのリソースの最適化や利用効率の向上を図り、無駄なコストを省き、より柔軟にシステムを構築できるようになりました。

仮想化によって、物理レイヤは、何が変わったのでしょうか？

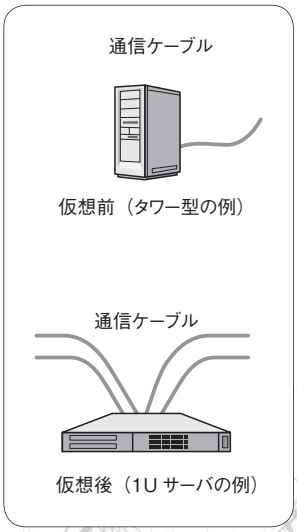
図2に、当時のサーバと、仮想化されたサーバの違いを示します。まずは形がまったく変わりました。タワー型PCが主な形状だった当時と比べ、現在は1～2Uの平たい形状で、基本ラックマウントして利用します。

さらに当時は、筐体と機能（メールサーバやファイルサーバなど）がそれぞれ1対1で対応し

▼図1　線種とLANの速度



▼図2　仮想化前後のサーバ



ていました。そのときの配線は、上位スイッチ(HUBなどを含む)に対して通信ケーブル1本だけで接続されていました。しかし仮想化したサーバの場合、単に上位スイッチとの接続だけでなく、マネージメントポートとの接続、ストレージとの接続、またVMwareであれば、VMotionとの接続など、仮想化の特徴を活かすことを考慮した複数の通信ケーブルを使っているのが実情です。

当時と比べ、サーバ自体の容積は小さくなりますが、消費電力や通信ケーブルの本数は増える傾向にあります。ラック内に設置された複数のサーバそれぞれから複数の通信ケーブルが伸びているため、ラック内、ラック間を行き交う通信ケーブルの量がとても多くなっています。

ストレージ

15年前は、ストレージを利用するユーザは限られていましたが、ここ数年サーバを単独で利用するのではなく、ストレージと併せて利用するケースが増えてきました。そしてその利用方法も、サーバとストレージを1対1で接続するのではなく、ネットワークを介して複数のサーバと複数のストレージを接続する方式を取る方法が主流になっています。ストレージもまた、より効率的に利用できるようサーバ仮想化技術とともに、いろいろな方法が急速に発達してきました。

ストレージのネットワークの主なものは、従来から利用されているFibre Channelによる SAN(Storage Area Network)、LANの一部としてのNAS(Network Attached Storage)やiSCSI(Internet Small Computer System Interface)、近年データセンター内で利用することを目的として制定されたFCoE(Fibre Channel over Ethernet)、そして高速かつ低コストなInfiniBandがあります。**図3**に、それぞれが利用するプロトコルを示します。

最新のワイヤリング事情

これだけサーバやストレージ側は技術に変化がありましたが、利用する通信ケーブルは、どう変化してきたでしょうか？

概要

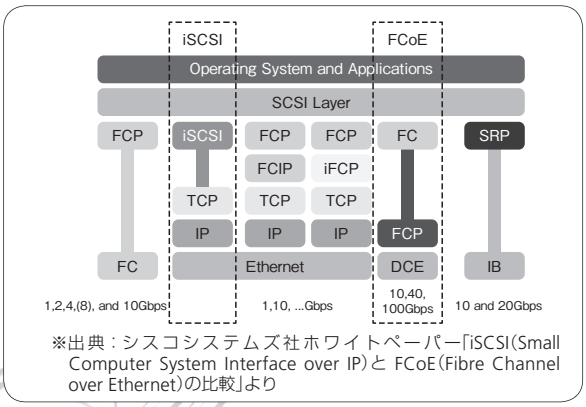
図4に、ネットワーク配線の概要を示します。それぞれ①～⑤までに分けて考えてみましょう。ただ、①のWAN回線については、それぞれ通信事業者のサービスによるところになりますので、今回は割愛します。

⑤ストレージ回線

まずは、図4の一番下に示したストレージの通信ケーブルから見ていきましょう。ストレージ部分で利用されるプロトコルは、先ほど図3で示したように、FC(Fibre Channel)、NAS、iSCSI、FCoEになります。

FC利用時、ストレージは複数または1つのストレージスイッチと接続します。このとき、HBA(Host Bus Adapter)と呼ばれるLANでいうところのNIC(Network Interface Card)に相当するボードが必要となります。これをサーバに取り付けることで、HBA＝FCスイッチ間の通信が可能になります。ここで利用される通信ケーブルは、ファイバチャネルと言われるように光

▼図3　iSCSIとFCoEのプロトコルスタックの比較



※出典：シスコシステムズ社ホワイトペーパー「iSCSI(Small Computer System Interface over IP)とFCoE(Fibre Channel over Ethernet)の比較」より

ファイバを用います。

具体的にはOM1、OM2、OS1という種類の光ファイバ(第3章参照)が利用されます。また、光ファイバは両端にコネクタの付いたもので、コネクタの種類は、SCコネクタもしくはLCコネクタが用いられます。FCの速度は、1G、2G、4G、8G、10Gbpsがあり、中でも4G、8Gbpsのものが多く利用されています。

次にNASやiSCSIですが、これらは基本的にLANと同じプロトコルで動きます。通信速度も、LANの速度となります。そのため通信ケーブルは、図4-②や④で利用しているUTPや、光ファイバをそのまま利用できます。

④LANサーバ回線

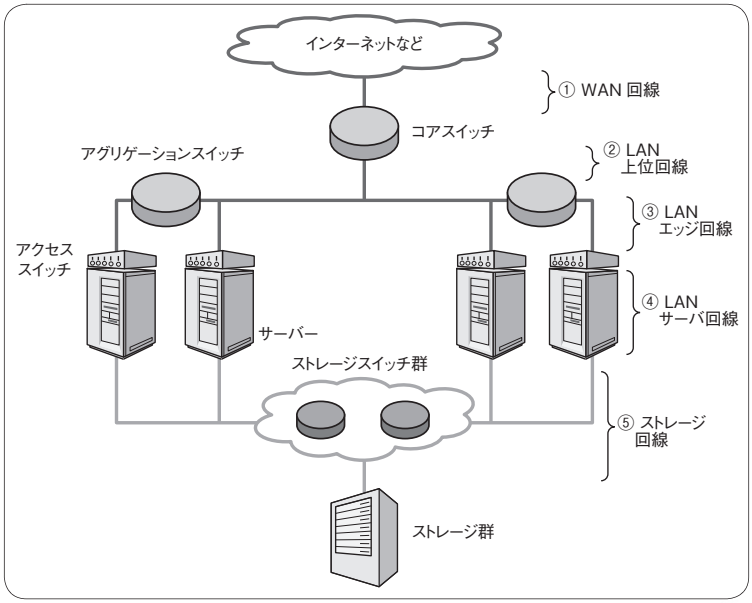
アクセススイッチ＝サーバ間の配線です。現在はタワー型のPCサーバを直置きして接続することはほとんどないと思います。1U/2Uサーバをラックに搭載して、直近のアクセススイッチと接続します。ラックに搭載する際に一番上の段にスイッチを設置することから、アクセススイッチをTOR(Top Of Rack)スイッチと呼ぶことがあります。

ラッキングしたときの模式的な絵を**図5**に示します。この名称は機能であるため、必ずしも上部にスイッチを設置しなくても、同じラック内のどこに設置しても、TORスイッチとなります。最近は、このラック1本を1モジュールと考え、ラック単位で増設することとしている例を見受けます。ラック1本が大きなサーバとスイッチと考え、ネットワークの考え方をシンプルにしようというものです。このTORスイッチとサーバ間の配線は、利用ケーブルの最小～最大長がだいたい決まります。規格では0.5～15mに限定しています。というのも、ラックの架列をまたいで、はるか向こうのサーバとこちらのスイッチを接続する必要がないからです。

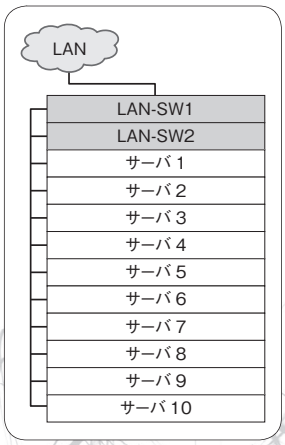
●Ethernet(UTP)

一番多く用いられるのは、UTPですが、一般的に利用しているカテゴリ5eのケーブルとRJ-45のモジュラープラグでは、最大でもGbEまでしか適応できません。しかし、現在1Gbps以上の速度を求められることが増えてきました。そのため、それ以上の速度をサポートするツイストペアケーブルを使う時代が到来したと言え

▼図4　ネットワーク配線概要



▼図5　サーバとスイッチ



ます。

本章の最初の図1で示したように、10GbE対応のツイストペアケーブルや、光ファイバケーブルを積極的に利用していく時代になってきました。**表1**に10GbEの通信ケーブルと距離の規格を示します。

InfiniBand

次にIB(InfiniBand)です。これは、サーバ側にNICではなくHCA(Host Channel Adapter)を用い、IBスイッチと接続します。利用速度は、10G、20G、40G、56Gbpsの4つから選べます。また近い将来、112Gbpsの製品も販売される予定です。FC、NAS、FCoEなどいくつかのプロトコルがある中で、高速であるにもかかわらず安価に実現できるのがIBです。そしてIBは、今までのケーブルとは違う構造を持つ専用のケーブルを用意する必要があります。

それはUTPでも光ファイバでもなく、同軸ケーブルです。それも同軸ケーブル＋コネクタという形状ではなく、両端に直接モジュールが取りついているDAC(Digital Attached Cable)です。そのため、中のケーブルが同軸であれ、光ファイバであれ、UTPであれ、ケーブルの種類を気にかける必要がありません。さらに敷設の際、ケーブル部分の測定をする必要もありません。単に機器のHCA部分に差し込めば良いという手軽なケーブルです。

規格化されている長さは、0.5m～15mまでとなります。**図6**にDACの例を示します。この15mを越して利用する場合は、両端にモジュールの付いた光ファイバ(AOC：Active Optical Cable)を用います。規格では100m～300m程度をサポートしています。光ファイバの場合も、両端にモジュールが付いているため、モジュール部分を専用の用具で清掃したりする必要がありません。ただこれだけ手軽に接続できるIBケーブルですが、利用するときに気をつけたいことを挙げてみました。

▼表1　10GbE規格(10GBASE-X(メタル))

規格	IEEE 802.3an、IEEE 802.3ak、IEEE802.3ap 10GBASE-			
	T			CX4
変調方式	PAM-16/128DSQ			64B/66B
伝送速度	3.125Gbps×4			10.3125Gbps
利用対数	4対			4対×2
種別	Cat6	Cat6A	Cat7	同軸
伝送距離	37m(※)	100m	100m	15m

(※)条件によっては55mまで延長可能。

(1)機器とモジュールの相性

機器メーカーが推奨するDAC/AOCを用いること。モジュール外寸は各社同じサイズになっていますが、実際には接続する機器との相性があるため、メーカー推奨品以外は動作しない可能性があります。

(2)ケーブルの測定不可能

ケーブル部分の測定をしなくて良いということは、逆に測定ができないということです。そのため、障害個所がケーブル内にあった場合、その障害点を見つけ出すことが困難になります。さらにAOCの場合、いくら両端のモジュールを扱うだけだからと言って、ケーブル部の機械的特性は、一般の光ケーブルと変わらないため、許容曲げ半径よりも厳しく曲げたり、ひねったりしないように気をつけて接続することが重要です。

▼図6　IBケーブル例

※Mellanox Technologie社：
Mellanox Copper Cable QSFP

せっかく購入したDACやAOCであっても、機器との相性や、ケーブル部の損傷に起因した不具合で、全取り替えになる場合もあります。そうすると、FCやNAS、FCoEのように、ケーブルだけ差し替えるよりもコスト増になってしまいます。これらのことをふまえて、お手軽とはいえ、接続時は慎重に作業することを心がけましょう。

●FCoE

FCoEを用いる場合、このTORスイッチ＝サーバ間は10GbEで接続しなければなりません。FCoEは、Ethernetと表記されてはいますが、実際には通常のEthernetとは違う新しいプロトコルと認識したほうが間違いありません[注1]。DCB（Data Center Bridging）という技術を用い、パケットロスなどをほとんどなくすなどの工夫を図ることで、FCのプロトコルをカプセリングして利用できます。このDCBの規格を満足させるためには、10GbEが必須となるわけです。

●仮想化

こうしてTORスイッチ、アクセススイッチ＝サーバ間は、規格の側面から高速性を要求されることがわかりました。そしてこの高速化を必要とするもう1つの理由が仮想化です。先ほど仮想化したサーバは、単一のサーバと比べ、通信ケーブルの本数が多くなることを説明しました。そのとき仮想化することで、通信ケーブルの本数を増やすのではなく、1本の高速なケーブル上に各種の通信をすべて載せてしまうことが可能となりました。

たとえば今までは GbEの通信ケーブルで1本1本がそれぞれ上位LANスイッチやストレージと接続していましたが、それを束ねて10GbEの通信ケーブルに重畳（ちょうじょう）させることが可能となりました。これで配線の本数を大幅に減らすことが可能となります。配線の本数が減ることで、通信ケーブルの管理が楽になったり、サーバが設置されているラック内の風通しがよくなり、サーバに十分冷風を行き渡らせることができるようになります。さらに1本の高速回線内を自由度高く仕切ることができます。

10Gbpsの中で「今日はバックアップを取るからバックアップ用の帯域を十分に確保しよう」「今日はイベントでサーバアクセス数が激増することが予想されるから、LANの帯域を十分に確保しよう」ということが可能となります。トラフィックコントロールなどを、いちいち現場で通信ケーブルを挿抜（そうばつ）し、機器の設定を見直さなくとも、リモートで簡単に設定できるため、プロビジョニング業務をスムーズに行うことができます。

ここで注意しておきたいことは、単に通信ケーブルの本数が減って管理が楽になるという側面だけでなく、不具合の側面もあります。万が一、その通信ケーブルが切れたときにどう復旧させるかが、非常に難しくなります。そのうえ、マネジメント回線も重畳させていた場合は、機器の状態を確認することさえ難しくなります。そのため、この高速かつ重要な通信ケーブルを、自作UTP/STPを用いて、サーバとスイッチ間を、15年前と同じような感覚の元自分たちで接続することは非常に危険な作業であると認識すべきかと思います。

②LAN上位回線

今まで確認したように、図5-③～④の部分は、すでに10GbEを必要としているプロトコルがあることがわかりました。そのため、各アグリゲーションスイッチとコアスイッチを集結する部分には、当然ながら10GbE以上の速度を持つ配線が必要になります。ここで必要となるのが40G/100Gです。

まず**表2**に10GbEの光ファイバの規格を示します。

10GbEは、GbE同様送受信に2芯の光ファイバを用います。このとき注意しなくてはいけな

注1　ANSI INCITS T11でFC-BB-5として標準化が完了。

いのが、MMF（Multi Mode optical Fiber）の種類です。従来コア径が62.5 μmか、50 μmかで伝送距離が変わることは、ご存じかと思います。しかし、50 μmのコア径であっても、光ファイバの種類によって伝送距離が違ってきます。

たとえば10GBASE-SRをとって見てみましょう。OM2の場合の伝送距離は82mと、100mにおよびません。しかし、同じコア径であってもOM3であれば、300mまで利用できます。さらにグレードの高いOM4というMMFもコア径50 μmですが、伝送距離はOM3よりさらに延びます。したがって、コア径だけで、MMFを判断するのではなく、その他の情報も加味してMMFの種類を確認してください。

そして、注意すべきは、従来から利用しているOM2は、40GbE以降では利用できなくなります。40GbEの場合、MMFはOM3以上のグレードのものに限定されます。どうして同じ50 μmのコア径の光ファイバであるにもかかわらず、利用できるファイバが限定されるのでしょうか？　それは、モジュール側の光源に秘密があります。

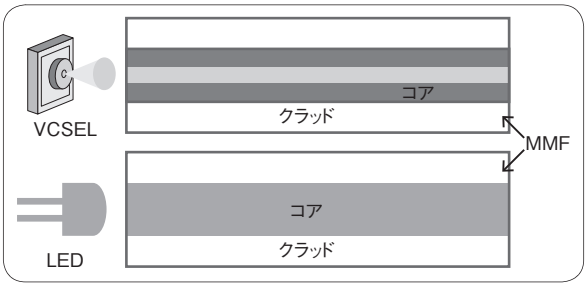
GbEまでは、LED光源を用いていたのですが、10GbEの光モジュールを作るにあたってはLED光源で実現できませんでした。そのため10GbEは新しい光源（VCSEL（Vertical-Cavity Surface-Emitting Laser；面発光）レーザ）を用いることになりました。**図7**に概要を示します。

GbEまではコアの中すべてに光源の光が行き渡っていたのですが、VCSELレーザになると、コアの中心のほんの一部しか光は通りません。光源の波形がレーザ光のため、より高速な伝送をすることができます。そのため10GbE以上の高速な伝送に対応できるのですが、コアの中心だけしか光が通過しないため、従来のOM1/OM2ではVCSELレーザの長所を引き出した伝送ができません。そのためVCSELレーザに最適化されたMMFが必要となり、OM3やOM4が開発されました。それぞれのファイバの特徴は、第3章をご覧ください。

そのようにVCSELレーザに最適化されたOM3/OM4ですが、40GbEの伝送となると、さらに4芯送信4芯受信合計8芯が必要になります。さらに100GbEになると、10芯送信10芯受信となります。SMF（OS1）であれば、1芯に4波を重ねることで対応しているため、芯数自体は2芯を利用して送受信します。

このように、40GbEや100GbEをMMFで利

▼図7　光源の違い



▼表2　10GbE規格（10GBASE-X（光ファイバ：LAN PHY））

規格	IEEE 802.3ae 10GBASE-							IEEE802.3aq 10GBASE-	
	LX4		SR			LR	ER	LRM	
変調方式	8B／10B		64B／66B						
伝送速度	3.125Gbps × 4		10.3125Gbps						
波長種別	850nm	1310nm WDM	850nm			1310nm	1550nm	1300nm	
光ファイバ種別	OM1	OS1	OM1	OM2	OM3	OS1		OM1	OM3
伝送距離	300m	10km	26m	82m	300m	10km	40km	220m	260m

用する場合は、一度に利用する芯数がとても多くなります。したがって、今まで利用していたSCコネクタやLCコネクタでは対応できません。そこで利用されているのがMPOコネクタです。MMFを用いた40GbE/100GbEの場合は、このMPOコネクタを用いて機器と接続します。SMFの場合は従来どおりSCコネクタやLCコネクタになります。**図8**にMPOコネクタを示します。黒い部分の中心に12芯/24芯分の穴が開いており、そこに光ファイバを挿入し、端面を研磨して接続します。

このように、単に光ファイバであれば高速伝送できていた時代は終了しました。MMFを利用する際は、光源の種類に注意する必要があります。とくにVCSELレーザを用いた40GbE/100GbEを伝送する場合は、OM3やOM4を複数芯用いる配線が要求される時代になりました。

配線形態

概要

通信配線というと、通信ケーブルの線の種類だけに注目しがちですが、実はその配線のトポロジーにも注意を払う必要があります。

サーバの仮想化と高速化により、簡単に論理ネットワークを構築することができるようになりました。そのため、物理配線については「問題なく接続されていて当たり前」であると思い込みがちです。しかし、実際にトラブルがあったときは、その物理配線が原因であることも少なくありません。そのため、物理配線のトポロジーを理解することは大切なことです。

▼図8　MPOコネクタ

※出典：古河電工ホームページより
（http://www.furukawa.co.jp/what/2002/connect020315.htm）

LANのツリー型

図5-②～④に示したように、LANはツリー型（スター型）です。この形状はEthernetの配線形態から来ています。Ethernetの初期は、10BASE-5など同軸を用いてバス型に接続していましたが、10BASE-T以降ツイストペアケーブルを利用することにより、HUBを介して多対多の接続ができるようになりました。これがスター型へのステップでした。

その後各スイッチなどの役割を精査してみると、WAN回線と接続し外部と接続する上位のコアスイッチ部分と、サーバや端末PCを接続するアクセススイッチ部分では機能が分かれてきました。そして、その（物理／論理）トポロジーは、おのずとツリー型になっていきました。

当然、Ethernet以外のプロトコルを利用していた場合は、また違った物理トポロジーになっていたかと思われます。

そして、この考えは、単にオフィスネットワークなどにとどまらず、巨大なデータセンターでもツリー型配線が規格化されています。**図9**にTIA-942A Telecommunicions Infrastructure Standard For Data Centersで定めている物理トポロジーを示します。明らかにツリー型です。これはEthernetをベースとしているためです。

FCoEのメッシュ型

そのデータセンターの配線トポロジーが、今変わりつつあります。先ほどお話ししたように、FCoEは、Ethernetとは別のDCBを利用しています。このDCBの配線は、スイッチ同士はほぼすべて接続する方式になっています。このときのスイッチは、データセンターファブリックスイッチと呼ばれ、スイッチ同士が接続されていることによって、自動でその設定内容を設定／

補完し合うことができる機能を有します。現在、TIAのデータセンターのワーキンググループでは、この配線トポロジーについて精査しているところです。近々TIA-942Aの追補1として、追記される予定です。このワーキンググループで検討しているトポロジー案を**図10**に示します。Fat-Tree(高速回線のツリー型)で、リーフ&スパインスイッチの利用です。リーフスイッチ(図5のアクセススイッチ相当)と、スパインスイッチ(図5のアグリゲーションスイッチ相当)が、FCoEプロトコルに基づき、メッシュで接続されています。

スター型配線の配線経路は1ルート(もしくは、冗長化のために2ルート)でしたが、メッシュ型になると、その経路は複数になり、かつ新しいリーフスイッチを導入するたびに、大量の配線が必要になります。そのため、配線経路は今以上に潤沢に確保しておく必要があることに留意してください。

先ほど、FCoEの配線は1本の太い回線のため、不具合を起こさないよう細心の注意を払うべきだと話しましたが、それはリーフスイッチ=サーバ間の配線のことを指します。この部分も仮想化かつ冗長化されていれば、1つのサーバ=リーフスイッチ間で不具合があったとしても、サービス自体がすぐに停止に至るわけではありません。このように物理的にも論理的にも冗長化を図ることで、より可用性の高いサービスを構築できるようになります。

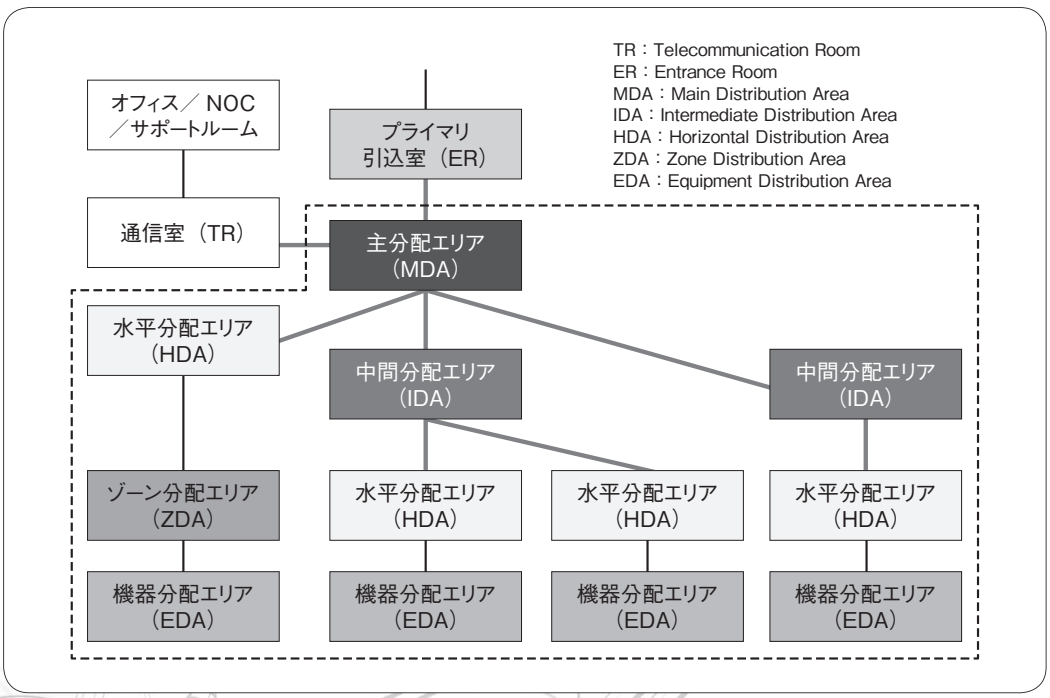
SDN(OpenFlow)のコントロール型(?)

最後に、最近のホットトピックとして、SDN(Software-Defined Network)の配線トポロジーについても、触れておきましょう。おもにOpenFlowを用いた配線形態です。ホップバイホップ方式と、オーバーレイ方式に分かれます。

●ホップバイホップ方式

OpenFlowに対応した機器同士をすべて接続しコントロールする方式。この時すべてが物理

▼図9　データセンタの配線(TIA-942A データセンタ規格の配線トポロジー)



的に接続されていなくても論理的に接続されていることでOKです。

●オーバーレイ方式

仮想化されていることを前提に、トンネリングによる論理接続のみで機器をコントロールする方式。

これらは、物理的に「このようなトポロジーで配線しなければならない」という決まりはありません。また、利用する通信速度がGbEであっても（FCoEのように必ず10GbEを利用しなければならないという制限があるわけでもないため）動作します。そのため、通信ケーブルは今まで慣れ親しんだカテゴリ5eのUTPでも実現できます。

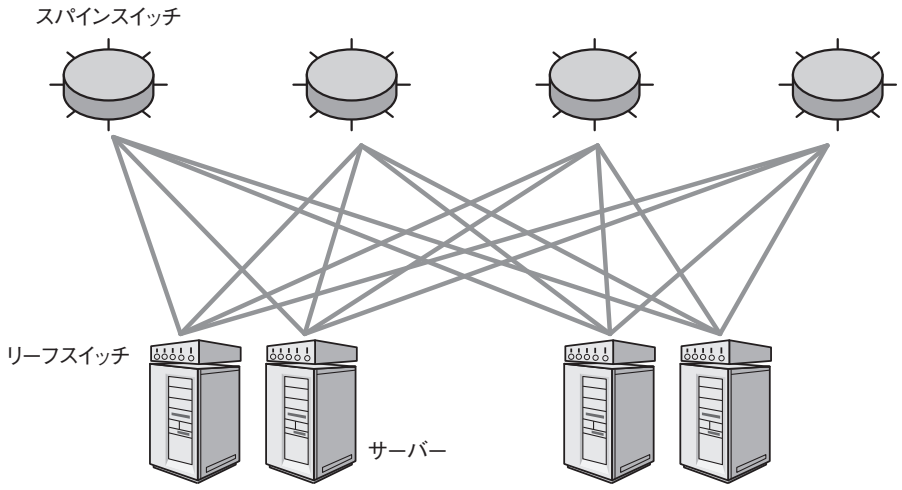
SDN（OpenFlow）の普及は、今後の配線形態にも大きく影響を与えることになりそうです。

このように、我々が利用しているLAN配線の世界も、レイヤが上の世界の新しい技術に影響を受けながら、新しい技術や線種を利用して、新陳代謝を図っていく必要があると思います。安価だから、扱いやすいからと、いつまでも同じ線種ばかりを利用するのではなく、新しい技術とともに新しい通信ケーブルを使ってほしいと思います。SD

●参考資料

BICSI：https://www.bicsi.org/
ISO：http://www.iso.org/iso/home.html
IEC：http://www.iec.ch/
IEEE 802.3：http://www.ieee802.org/3/
TIA：http://www.tiaonline.org/
IBM：http://www.ibm.com/jp/ja/
ONF：https://www.opennetworking.org/
CISCO Systems：http://www.cisco.com/web/JP/
Brocade Communications：http://www.brocadejapan.com/
Mellanox：http://www.mellanox.co.jp/
通信興業：http://www.tsuko.co.jp/

▼図10　ファブリック配線トポロジー



※出典：2012BICSI 日本支部カンファレンス資料「Updates to TIA & ISO data center standards to reflect industry changes」より

第2章

奥が深いLANケーブルの構造と配線技術

職業能力開発総合大学校
菊池 拓男　KIKUCHI Takuo

はじめに

本章では、LAN構築現場で実際に必要とされるLANケーブルの知識と配線技術について解説します。

LANケーブルの性能と構造

LANケーブルとして一般的に使われているものは、ツイストペアケーブルと呼ばれるものです。その性能や構造によっていくつかに分けられるので、それを紹介します。情報配線システムの用途に応じて最適なものを選択することが必要です。

カテゴリ

LAN配線の性能を表す言葉として“カテゴリ”と“クラス”があります。少しややこしいですが、カテゴリはケーブルやモジュラコネクタなどの性能、クラスは配線全体の性能を示しています。たとえば、カテゴリ5のケーブルや部材は配線性能クラスDを提供し、カテゴリ6の部材は配線性能クラスEを提供します。

なお、カテゴリ5e[enhanced]はカテゴリ5の改良版です。カテゴリ6Aの[A]は「Augmented」の略です。

現在、Ethernetの高速化により、より帯域が広いケーブルが必要になっています。また、カテゴリ6やカテゴリ6Aのケーブルは、その性能を満たすため、各対のよりを増やしたり、ケーブル構造に工夫をしたりしています。

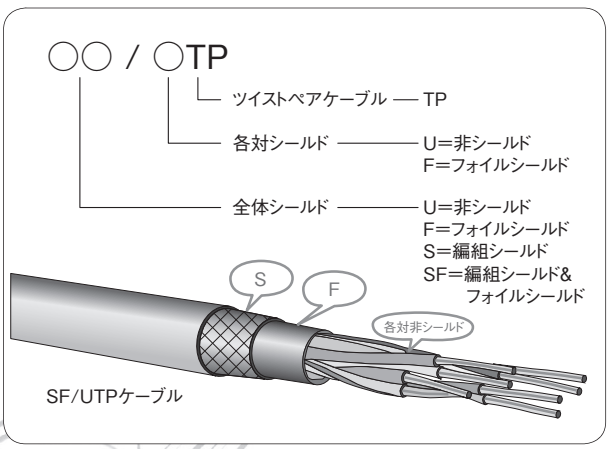
構造

ツイストペアケーブルは、その構造により、おもにLANで使用されているU/UTPケーブル、シールド配線として使用されているF/UTPに分類されます。なお、この分類方法はISO(JIS)によるものであり、一般的に国内で見か

▼表1　カテゴリとクラス

カテゴリ名	クラス名	保証周波数帯域
カテゴリ3	クラスC	～16MHz
カテゴリ5	クラスD	～100MHz
カテゴリ6	クラスE	～250MHz
カテゴリ6A	クラスEA	～500MHz

▼図1　ツイストペアケーブルの表記法



けるものとは若干異なります。この表記法は、**図1**のルールにより表記されています。

●U/UTPケーブル

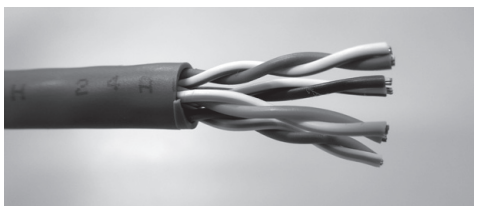
U/UTP(Unshielded Twisted Pair cable)ケーブルは、現在最も使用されているケーブルでよく目にするものです(**写真1**)。外被の内側にシールド(全体シールド)がなく、各対へのシールドもない構造です。

●F/UTPケーブル

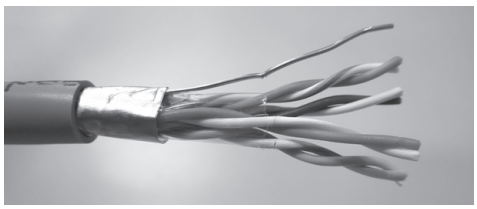
F/UTP(Foiled-unshielded Twisted Pair cable)ケーブルは、外被の内側に全体をシールドする薄い金属箔(フォイル)があり、各対へのシールドがない構造です(**写真2**)。

このほか、外被の内側に全体をシールドするフォイルシールドと網組シールドがあるSF/UTPケーブルや、各対にもシールドがあるカテゴリ7ケーブルで使用されているS/FTPケーブル(**写真3**)があります。

▼写真1　U/UTPケーブル



▼写真2　F/UTPケーブル



▼写真3　S/FTPケーブル

意外と知らないケーブルの構造

ここまで基本的なケーブル構造について解説しましたが、ケーブルの構造にはこのようなものもあります。

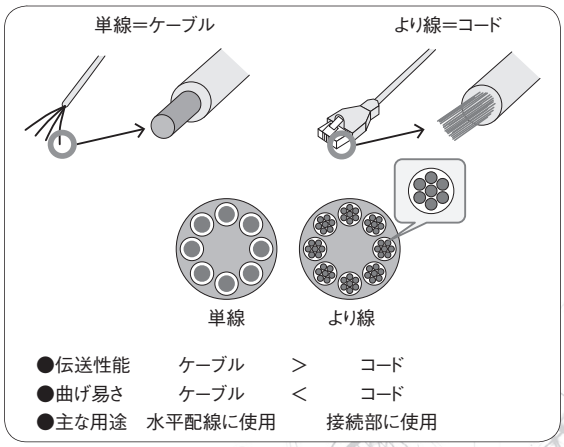
単線とより線

ケーブルには単線とより線があります(**図2**)。通常、パソコンまわりで使用しているモジュラプラグが付いた柔らかめのケーブルはより線です。一方、モジュラジャックが成端[注1]されているケーブルは単線です。

図2からわかるようにモジュラプラグが成端されるケーブルの芯線はより線です。したがって、モジュラプラグは、一般的により線用です。このことを知らずにモジュラプラグを作っている方も見受けられます。単線を成端するとうまく成端できない場合があるのです(より線と単線の両方に使えるものもあります)。パッチコード[注2]が調子が悪い……などという場合は一度調べてみてください。また、単線の場合、カテゴリが高くなるほど芯線の太さは太くなります。

注1　ケーブルやワイヤの先端に接続用端子を取り付けること。
注2　ケーブル両端にモジュラプラグが成端されたもの。

▼図2　単線とより線



AケーブルとBケーブル?

モジュラプラグ(RJ45)は共通ですが、結線の方法としてはT568AとT568Bがあります[注3]。実はケーブル構造にもこの差があるのです。一度、お使いのケーブルを見てみてください。青の対の向かい側に何色がありますか？　橙色がある場合はAケーブル、緑色がある場合はBケーブルです。したがって、Aケーブルの場合はT568Aで成端するべきなのです。RJ45については、後述するLANケーブルの作り方でも改めて取り上げます。

カテゴリが大きくなると?

各芯線を対にしているよりのピッチは、対の色ごとに異なっています。また、カテゴリが大きくなるにつれて、このよりのピッチは細かくなります。さらに、ケーブル中央に十字介在などを付けたり、ケーブル断面を楕円形にしたりして、ケーブルの構造を工夫もしています。これらはみな、ケーブル自身ができるノイズ対策なのです。

非常に重要な配線技術

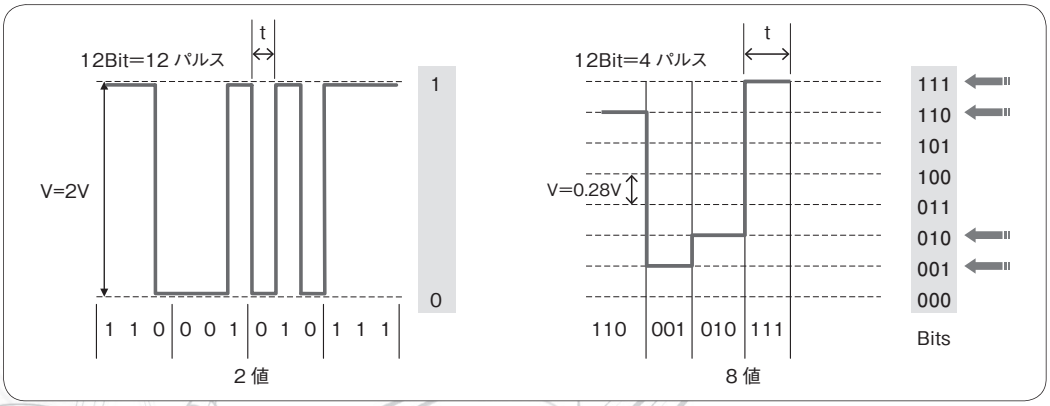
ここ数年、LAN技術は大きく進歩しており、ギガビットEthernetも一般的なものとなっています。それに対応したネットワーク機器には関心があるものの、インフラとなる配線にはあまり関心がないという方も多いのではないでしょうか。ある統計によるとネットワーク障害の原因の7割は配線にある、ともいわれているくらいです。かたや配線、されど配線です。もはや素人が見よう見まねで配線する、という時代ではないのです。ネットワーク機器が変わっても配線は今後も長い間使用するものです。普段、床下や壁の中で目にしない配線が、いかに重要であるのかを解説していきます。

Ethernetでは、100Mbpsでは3値、1Gbpsでは5値、10Gbpsでは16値ものコーディング(符号化方式)が用いられています。

図3は8値と2値の符号化を比較しています。2値の場合12個のパルスが必要ですが、8値では4個のパルスで表すことができます(このパルスが周波数に相当します)。一方、各信号を表すレベル(電圧)は2値の場合は2Vとすると、8値の場合は0.28Vとなってしまいます。周波数をどんどん高くするとそれに対応したケーブルや部材が必要になります。また、どんどん信号あたりの電圧が小さくなりますから、ノイズに非常に弱くなっているのです。これらのことから、現在の1000Base-Tなどは従来の10Base-Tとは、もはや比べものにならないほど技術レベルが進歩しています。そのことに対応した配線技

注3　市販のストレートケーブルは両端がT568Bになっています。片方をT568Aにするとクロスケーブルになります。

▼図3　2値と多値の符号化の違い(参考資料：R&Mホワイトペーパー)



術が必要なのですが、残念ながら関心度は低いままです。繰り返しますが、もはや誰でもLAN配線はできる、という時代ではないのです。

LANケーブルの配線でやってはいけないこと

ケーブル配線には守るべきルールがあります。これを「規格」と呼んでいますが、LAN配線の規格は、日本工業規格ではJIS X 5150であり、それ以外にもANSI-TIA/EIA-568.C、ISO 11801などがあります。ここではそれらの規格に書かれている重要事項について解説しましょう。

●ケーブルの対のより戻し

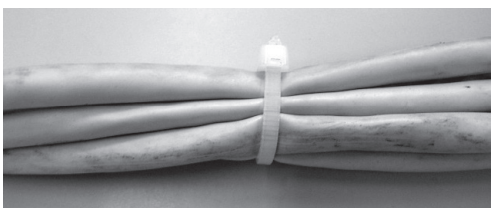
ツイストペアケーブルはノイズの影響を最小限にするため、ケーブルの芯線に「より」を作っています。そのよりをできる限り戻さないこと、戻るような原因を作らないこと、が肝心です。よりが大きく戻る可能性があるのは、ケーブルを成端するとき、ケーブルがねじれているとき、曲げるときです。

●曲げ

ケーブルにはその特性を維持するために守るべき最小の曲げ半径が決まっています。決められた曲げ半径を超えて曲げた場合は、漏話(ろうわ)[注4]が生じたり、反射が生じたりして、ケーブルの特性が悪くなります。規格では4対UTPケーブルの水平配線時の許容曲げ半径は、無負荷条件(敷設後の状態)でケーブル外径の4倍以上(半径約25mmくらい)、敷設中の負荷条件(引っ張りに対する)では、ケーブル外径の8倍です。

注4) 伝送中の信号が信号線から漏れること。

▼写真4　固定バンドによる締めすぎ



●張力と側圧

ケーブルを強い力で引くと、その力で心線の対より構造が乱れ特性劣化の原因となります。したがって、ケーブルを配線する際は、許容された張力以下で行わなければなりません。また、ケーブルが潰れるなど構造が変化するような力を加えてもいけません。とくにケーブルを固定するときや整線するときには、ケーブル外被が変形するほど強く締め付けないよう注意しましょう。固定バンドなどでまとめることはなるべく避けましょう(**写真4**)。

配線施工の実際

このような基本をふまえたうえで、実際にサーバ周りでどのような配線を行うべきか、見ていきましょう。

サーバラックの周りの配線

サーバルームなどにある19インチラックへの配線時には、次のポイントに注意します。

●自作ケーブルは品質保証できない

クロスコネクトジャンパ、パッチコード、機器コード[注5]などに用いるケーブルは、それを接続する水平ケーブルと同等以上のカテゴリを使用しなければなりません。また、パッチコードは高品質の確保のため、自前で作成することは避けてください。品質が保証されたパッチコードを使用しなければいけません。

●カテゴリ6ケーブルは整線してはいけない

ケーブルには余長が必ず必要です。成端するジャックには寿命があり、抜き差しを繰り返した場合に再成端する必要が出てきます。その場合、余長がなければ再成端できす、ケーブルを張りかえなければいけません。また、将来的なパネルやポートの入れ替え、機器の移動の可能性を考慮し、対応できる長さを確保しなければ

注5) 機器を配線盤に接続するコード。

なりません。さらにケーブルはメンテナンス性や美観を十分に考慮し整線しなければなりません(**写真5**)。

ただし、カテゴリ6以上のU/UTPケーブルの整線には気をつけなければなりません。カテゴリ6のケーブルは高い周波数を伝送するため、ケーブル内だけでなく隣接したケーブル間でエイリアンクロストーク(近接漏話)が発生することがあります。したがって、カテゴリ6のシールドなしUTPケーブルを複数本並べて敷設する際は、同じ色の対が平行に隣接しないように、長い距離を密接して平行に配置しないことが重要です(**写真6**)。つまり、なるべく乱雑に配置する(「スパゲティ配線」と呼んでいます)などの工夫が必要になります。余長も同様です。余長をまるめておくと、長い距離ケーブルを配線している状況と同じになるからです。

●管理とセキュリティに配慮する

ケーブル管理のためにラベリングを必ず行いましょう。パッチコードなどは誤接続しやすいもので、それを避けるためにも必要です。また、ネットワークのセキュリティで一番危険であるのは配線系であるという認識を持ち、モジュラジャックの口に鍵のかかる蓋をしたり、パッチコードも鍵がなければ抜けないようにしたりして配線自体を閉じた系にすることも検討の必要があるでしょう。

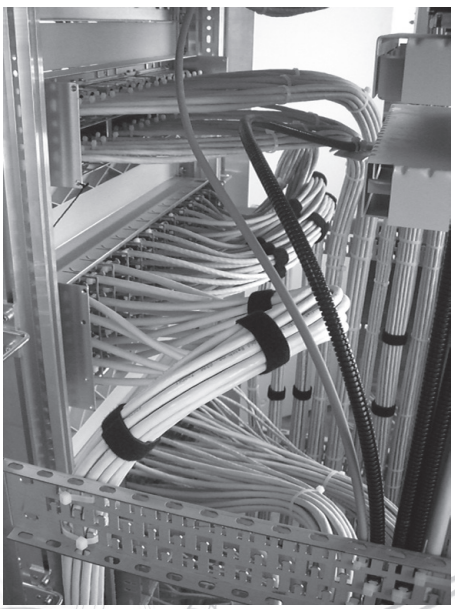
●ノイズを避ける

近年増加しているシールドケーブルを成端する場合には、同時に接地工事が必要になるので注意が必要です。ケーブルのすべてのシールドは適切に各配線盤で設置しなければなりません。また、通線ルートの環境を十分把握し、ノイズ源から避けた配線をすることも重要です。

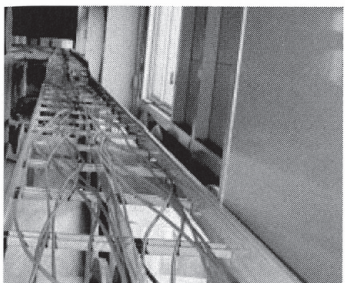
オフィス周りの配線

普段、筆者達が仕事をしている場、ワークエリアでもさまざまな配線上の決まりがあります。たとえば、「ワークエリアコードは5m以下であることが望ましい」という決まりがあるのはご存じでしたか？ これはつまり、PCから通信アウトレット(情報用コンセント)まで伸びたケーブルの長さは5m以下にしてほしい、ということです。みなさんの周りはどうでしょうか？ 筆者

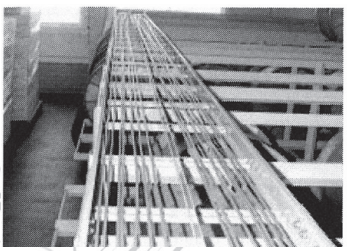
▼写真5 整線されたケーブル



▼写真6a UTPケーブルが平行でない場合(これで正解)(参考資料：INIP-Silver認定テキスト)



▼写真6b UTPケーブルが平行になっている場合



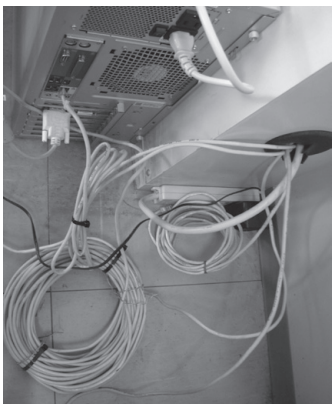
の周りにも5m以上のケーブルは結構あります。これはネットワークの品質を落としている原因なのです。これ以外にも、次のような決まりがあります。

・ワークエリアコードの品質（カテゴリなど）は、水平配線ケーブルの品質以上にする
・ねじれ、キンク[注6]、過度の曲げを作らない
・ケーブルを机や椅子などでつぶしたり、過度の側圧を加えない

また本来、水平配線はケーブルの両端にモジュラジャックを成端し、通信アウトレットを設置して、そこからパッチコードを使ってネットワーク機器に接続するとルールで決められています。**写真7**はよく目にする配線ですが、このようにモジュラプラグが床から出ている、なんてことはあってはいけないのです。電気のプラグが壁から出ている、なんてことがないのと同じです。

注6　ケーブルが極端にねじれ、くの字型に折れ曲がった状態。

▼写真7　オフィス周りでよく目にする「やってはいけない配線」



プロに聞く、LANケーブルの簡単な作り方

RJ45コネクタの作成

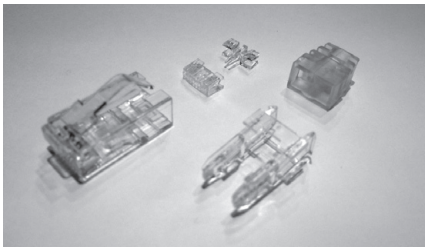
RJ45コネクタは8極8芯で、メス型のモジュラジャック（**写真8**）とオス型のモジュラプラグ（**写真9**）に分けられます。モジュラジャックの成端はケーブルリンクを構築する際にケーブル両端で必要になる作業です。モジュラプラグの成端はパッチコードを作成する際に必要となります。モジュラプラグの成端は、みなさんも一度は体験したことがあるかもしれません。しかしながら、現在の高速LANの配線では、けっして素人が見よう見まねで行う作業ではありません。写真9のように部品の点数も多く、作成には特別な技術が必要です。ただ、何らかの理由により自作することや、配線をチェックする場合もあると思います。ここではそのポイントを解説していきます。

使用工具

モジュラコネクタの成端には、ケーブルストリッパ（**写真10**）、かしめ工具（**写真11**）が必要です。ケーブルストリッパは、ケーブル外被を剥く際に芯線に傷がつかないよう、刃の出方などを調整をしなければなりません。また、かしめ工具もどれを使っても同じと思われるかもしれませんが、使用するモジュラ

▼写真8　モジュラジャック（NR3061）

▼写真9　モジュラプラグ（SP688-C）



▼写真10　ケーブルストリッパ

▼写真11　かしめ工具

プラグに対応した専用の工具を使用しなければいけません。

モジュラコネクタのチェックポイント

成端の際には、次のポイントに注意してください。これらは目視により確認できますので、必ず点検を行いましょう。

●結線(ピン割り当て)

ケーブルを成端する際に、定められたピン割り当てどおりに結線を行わなければなりません。仮に正しい結線方法ではなくても、ケーブルの心線がつながっていれば電気的導通は取れますが、それぞれのピンには役割があるため通信が正確には行えません。たとえば、ピンの配置を守らず間違った結線を行うと、10BASE-Tではある程度の長さまで正常に動作しますが、100BASE-TXではノイズの影響により10m程度を超えるとエラーとなる可能性があります。

●より戻し長

より戻し長はできる限り短くしなければなりません。ケーブルの外被を必要以上に除去すると、その分だけより戻し部分が増えるため、成端に必要な部分のみを除去してください。プラグやジャックなどの成端部分では、対のより戻しはできる限り小さくしましょう。参考値として、カテゴリ5のケーブルの成端ではケーブルのよりを成端点(切り口)から最低13mmの点まで、カテゴリ3で75mm、カテゴリ6では6mmの点まで維持することが望ましいとされています。

●外被および心線への傷つけ

外被や心線を傷つけないようにします。とくに、外被を除去するとき、注意が必要です。ジャケットストリッパの外被噛みこみ圧を調整していないと、心線を傷つけてしまう場合があります。外被除去後に、心線の導体が露出していないかどうか確認を行ってください。

●心線押し込み不足

心線はしっかりと奥まで押し込み、IDC(Insulation Displacement Connectors)接続が十分にされるよう成端してください。ジャック成端の場合、心線が手で簡単に抜けるようでは押し込みが足りません。また、プラグ成端の場合は、心線をプラグ先端までしっかりと押し込んでください。図のように心線が奥まで差し込まれず、心線溝に挿入されているだけの場合は、導通されていません(このことは、導通チェッカにより確認できます)。

●シールド処理

シールド処理は、定められた方法により適切

Column 技能五輪でのモジュラコネクタ成端

技能五輪大会でも、このモジュラコネクタの成端を行っています。いかに速く品質良く成端できるかを競う「モジュラコネクタ成端スピード競技」です。これは簡単に言うと30分間でモジュラコネクタとモジュラプラグを何個作れるか？という競技です。もちろん、ワイヤマップ(電気的な導通)が完全にとれていることは言うまでもありません。現在の大会の上位入賞者は、30分間でモジュラコネクタを60個、モジュラプラグを60個程度作ります。まさに神業です。このレベルにくると、ワイヤマップエラーを起こすこともほとんどありません。

一方、国際大会ではこの競技はあまり評判がよくありません。なぜなら、とくにヨーロッパ各国では、「工事者がモジュラプラグを作ることはほとんどない」という事情があるからです。パッチコードは工場で作成されたものを使うのが常識だからです。電気製品のプラグは普通、現場で作りませんよね？　それと一緒なのです。

に処理しなければなりません。不完全な処理はかえってノイズの影響を受けやすくなり、ショートなどの原因にもなります。

測定試験

LAN配線試験を行う測定試験機には、導通試験機、LANケーブルテスタの2種類があります。導通試験機は、電気的な導通のみを確認するための試験機で、簡易的に成端間違いがないかの試験をする場合や、配線長が短い場合などに有効です。

しかし、これまで述べてきたように現在のLAN配線は、保証された帯域でケーブルの性能を満足するかどうか、が重要なのです。「つながっている」だけでは、もはやネットワーク性能を満足させることはできないのです。それを調べるため、LANケーブルテスタ（**写真12**）を用います。LANケーブルテスタは、ツイストペアケーブルの配線システム性能を、規格で決められた周波数範囲と項目に関して測定する試験機です。

LANケーブルテスタを用いて行う試験項目のうち、とくに重要なものは次の5つです。

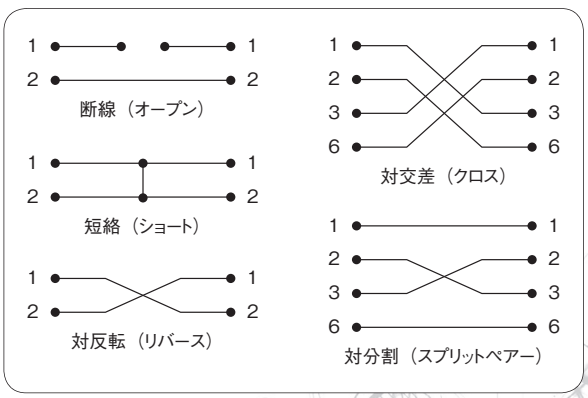
- **ワイヤマップ（Wire-map）**
  ワイヤマップ試験は、リンクまたはチャネルの電気的な導通試験です。ただし、**図4**のようにいくつかの項目がありますので注意が必要です。たとえば、スプリットペアーは通常の導通試験機では調べることができません。
- **配線長（Length）**
  配線システムの終端間の長さを測定します。
- **挿入損失（IL：Insertion Loss）**
  配線システムの両端にある送信器と受信器の間で生じる信号の減衰による損失を調べます。
- **反射減衰量（RL：Return Loss）**
  ケーブル導体のインピーダンスが変化することにより、反射するエネルギーを測定します。
- **近端漏話減衰量（NEXT：Near End Cross Talk）**
  配線システムの送信器から受信器に信号が伝播する間に、隣接対へ信号が漏れることを漏話と呼びます。このうち、送信側（近端側で）結合量を測定した値です。この値が悪いとネットワークに重大な影響を与えます。

◆ ◆ ◆

ここまで、つながるだけではなく、正しく通信を行うための配線技術について解説しました。配線にもさまざまな配慮や、高度な技術が必要であることがわかっていただけたのではないでしょうか。オフィスで配線することがあれば、気をつけてみてください。SD

▼写真12　LANケーブルテスタ（DTX-1800）

▼図4　ワイヤマップ項目



第3章

プロに聞く、光ファイバケーブルの知識と接続

古河電気工業株式会社 アクセスネットワーク部
内田 隆章　UCHIDA Takaaki

はじめに

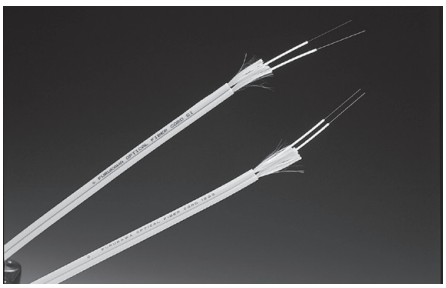
FTTH(Fiber To The Home)が広く普及してきており、すでに身近なところに光ファイバが使われています。この章では、FTTHやLANで使われている光ファイバの特性や接続について紹介します。

光ファイバの構造

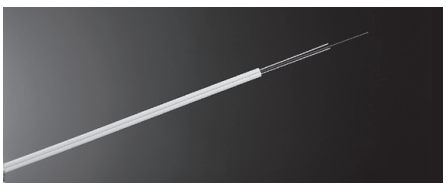
光ファイバは、屈折率の異なる二重のガラス層でできており、コアと呼ばれる中心部を光が全反射して、外側のクラッドから漏れないことで光を伝搬する髪の毛ほどの細さの繊維です。クラッド径125 μmの石英でできた光ファイバは、被覆径250 μmのUV心線、または900 μmのナイロン心線で覆われたものを光ファイバ心線と呼びます(**図1**)。心線のままですと非常に細く取り扱いが難しいので、さらに外径2mmや3mmのシースで覆った光コード(**写真1**)や、光ケーブルという形で一般的には使われています(**写真2**)。

また、通信局間を結ぶ光ケーブルの中には、心線を並べてテープ化したテープ心線が使われています。

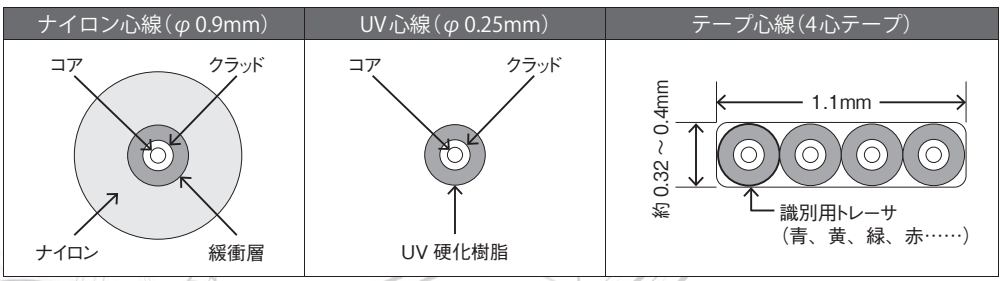
▼写真1　光ファイバコード



▼写真2　インドアケーブル



▼図1　光ファイバ心線の構造



光ファイバの種類

光ファイバは、光の伝搬方式の違いによって、大きく2種類に分けられます。光の伝搬モードが単一のシングルモードファイバ(SMファイバ)と、複数のモードのマルチモードファイバ(MMファイバ)です。

SMファイバはコア径が約10 μmで伝送特性も優れていますが(**表1**)、その分接続が難しく、MMファイバはコア径が50または60 μmと大きく、接続も比較的簡単ですが、伝送特性はSMファイバに比べると劣ります(**表2**)。

▼表1　ファイバ種別と波長別減衰量

	波長別最大ケーブル減衰量 dB/km				
ファイバ種	850nm	1300nm	1310nm	1383nm	1550nm
OS1	—	—	1.0	—	1.0
OS2	—	—	0.4	0.4	0.4
OM1	3.5	1.5	—	—	—
OM2	3.5	1.5	—	—	—
OM3	3.5	1.5	—	—	—
OM4	3.5	1.5	—	—	—

(※JISX5150より)

▼表2　マルチモード光ファイバケーブルの帯域

		最小モード帯域MHz・km		
		全モード励振帯域		限定モード励振帯域
波長		850nm	1300nm	850nm
光ファイバ種別	コア径(μm)			
OM1	50または62.5	200	500	規定なし
OM2	50または62.5	500	500	規定なし
OM3	50	1500	500	2000
OM4	50	3500	500	4700

(※JISX5150より)

光ファイバの接続方法

光ファイバを接続するには、光がきちんと伝搬するように光の通り道である、光ファイバのコア同士をつなぎ合せる必要があります。そこで現在では、次の3つの方法によって接続が行われています。

・融着接続
・メカニカルスプライス
・コネクタ接続

融着接続は、融着接続機と呼ばれる専用の装置を使って、光ファイバの先端を溶かして光ファイバ同士を接続する方法で、接続損失も小さく、接続部の機械的強度も大きく、信頼性の高い接続方法です。高価な融着接続機を必要としますが、ほぼ自動で接続できます(**写真4**)。

融着接続機は、カメラを搭載していて光ファイバの状態を検査し、2本の電極間で気中放電を飛ばして石英ガラスを溶かして光ファイバの接続を行います。融着接続後は、搭載している加熱器で、補強熱スリーブを収縮するこ

Column　最近の動向

光ファイバの弱点として曲げに弱いという点があります。小さく丸く巻いてしまうと、光が全反射できずにクラッドに漏れて損失が発生してしまいます。しかしながら最近は直径30mm以下のループ状に巻いても損失が発生しないような、曲げに強い光ファイバも使われ始めています(**写真3**)。

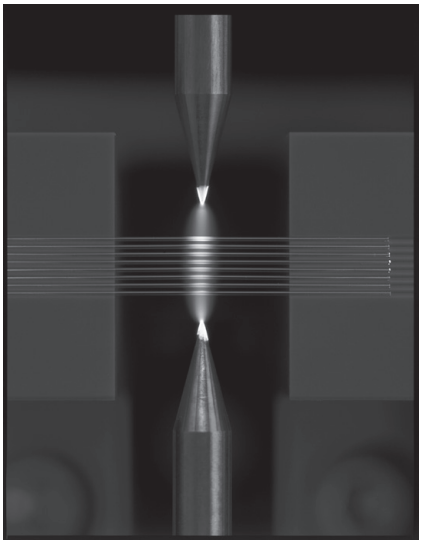
▼写真3　曲げに強いファイバ

とで、接続部を保護します(**写真5**)。

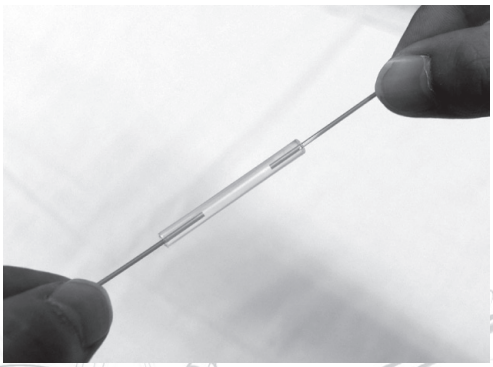
メカニカルスプライスは、精密なV溝を用いて光ファイバのクラッドをつなぎ合せ、突き合せた光ファイバの端面を屈折率整合剤によって反射を抑えて接続します(**写真6**)。専用の工具が必要ですが、小型・軽量・電源が不要ですので、宅内や架空での接続にも広く使われています。メカニカルスプライスは心線同士を接続しますが、メカニカルスプライスの技術を利用した、現場付け簡単コネクタというコネクタ組立方法もあります(**写真7**)。

コネクタ接続は、あらかじめ光ファイバの先端に取り付けられたコネクタ同士のかん合で簡単に接続ができるものです。規格に応じて、さまざまなコネクタ形状がありますが、誰でも簡単に着脱可能なメリットがありますが、接続損失は比較的大きい傾向があります。また、コネクタの大きさだけ接続点が大きくなりますので広い収納スペースを必要とします。

▼写真4　8本の光ファイバを放電で一括融着接続



▼写真5　融着接続後の接続部補強状態



光ファイバの接続手順

では、具体的な光ファイバの接続手順と必要な工具を紹介します。前述のように光ファイバは被覆で保護されていますので、専用の工具を使って光ファイバの前処理をしてから接続をします。

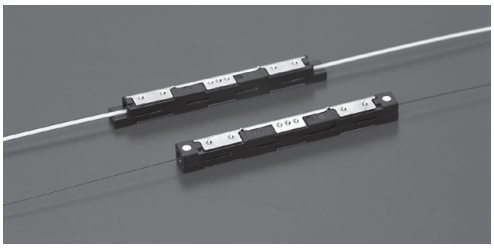
①被覆除去

はじめに心線の被覆を剥いでクラッドをむき出しにします。被覆を剥いだあとは被覆屑が残っていますのでアルコールを湿らせたワイプ紙などで拭き取ります。

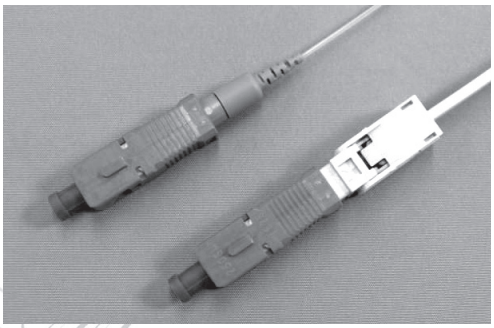
②切断

次に光ファイバを切断します。光ファイバを

▼写真6　メカニカルスプライス接続



▼写真7　現場付け簡単コネクタ



接続する際には、光ファイバの端面が鏡面のように平滑になっていないと、接続点での損失が大きくなりますので、専用の光ファイバカッタで切断します。最近の光ファイバカッタは、レバーを一度押すだけで簡単に平滑な切断面が得られ(**写真8、9**)、かつ切断屑が屑箱に自動回収するような機能を備えており、誰でも簡単に切断できるようになっています。

③接続

融着接続の場合は、融着接続機(**写真10**)に前処理した光ファイバをセットします。あとはスタートボタンを押せば、融着接続機がファイバの検査から放電による接続、接続後の検査まで自動で行います。

融着接続機はFTTH工事用に小型・軽量化が図られ、バッテリ駆動も一般的になっていますので、電柱の上の架空工事から宅内まで幅広く使われています。

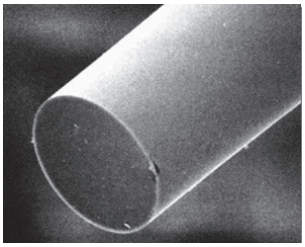
メカニカルスプライスの場合は、組立工具に素子をセットし、左右から前処理した光ファイバを挿入させ、光ファイバ同士を当てて素子内で光ファイバを機械的に把持(はじ)します。

光ファイバの接続作業

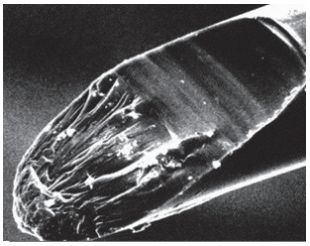
街中でバケット車が止まっていて電柱の上で何か作業をしていたら、光ファイバの接続作業や保守をしていることが多くあります(**写真11**)。また地下配線ケーブルの場合はマンホールの中で接続作業など、実際の光ファイバの接続作業は環境的にはけっして良くないところが一般的です。一方で光ファイバは非常に細く繊細ですので、ほこりやゴミなどがあると、きちんと接続できなかったり、接続損失が大きくなってしまうので注意が必要です。

そのため、最近の融着接続機は、防滴・防塵性能や耐落下・衝撃性を特徴にしたものもあります。省電力化や自動化が進み、誰でも簡単・確実に接続できるように進歩しています(**写真12**)。SD

▼写真8　カッタによる切断面



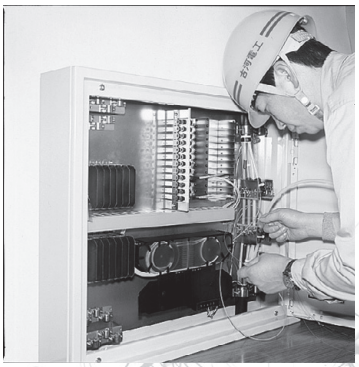
▼写真9　折れた光ファイバの断面例



▼写真10　古河電工製 S123M4 融着接続機

▼写真11　架空での融着接続作業工事例

▼写真12　局内成端箱での接続作業例



テクノロジの勃興を俯瞰する

ハードディスクがなくなる日は近い？

SSDストレージ爆発的普及の理由

岩田郁雄　*IWATA Kunio*
TwitterID：@STRG_GURU

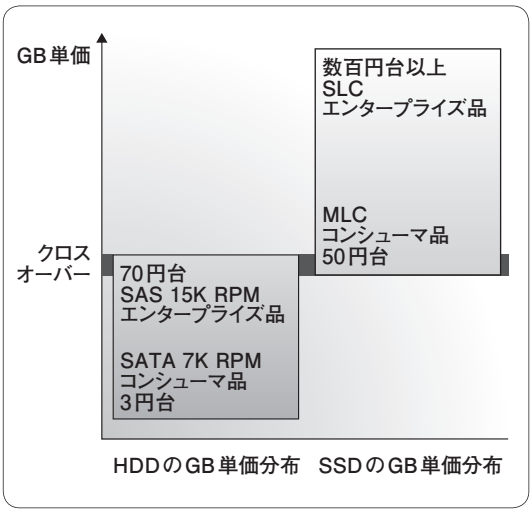
HDD対SSD宿命の対決

筆者がハードディスクドライブ（以下HDD）とデータストレージの仕事に関わってから、早いもので20年以上が経ってしまいました。当時の勤務先で初めてHDDの開発製造部門に異動になったとき、周囲からは「HDDなんてすぐにMOか半導体に置き替っちゃうよ」と無責任なコメントで送り出されたのですが、それから数十年、粘り腰を見せたHDDは驚異的な記録密度の伸びで他の競合記録媒体を圧倒し、いまだにコンピュータ機器の二次記憶装置としての王座を守っており、映像機器などの用途まで進出することで年間およそ6億台の出荷を見るまでになりました。一方でNAND Flashを記憶媒体とするSSDも急速な勢いでHDDを追撃していますが、出荷台数ではHDDの5%程度と大きく水をあけられています。

HDDとSSDを取り巻く状況

しかし、この数年で風向きが変わってきたのも事実です。1つの背景にNAND Flashデバイスの容量単価の下落があります。HDDの容量単価と比較すると、一般的にはまだNAND Flashの容量単価が上回っていますが、昨年になって、一部に逆転現象が見られるようになりました（**図1**）。インターネットで入手したストリートプライスの最低価格（2012年12月現在）で比較すると、HDDのGB単価は、SATAの大容量品（おもに5,400～7,200回転）が3円台と格安ですが、サーバに使用されるエンタープライズ向け製品（おもに15,000回転）では70円台以上に跳ね上がります。一方で、SSDのほうはMLCのコンシューマ品で50円台まで落ちてきています。これによって、HDDのGB単価レンジとSSDのレ

▼図1　HDDとSSDのストリート価格比較（筆者の独自調査結果）



ンジが交差するクロスオーバー領域が現れるに至りました。簡単に言えば「高いHDDよりも安いSSDのほうが容量単価が安くなった」と結論付けることができます。

もちろん、エンタープライズ向けHDDとコンシューマ向けのMLC搭載SSDと、同列で比較すべきではありませんが、これはこれで1つの大きな時代の転換点ではないかと思います。MLCのNAND Flashは消去回数の上限値がSLCよりも少ないなどの制限はありますが、SSD内部でのウェアレベリング方式や寿命管理のしくみの改善などで、一定の制限はあるものの、耐久性に関しても十分な実用性を備えるようになってきています。

もう1つの背景として、HDDのアクセス速度に関する制約があります。半導体の実装密度がムーアの法則に従って、年率約58%で集積化が進むと同時に高速化し、一方HDDの面記録密度も年率40～100%で集積化が進んで行ったのに対し、HDDのランダムアクセス性能は過去ほとんど改善していません。**表1**におおよそ過去10年間のテクノロジの比較を行いました。CPUに関しては、2000年当時のものと比較して2011のものは単純なクロック×コア数の比較でも6.4倍。ベンチマークの数値では28.7倍（メモリの速度やバス幅も影響している）に改善しています。またメモリモジュール（DIMM）の速度では、この間に4倍になっています。

一方でHDDに関しては、面記録密度の高度化に伴って媒体からデータを読み出す速度（最大メディアデータ速度）は確実に改善していますが、ランダムアクセス性能に寄与する平均シーク時間に関してはほとんど改善が見られません。またランダムアクセスに影響するもう1つの要因であるディスクの回転数も15,000RPMで頭打ちになっています。平均シーク時間と平均レーテンシ時間（1回転にかかる時間の半分）から理論上のIOPS（Input Output Per Second）を計算すると、過去10年間に3%しか改善していないことになります。これはCPUやメモリの速度改善に対してHDDの速度改善が取り残されているということを意味します。また、HDDメーカー関係者の話を聞いても、HDD回転数の高速化やシーク時間の短縮などは、今後の開発ロードマップには入っていない模様です。

仮想化環境の普及と要求されるストレージ性能とのギャップ

一方でサーバの使われ方は、仮想化OSの普及によって過去10年間に大きく変化しました。現在物理サーバの出荷台数の伸びが年間で2～3%なのに対して、仮想サーバ（仮想化OS上のVM）の数は25%程度で増加しています。これだけサーバの仮想化が進んでいるということになります。仮想化環境では物理サーバ上に複数の仮想サーバが動作するため、ストレージに対するアクセス負荷は増大します。とくに問題となるのがランダムアクセスの増加です。仮想化環境下では共有ストレージの性能がボトルネックになり、思ったような性能が出せない、あるいは思ったようにサーバの集約化が進まないといった事態も頻繁に発生しています。

▼表1　2000年時点と2011年時点のテクノロジ比較

比較対象		2000年	2011年	変化
CPU	代表的な製品	Pentium 4 1.5GHz 1 Core	Westmere 5620 2.4GHz 4 Core	-
	クロック×コア数比較	1	6.4	6.4倍
	Passmark bench数値	174	4994	28.7倍
DIMM	代表的な製品	DDR PC-2100	DDR3 PC-8500	4倍
HDD	代表的な製品	Cheetah 15K.2	Cheetah 15K.7	-
	最大メディアデータ速度	891Mbps	2370Mbps	2.7倍
	理論上のIOPS	179	185	1.03倍

SSDストレージ爆発的普及の理由

たとえば、今まで15,000RPMのSAS HDD1台(おおむね180IOPS程度)が搭載されていた1Uサーバを仮想化環境に移行すると仮定します。物理サーバ1台にハイパーバイザ(仮想化OS)を導入し、そのうえに仮想サーバ(VM)を20VM定義したとします。仮想化環境に移行後も1Uサーバのときと同じ性能を期待するのであれば、ストレージに関しては最低でも180×20 = 3,600IOPSが必要になります。実は、この値はストレージにとってはかなり厳しい性能要求です。リードアクセスは良いとしても、ランダムライトに関して性能を出すのはとくにRAID構成を取った場合には容易ではありません。

HDDを使ったストレージシステムでランダム性能を向上させようとすると、

①HDDの台数を増やして並列にアクセスさせる
②HDD1台の容量を制限して(外周部だけを)使用することでヘッドのシーク距離を減らす
③RAID5やRAID6ではなくパリティ計算の不要なRAID0+1を使用する

などの手法がありますが、ストレージシステムの規模は肥大化し、容量単価ひいては消費電力や設置面積などを考慮したTCO(Total Cost of Operation)もかなり高いものについてしまいます。

もう1つの解決策として、HDDの代わりに、スピードの速いSSDに差し替えて使用するというアプローチもあります。実際にHDDやSSDではどの程度の性能が出せるのでしょうか？**表2**に最新のエンタープライズ向け(24時間×365日稼働を前提として設計されているもの)製品の理論上のランダムIOPS(READの場合)をまとめてみました。ランダムIOPSとは小ブロックのランダムI/Oを1秒間に何回実行できるかを意味します。ライトの場合にはシーク時間が余計にかかるので、この値よりも小さくなります(表1のHDDは3.5インチ、表2のHDDは2.5インチの製品なのでIOPS値は異なります)。

機械的可動部分の動作に制約されるHDDに比較してSSDのIOPSが圧倒的に高いのがわかるでしょう。

一方でランダム性能を制限する要素として、RAIDにおけるライト・ペナルティも考慮する必要があります。HDD単体で使用する場合は小ブロックの書き込みはHDDに対する1回の書き込みで済みますが、RAID構成で使用した場合、パリティデータの更新に際し旧データや旧パリティを読み出して新たなパリティを計算する処理が入りますので、複数回のアクセスに増えてしまいます。小ブロックの書き込みが何回のアクセスに増幅されるかをライト・ペナルティと呼びます。代表的なRAIDレベルとライト・ペナルティの関係を**表3**にまとめました。

100,000IOPSのSSDをRAID6に組み込んだ場合、ライト・ペナルティにより1台当たりのライト時のIOPSは計算上16,667に減ってしまいます。それでもHDDに比較すると驚異的な性能ではあります。

ただし、上記の計算上の性能はHDDやSSDを束ねてRAID化するストレージコントローラ側にボトルネックがないという仮定のものです。実際にはボトルネックがありますので、HDDをSSDに置き換えただけでは驚異的な性能が出ないのが実際です。また、SSDの100,000IOPSという性能値も理想的な状態で計測したもので、

▼表2　エンタープライズ向け製品の理論上のIOPS(READ)

HDD/SSD種別	理論上のIOPS
SATA　7,200RPM	80
SAS　10,000RPM	150
SAS　15,000RPM	210
SSD (SLC)	100,000

▼表3　RAIDレベルとライト・ペナルティ

RAIDレベル	ライト・ペナルティ
RAID 0	1
RAID 1	2
RAID 5	4
RAID 6	6

HDDと同じようなブロックサイズや論理アドレスの使い方で読み書きしてもそれほどの性能が出せなかったりする場合もあります。加えて、HDDと遜色ない使用方法に耐えられるのは通常はSSDの中でもSLCのFlashを使用した高級品になりますので、GB単価を考慮すると、ストレージ全体をSSD化するのは難しく、たとえば総容量の5%～10%程度をSSD、残りをHDDで構成し、高速なSSDをキャッシュとしてあるいはティアリング(使用頻度の高いデータを高速なSSD上に移動させる)としての使用方法が一般的でした。

サーバを加速するFlashストレージの登場

ここまで述べてきた状況をふまえて、ストレージの性能ボトルネックを解消すべく、Flashを応用したストレージ技術が開発され市場に投入されてきました。現状エンタープライズ系で選べるFlashを使用したストレージソリューションとして次のものが存在します。

- 既存のストレージの一部をSSDにしてキャッシングやティアリングを行う方式
- PCIeスロットに装着するFlashデバイス(SSC)
- オールFlashのストレージ・アプライアンス

まず、既存のストレージの一部をSSDにしてキャッシングやティアリングを行う方式ですが、これはストレージベンダ各社がミッドレンジ以上の製品で、すでに提供しているソリューションです。ストレージコントローラ配下にHDD/SSDを収納するJBOD(Just a Bunch Of Disks：複数のHDDをまとめる技術)を接続してシステムを構成しますが、データの種類(アクセス頻度やブロックサイズ)に応じて、超高速なSSD、高速なエンタープライズHDD、そして低速だが容量が大きいニアラインHDDで作られたボリュームにデータを保管します。すべての容量をSSDで構成すれば非常に高速なストレージが実現できますが、容量単価の問題でそこまではいかず、多くの場合では総容量の5%～10%程度をSSDで実現し、ストレージコントローラの機能で、SSDの領域をキャッシュとして、またはティアリング(アクセス頻度の高いデータを高速なデバイスに自動的に移行する)として使用します。SSDボリュームをそれほど大きく取らないケースでは、ストレージコントローラ内部のPCIeバスに後述のPCIeスロット装着型のFlashデバイスを取り付けて使用する場合もあります。

この方式の利点は既存のストレージ環境を変更せずに、システムの性能を向上できるという点にあります。一方で、超高速なSSDボリュームへのキャッシュやティアリングと言っても、要求されたデータが常に必ずそこに存在するわけではなく、キャッシュミスが発生してHDDにアクセスに行かなければならないケースもあり、その場合にはデータのアクセス時間が数十ミリ秒程度かかってしまいます。また、キャッシュとして使われるためにSSDへのアクセスが集中し、Flashの消耗が進んだり、キャッシングやティアリングされるデータブロックの単位が意外と大きく、データの入れ替えにオーバーヘッドがかかってしまうといった問題も指摘されています。

次にPCIeスロットに装着するタイプのFlashデバイスの紹介をします。ストレージの業界団体であるSNIA(Storage Networking Industry Association)ではこのカテゴリの製品を、SSC(Solid State Card)と称してSSDとは区別しています。SSDの場合、基本的な動作はHDDのエミュレーションであり、インターフェースやコマンドセット、そしてブロックサイズなどHDDと同じものを使用します。そのため、HDDと差し替えて使用できますが、反面、性能的な限界も出てきてしまいます。現状での一番早いインターフェースはSASの6Gbps(600MB/Sec(片方向))ですが、SSDの内部はもっと早いものを設計することはできますし、アクセスする

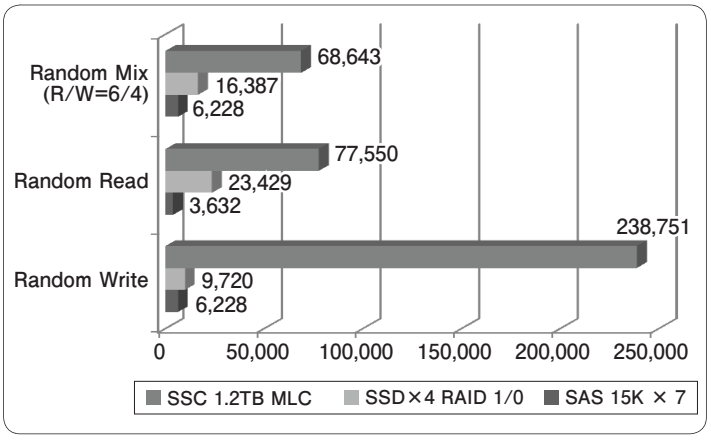
ためのブロックサイズやコマンドセットなどもHDDのエミュレーションでは本来の性能を出し切ることはできません。

SSCではSSDで行っていたHDDとの互換性を捨ててしまい、PCIeのバンド幅を最大限活かせるように内部設計も最適化しています。したがって、性能的にもSSDを単体で使用したものに比較して飛躍的な高性能を実現できるのです。例としてA社の製品では、PCIeのバンド幅は最大でx16(8GByte/Sec)、容量も最大で10TB(MLC)のモデルが用意されています。

SSCはサーバ1台を高速化するには最適なソリューションで、高速なドライブとしての使用はもちろん、別売りのソフトウェアを併用して、キャッシュとして使用することもできます。非常に大規模なユーザセットを抱えるネットワークサービス(たとえばFacebook等)ではSSCを使用した性能向上を図っています。

実際にSSCがどの程度の性能を発揮するのか、当方で測定した性能比較データを**図2**に示します。これは4KBのブロックサイズでSSC(1.2TB MLC)、SSD × 4(RAID 1/0)およびSAS HDD(15K RPM)×7のランダム性能をIOMETERで測定したものです。SSCはとくにランダムライトにおいて卓越した性能を示しているのがわかります。

▼図2　HDD, SSD, SSCのランダム性能IOPS比較(4Kブロック)
(http://cn.teldevice.co.jp/product/detail/iodrive/featureより)



一方で、すべてFlashで実装したストレージ・アプライアンス製品も普及を始めました。これは専用の筐体の中に、新規に設計したFlashデバイスを実装して、高密度化高速化を図ったもので、例としてB社の製品が挙げられます。同社の最新シリーズは、最大物理容量で32TB、最高性能で100万IOPSを謳(うた)っている製品です。同社ではVIMMと呼ばれる自社開発のFlashモジュールをさらに独自技術のスイッチベースの接続機構によって相互接続し、高いIOPSを引き出しています。またホストへの接続として、FC(Fibre Channel)、InfiniBand、iSCSI、PCIeを用意し、既存のストレージサブシステムと同様に複数のサーバへの接続をサポートしています。

新世代Flashストレージ製品の登場

SSCは1台のサーバを高速化するには最適のソリューションなのですが、大規模な仮想化環境やVDI(デスクトップ仮想化)環境などで、物理サーバを複数立ててストレージを共有する必要がある場合には多少使いづらいものになってきます。また、サーバ内部に装着する関係から、冗長性を保つためにはSSCを搭載したサーバ間でミラーリングを行う必要があります。一方で、オールFlashのストレージ・アプライアンスは共有ストレージとしての使用には向くものの、記憶容量単価が非常に高いという難点があります。

こういった問題点を先進的なテクノロジの導入によって解決した、新世代のFlashストレージが開発されて、北米などの一部地域で出荷が始まっています。代表的な製品としてPure Storage社のFA-300シリーズが挙げられます。今回、実機を評価する

▼写真1　Pure Storage FA-300シリーズ

機会に恵まれたので、この製品の内容について紹介したいと思います。

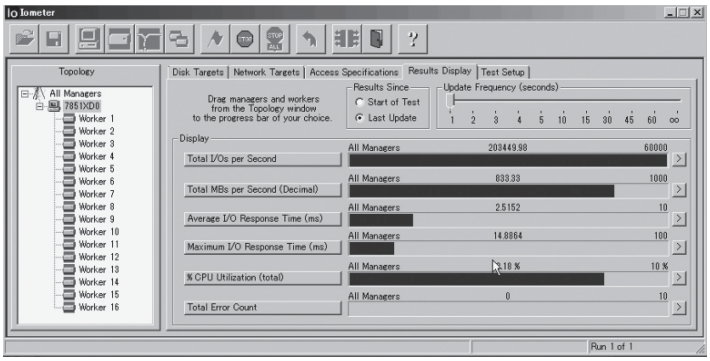
FA-300シリーズ(**写真1**)はオールFlashのストレージサブシステムで、記憶媒体としてはMLCのSSDを用いながら、インラインでの圧縮と重複排除によって、実効的な記憶容量を5倍以上に拡大し、容量単価をHDD(エンタープライズ向けのSASドライブ)並みに近づけた製品です。写真は現状での最大構成で、コントローラ2台のHA構成の下にストレージシェルフが2台、SASインターフェース(6Gbps×4レーン)で接続されています。ハードウェアはすべて実績のある既製品を採用し、SPOF(Single Point of Failure)がないよう冗長化が図られています。

ストレージシェルフあたり物理容量5.5TBのSSDが搭載されており、前述の圧縮および重複排除により、実効の記憶容量は最低でも5倍程度は期待できるので、RAIDの冗長性を考慮するとシェルフあたり20TB、最大構成で40TB程度の実効容量を見込めます。ホストインターフェースはFC8Gか10Gのiscsiが選択可能で、最大で1コントローラあたり6本のインターフェースを装備できます。

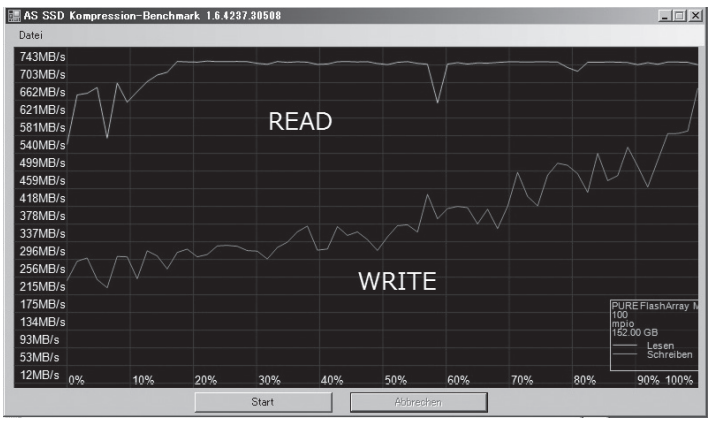
肝心の性能ですが、Pure Storage社のカタログ上のスペックでは、4Kブロックサイズのランダムライトが100,000IOPS、4Kブロックサイズのランダムリード／ライトミックス(80：20)で200,000IOPSとなっています。実際に当方で測定したところ、理想的な条件で測定すれば前述の性能を出せることが実測されました(**図3**)。これは非常に圧縮効率の高いデータを使用して測定した場合の話で、実際の性能はデータの種類によって異なります。

図4はAS_SSDベンチマークというツールで性能を測定した結果で、データの圧縮率によって性能が変化する様子を現しています。リードの場合は圧縮率が非常に小さい場合を除いてほ

▼図3　IOMETERでの性能測定結果
［4K Block Random Read/Write（80:20）］



▼図4：データ圧縮率による性能の変化



ぼ一定の性能を見せているのに対してライトの場合は、圧縮率と性能に相関が見いだせます。先ほどの最大性能は、このグラフで言うと一番右側、すなわち圧縮が非常に効果的な場合のIOPS値になります。

同社の製品はRAID-3Dという独自の方式で高度なデータ保護のしくみを提供しています。これはRAID6をさらに強化したもので、RAID6のレイヤの下にさらに2段階のデータ保護のしくみを導入し、全体で三重の保護を適用しています。またFlash-Careという独自開発の処理方式の採用により、データの書き込み単位をSSD内部でのFlashへの書き込み単位に合わせることで、SSD内での無駄な書き込みや消去を排除し、MLCのSSDを記録媒体に使用しながらも、高速性と長寿命性を達成しています。また、データをまとめてからSSDに書き込むことで、前述のライト・ペナルティの発生を防ぎ、性能の最適化を図っています。

操作性は良好で、洗練されたWEB GUIか古典的なCLIのいずれからでも同じ操作が行えます。ボリュームの作成もボリューム名を決めてプロビジョニングサイズ(ホストに見せるボリュームサイズ)を決めたら、接続先のホストを定義して接続するだけの作業でした。SSDは全体が1つのストレージプールとして管理されていますので、SSDの物理的な位置や本数などの情報はまったく気にかけずにボリューム作成と管理が行えました。障害の通知機能や耐障害性なども、従来のストレージ製品と比較して遜色ないレベルでした。

この製品は2012年6月から量産出荷が始まった比較的新しい製品ですが、北米ではすでに100セットを超える台数が本番環境で稼働しています。メーカーでは、とくに高いランダム性能を要求される、サーバの仮想化(VM)、デスクトップの仮想化(VDI)、リレーショナル・データベースと解析の分野を中心に販売しています。実際の導入例として米国でのユーザ事例が公開されていますが、eMeter(Siemensの一部門)で仮想サーバおよびデータベース用として、yodle社でデータベースの解析用として、STIという教育機関で2,000ユーザを超えるVDIサービス用として、TripPak(Xeroxの一部門)でOLTPおよび仮想サーバ用としてなどの例が紹介されています。圧縮と重複排除によるデータの圧縮率としては、データベースの場合で4～5分の1、VMやVDIの場合で5～10分の1という実績が報告されています。また、メーカーによるとVDIの場合で20～30分の1に圧縮できた事例もあるそうです。

HDDが終わる日はいつか?

HDDが半導体記憶装置に完全に置き換わる日はいつか来るのでしょうが、もう少し年月を要すると思われます。その中で、この数年はとくにNAND Flashの適用エリアが拡大してきたのを強く感じます。一方で、現在、半導体メーカー各社はNAND Flashの記録密度向上に取り組むとともに、次世代記憶デバイスの開発にも精力を注いでいます。有力な候補として、磁気抵抗メモリ(MRAM)、抵抗変化メモリ(ReRAM)、相変化メモリ(PRAM)などが挙げられます。これらのうちのどれかが立ち上ってくると、たとえばメインメモリがすべて不揮発になるなど、コンピュータのアーキテクチャも根本的に変わってくるのかもしれません。今後数年間の変化から目が離せません。SD

■参考

Pure Storage社Webページ
http://www.purestorage.com/

著者紹介

岩田郁雄

日本IBMをはじめ、複数のメーカーにて長年HDDおよびストレージシステムの開発と販売に従事。HDD/SSDの応用技術とストレージ一般が得意分野。現在は海外製IT機器やソフトウェアの評価と国内への導入をおもな業務としている。IEEE会員。

OSとネットワーク、
IT環境を支えるエンジニアの総合誌

Software Design

毎月18日発売

5% OFF& 送料無料

年間定期購読のご案内

富士山マガジンサービス版

年間購読なら割引料金で購読できます!

全国どこでも直接お届けしています!

1年購読（12回）

14,580円（税込み，送料無料）1冊あたり1,215円（5%割引）

- インターネットから最新号、バックナンバーも1冊からお求めいただけます！
- 紙版のほかにデジタル版もご購入いただけます！
  デジタル版はPCのほかにiPad/iPhoneにも対応し、購入するとどちらでも追加料金を払うことなく読むことができます。

［月額払い］スタートしました！

デジタル版Software Design

Webで購入

家でも外出先でも

※ご利用に際しては、／~＼Fujisan.co.jp（http://www.fujisan.co.jp/）に記載の利用規約に準じます。

お申込み方法

1 >> ／~＼Fujisan.co.jp クイックアクセス
http://www.fujisan.co.jp/sd/

2 >> 定期購読受付専用ダイヤル
0120-223-223（年中無休、24時間対応）

全業務をクラウド化してISMS認証を取得

AWS専業のインテグレータであるサーバーワークス社は、2012年12月にISMS（情報セキュリティマネジメントシステム）認証を取得しました。そのアプローチをブログで公表したところ、先進的な取り組みが大きな話題を呼びました。本稿では、ブログでは書ききれなかったISMS認証取得の背景、実際の運用について詳解し、その舞台裏を公開します。

サーバーワークスの取り組みとは？

社内LAN撲滅運動！

株式会社サーバーワークス　代表取締役
大石 良　*OOISHI Ryo*
Twitter @ooishi

AWS専業インテグレータとして

みなさん、こんにちは。サーバーワークスの大石です。当社はAWS専業のインテグレータとして、Cloudworks（cloudworks.jp）という日本語のEC2/EBSコンソールと、AWS導入・運用支援サービスを提供している企業です。

当社はもともと大学・専門学校向けの「合否照会」サービスの運用を行っていました。このサービスは「合格発表日の午前10時から10時15分までだけのためにサーバが200台くらい必要」という、極端なスパイクトラフィックが発生するものでした。そのため当社は投資効率を高めるため、その対応にずっと苦労してきました。そんな中で、社内のエンジニアがEC2に目をつけて、2007年に試験的に利用しはじめることになります。筆者自身、その使い勝手のよさ、コスト効率の良さに衝撃を受け、2008年1月に「社内サーバ購入禁止令」を発布しました。「これから先、社内ではサーバを購入しない。サーバが必要だったらすべてAWSを使うべし」と決めたのです。

そして2009年からは本格的に外販を開始しました。CloudworksというEC2/EBSを日本語で簡単に操作できるためのWebサービスも提供し、新規のSI案件はすべてAWSで実装するという方針を打ち出しました。

意外な顧客層

当初こそ営業活動には苦心したものの、2011年の後半からは、本格的に企業によるAWSの導入が進みはじめました。とくに2011年8月にAmazon VPCが東京リージョンで利用できるようになってからというもの、当社がVPNのオペレーションをすべて引き受ける「VPCスターターパック」を提供しはじめた効果もあり、これまで考えられなかったような機密情報を含むシステムをAWS上に展開するプロジェクトが増えてきたのです。とくに大企業、上場企業によるAWS利用が進むにつれ、私たちに求められるものもセキュリティや社内のオペレーションなど、「エンタープライズ要求への適合」へと変わっていきました。この流れに呼応するため、私たちはISMS認証を取得し、エンタープライズ案件の要求に応えられるような体制構築を目指すことにしました。

まずはポリシーの確立！

ISMS認証取得に先立ち、一番はじめに行うこと、そしてもっとも大切なことが「ポリシーの定義」です。ポリシーに「私たちが、クラウドに対するニーズへの対応と高いセキュリティの両

立を目指すためにISMSを確立すること」を明示するとともに、ISMS事務局メンバに次の3点を依頼しました。

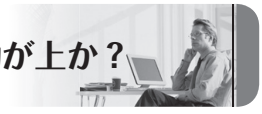
①ISMSの構築によって生産性や業務の効率の低下が最小限になるよう、セキュリティと効率を両立させること
②他社サービスを利用する場合、自社で運用するケースと比べて、どちらが高いセキュリティを維持する実行力があるか、客観的に比較すること
③BYOD(Bring Your Own Device)に対応したシステムを構築すること

セキュリティと効率の両立

よく「ISMS認証を取得したので、社外にPCが持ち出せなくなった」など業務効率を無視したオペレーションができてしまうケースを聞くことがあります。筆者は、これを経営者の問題だと考えます。ISMS認証の取得が目的化してしまえば、業務効率や現場の意見など聞かずにオペレーションを設計してしまうことは半ば当然です。経営が、はっきり「両立すべし」と言わなければこの問題は避けられません。筆者たちは「よりたくさんの方にクラウド、とくにAWSの魅力を知ってもらう」ことを目的としてISMS認証の取得に踏み切りました。

インフラやサービスの用意にかかる時間が驚くほど短縮されたのに、人間の部分が改善されないのではクラウドのメリットが半減してしまいます。筆者たちは、クラウドを必要とするお客様がそのメリットを最大限に引き出せるよう、そして人間がプロジェクト推進の妨げにならないよう、業務効率を落とさずにセキュリティを高める方法を徹底して追求することにしました。

どちらの能力が上か？



よくクラウドに反対される方から「外部に機密情報を預けるのは心配だ」といった意見を耳にします。ですが、ほとんどのケースで「自社で運用し続けることのリスク」との比較は行っていないように見受けられます。「クラウドが危険」と言えるのは、それが「自分たちが社内で維持した場合よりも危険」な場合に限定されるはずです。

筆者たちは、会社全体で高いセキュリティレベルを維持するために、「自社と、どちらがより高度なセキュリティレベルを継続的に維持する能力があるか？」という視点で、合理的に各サービスの利用可否を決めることにしたのです。

BYOD(Bring Your Own Device)

最後がBYODです。ISMS認証取得を開始するまで、当社ではiPhoneやAndroidを勝手に持ち込む「野良BYOD」が蔓延していました。ですが、筆者たちはこれを安易に「禁止」するのではなく、むしろ「BYODによって社員も会社もハッピーに！」という道を模索することにしたのです。

そもそも、ITに携わるすべての人にとって、自分のコンピュータは「自分の仕事道具」です。料理人が包丁をお店から借りたり、野球選手がグローブを球団から借りたりしないのと一緒で、コンピュータは自分の肌に合うもの、手になじむものが一番だと考えています。そうしたBYODの良さと会社のセキュリティをどう両立させるのか、その点をよく考慮してもらえるよう、事前にお願いしておいたのです。

具体的なサービスとこうしたポリシーに従って、ISMS事務局の方でひとつひとつ社内で必要とされる業務と、それを実現するためのシステム(または外部のサービス)、そしてそれらを統合するしくみなどについて検討が加えられていきました。それが**表1**です。

当社で使用しているサービスについて

Box(ファイル共有)

当初はDropbox for Teamsを検討していまし

たが、情報セキュリティを保つうえで必要と考えられる「権限の割り当て」が利用者個人に委ねられてしまい、会社としてセキュリティマネジメントが実現できないと判断して諦めました。Boxは同期の性能こそDropboxに劣りますが、企業が利用するために必要十分な管理機能が備わっており、GBを超えるサイズのファイルを日常的にやりとりする環境でなければ、社内ファイルサーバの代替として十分に機能すると考えます。利用当初は日本語に起因するバグに悩まされましたが、ほとんど当社がバグ取りしたと思います(笑)。

▼表1　サービスとポリシーの検討

分類	業務	サービス	備考
開発	チケット管理	Redmine(AWS)	
	ソースコード管理	Subversion(AWS)	
		GitHub	有料版でプライベートリポジトリを利用
	継続ビルド	Jenkins(AWS)	
	開発環境	EC2(AWS)	
営業	案件管理	Salesforce	SalesCloudを利用
共通	ファイル管理	Box	Enterprise
	出退勤管理	Force.com+チームスピリット	勤怠管理および交通費精算
	コミュニケーション	Google Apps	メール(社外との連絡用)、カレンダー
		Yammer	社内メールは禁止。社内のコミュニケーションはすべてYammerで管理
管理	ID管理	ActiveDirectory	
	ID統合	OneLogin	AD上のIDをシンクロして、シングルサインオン
	MDM	サイバネット	

OneLogin(シングルサインオン)

これだけ複数のサービスを利用すると、どうしてもID・パスワード管理がたいへんになってきます。とくに当社ではAWSアカウントだけで数百という数があり、これらの管理を個人に委ねることは難しいという認識がありました。そこで、ActiveDirectoryとID連携ができるシングルサインオンのサービスであるOneLoginを利用し、パスワード管理を属人化せずに、安全性と利便性を両立させることにしました。

MDM(モバイル端末管理)

MDMも多くの種類がありますが、当社では業務の都合上Windows、Mac OS X、iOS、Androidの4種類に対応したものという条件で選定し、サイバネット社のMDMを利用しました。

BYOD(Bring Your Own Device)

このように「積極的に外部のサービスを組み合わせることでセキュリティレベルを高める」システムを作ってきましたが、当然コストの増大が社内でも問題になりました。これをカバーすべく、BYODと組み合わせることでコスト最適化を図ろうと考えました。

どのみち、社内への端末持ち込みを禁止しても、エンジニア軍団である当社であれば誰かがハックして抜け道を作ってしまいます。そんな不毛なイタチごっこに労力を費やすのではなく、BYODによってコスト削減と、社員のモチベーションアップを両立させようと考えたのです。

そこで筆者たちは、「BYODによって会社が得られるメリットを、賛同してくれた社員と共有する」ために、BYODによる報奨金を用意することにしました。PCであれば毎月3,000円、携帯(iOS、Android)は毎月2,000円という具合に、業務で利用することで会社が得られる金銭的なメリットを還元することにしたのです。

ただし、当然業務で使うことになりますので制限もあります。まず大前提は、当社が選定したMDMの管理下に入ってもらうこと。これは、端末をなくしたときに管理者がリモートでワイプしたり、グレーなP2Pソフトウェアなどがインストールできないということを意味します。最初は反対も予想していましたが、野良BYODを続けるよりも、会社のMDMを入れておいたほうが結果的に自分を守ることになるわけです

から、スムーズに導入できています。

将来的には、会社で準備するデスクトップPCなどは全廃し、すべてBYODへ移行していく方針です。

そして残った2台の社内サーバ

前述のようなポリシーに従ってセキュリティを高める運用を突き詰めた結果、社内に残るサーバがたった2台となりました。ActiveDirectory（以降AD）のドメインコントローラと、IP電話のために設置しているAsteriskサーバです。

もともとADは社内がWindows端末だけだった時代に導入したものです。端末がWindowsに限られる場合、ADは非常に強力な内部統制を実現できます。ですが、MacやiPhone、Androidなど社内の情報にアクセスするための端末が多様化している現在では、ドメインコントローラによるポリシーの定義は意味が希薄になりつつあります。そのため、私たちはADによるポリシーの配布は諦めて、ポリシー適用をMDMに任せることにしました。筆者たちが選択したMDMでは、Windows、Mac OS Xに加えてiOS、Android端末も一元管理できることから、高度なポリシー運用ができると判断しました。

もう1台のAsteriskサーバですが、こちらも外部サービスに移管の予定です。

せっかくBYODでiPhoneやAndroidを持ち込んでいるわけですから、そのまま電話として使ってもらいたいという発想です。社内に端末があるときは無線LAN経由で着信し、外出先では050番号で受発信できるようにする計画です。これによって、BYODで持ち込まれた端末でも通話料を厳密に個人負担と会社負担とで分けることができるようになり、さらに会社で電話機を購入する必要もなくなります。

そしてこの2台のクラウド化が完了すれば、当社からは社内サーバが一掃されることになります。こうなると、**もはやLANという概念がなくなるのです**（！）。

BYOC（Bring Your Own Chair）

余談ですが、最近では「椅子を持ち込みたいのでBYOC（Chair）もはじめてください」という話が出ています。職業がら腰を痛める人が多いのですが、個人に合う椅子は会社支給のものよりも持ち込んだほうが良い、という考えからです。まだ報奨金制度はできていませんが（笑）。

意識を変える、仕事の強みを作る

もともと社内LANという概念は「物理的にPC同士がつながっていた」時代に生まれたものです。物理的に端末同士つながっていて、さらに端末が移動することがない環境であれば、今でも社内LANというコンセプトはセキュリティレベルを維持するために便利な機構だと思います。ですが、現代に生きる私たちは業務にMacを使い、外出先ではiPhoneでメールを読み、カフェで業務日報を書くことが当たり前。むしろ「そうしたワークスタイルを実現できるかどうかが、技術者にとって会社選定の主要な理由の1つ」にもなりつつある現在、優秀な社員を獲得するために、社員をオフィスという名の牢屋に閉じ込めるような働き方ではなく、先進的かつ人間的な働き方を実現できる環境を作ることが不可欠であると考えています。

その目的を達成するためには、社内でもリモートでも在宅でも、同じように仕事ができ、同じようにパフォーマンスが発揮できる環境と文化が必要です。筆者たちが作りたかったのは、社内LANという名前の「ベルリンの壁」ではありません。社内LANと外を壁で隔てるという発想では、「社外にPCを持ち出すな」「社内にスマートフォンを持ち込むな」という発想になってしまいます。私たちが求めていたものは、テクノロジの恩恵を最大限活かして、どこでも好きなスタイルで仕事ができ、それでいて、ちゃんとセキュ

リティについても考慮されたしくみなのです。

筆者たちは、今回作ったこのしくみが、将来的に筆者たちの強みになると考えています。セキュリティ、コスト、運用負荷、ワークスタイル。両立が難しいと思われていたこれらの課題を技術の力で両立し、公的なお墨付きも得ています。その事実が、優秀な社員の獲得と、お客様からの信頼の両方を得ることにつながると確信しています。

Active Directoryが残る理由とは

しかしながら、ADは今後も使い続けようと考えています。AWSをはじめとする多数のクラウドサービスがADに対応している[注1]ことに加え、ID管理のしくみとして技術的にも優れており、費用対効果が高いと判断したためです。

Amazon VPC内にWindowsインスタンスを立て、そちらで行うドメインコントローラによるID管理は引き続き運用する予定です。OneLoginのエージェントがID情報を同期してくれますので、多数の外部サービスを使っている場合でも、ドメインからIDを削除すれば利用停止できるようになっています。

注1) IAM(AWS Identity and Access Management)の機能で、ドメインコントローラとID統合を行うことができます。

終わりなき旅

筆者たちの取り組みは、今はかなり尖った先進的なものに見られるかもしれませんが、けっして奇をてらったりリスクを恐れずチャレンジした、というものではありません。むしろ慎重にかつていねいに検討を重ねた結果です。

筆者たちが、大切なお金を銀行に預けることが当たり前のように、大切なデータをクラウドに預けることが当たり前になる世の中はすでにやってきています。筆者たちの取り組みが、クラウド時代におけるセキュリティに対する考え方を推し進める一助になれば望外の喜びです。SD

バックナンバーのお知らせ　BACKNUMBER

バックナンバー好評発売中！常備取り扱い店もしくはWebより購入いただけます

本誌バックナンバー（紙版）はお近くの書店に注文されるか、下記のバックナンバー常備取り扱い店にてお求めいただけます。また、「Fujisan.co.jp」（http://www.fujisan.co.jp/sd）や、e-hon（http://www.e-hon.ne.jp）にて、Webから注文し、ご自宅やお近くの書店へお届けするサービスもございます。

2013年2月号

第1特集
UNIXコマンド、fork、pipeを復習し、高度なスクリプティングへ
シェルスクリプティング道場

第2特集
忙しいITエンジニアのための
超効率的勉強法

一般記事
・Samba 4.0.0ファーストインプレッション

1,280円

2013年1月号

第1特集
いざというときに備える
システムバックアップ

第2特集
IT業界のキーパーソンに聞く
2013年に来そうな「技術」・「ビジョン」はこれだ!

特別付録
法輪寺電電宮情報安全護符シール

1,380円

2012年12月号

第1特集
判断をあおぐ／経緯を説明する／手順の理解を得る文章を書くためのアタマの整理術
なぜエンジニアは文章が下手なのか?

第2特集
高速・高機能HTTPサーバ
Nginx構築・設定マニュアル

一般記事
・エブリデープログラマの発想と実践

1,280円

2012年11月号

第1特集
もし、OpenFlowでやれと言われたら?
SDN、仮想化でネットワークはどうなる

第2特集
サーバの運用支援に
グラフィカルなリソース監視ツールを!
Muninが手放せない理由

一般記事
・SkeedSilverBulletとは?

1,280円

2012年10月号

第1特集
サーバ管理自動化の恩恵とリスクを見直しませんか?
Chef入門

第2特集
Iprコマンドが動く裏側のしくみがわかる!
Linuxプリント環境の教科書

一般記事
・SSH力をつけよう!
・JSX入門［前編］

1,280円

2012年9月号

第1特集
理解の壁を乗り越えるFinal Answer!
C言語のポインタは必要ですか?

第2特集
セキュリティ向上をあきらめない管理者になる!
SELinuxを無効にしない理由

一般記事
・機械学習ライブラリ『Mahout』入門［後編］

1,280円

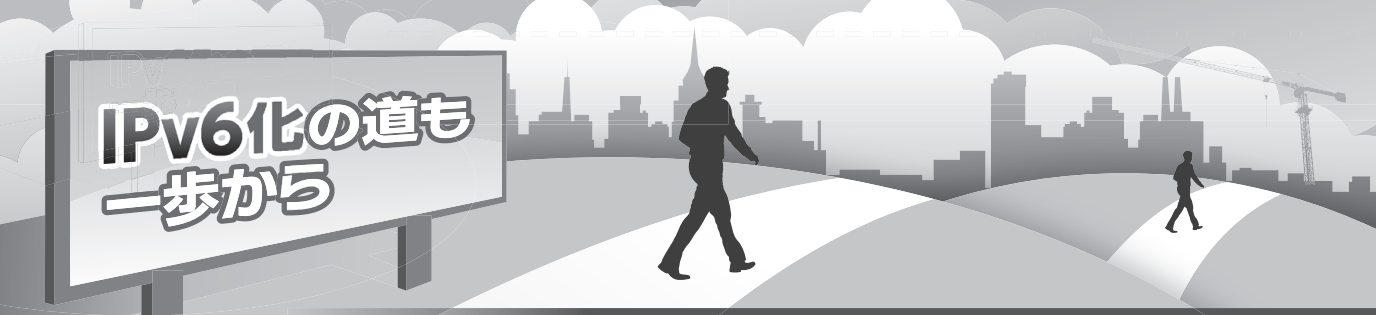
Software Design　バックナンバー常備取り扱い店							
北海道	札幌市中央区	MARUZEN&ジュンク堂書店　札幌店	011-223-1911	神奈川県	川崎市高津区	文教堂書店　溝の口本店	044-812-0063
	札幌市中央区	紀伊國屋書店　札幌本店	011-231-2131	静岡県	静岡市葵区	戸田書店　静岡本店	054-205-6111
東京都	豊島区	ジュンク堂書店　池袋本店	03-5956-6111	愛知県	名古屋市中区	三洋堂書店　上前津店	052-251-8334
	新宿区	紀伊國屋書店　新宿本店	03-3354-0131	大阪府	大阪市北区	ジュンク堂書店　大阪本店	06-4799-1090
	渋谷区	紀伊國屋書店　新宿南店	03-5361-3315	兵庫県	神戸市中央区	ジュンク堂書店　三宮店	078-392-1001
	千代田区	書泉ブックタワー	03-5296-0051	広島県	広島市南区	ジュンク堂書店　広島駅前店	082-568-3000
	千代田区	丸善　丸の内本店	03-5288-8881		広島市中区	丸善　広島店	082-504-6210
	中央区	八重洲ブックセンター本店	03-3281-1811	福岡県	福岡市中央区	ジュンク堂書店　福岡店	092-738-3322
	渋谷区	MARUZEN &ジュンク堂書店　渋谷店	03-5456-2111				

※店舗によってバックナンバー取り扱い期間などが異なります。在庫などは各書店にご確認ください。

デジタル版のお知らせ　DIGITAL

デジタル版Software Design 好評発売中！紙版で在庫切れのバックナンバーも購入できます

本誌デジタル版は「Fujisan.co.jp」（http://www.fujisan.co.jp/sd）と、「雑誌オンライン.com」（http://www.zasshi-online.com/）で購入できます。最新号、バックナンバーとも1冊のみの購入はもちろん、定期購読も対応。1年間定期購読なら5%割引になります。デジタル版はPCのほかにiPad／iPhoneにも対応し、購入するとどちらでも追加料金を払うことなく、読むことができます。



第4回 IPv6の最新情報／事例を効率よく収集する方法

IPv6普及・高度化推進協議会　IPv4/IPv6共存WG　アプリケーションのIPv6対応検討SWG
廣海 緑里 HIROMI Ruri　**渡辺 露文** WATANABE Tsuyufumi　**新 善文** ATARASHI Yoshifumi　**藤崎 智宏** FUJISAKI Tomohiro

IPv6の最新情報／最新事例を収集するには

本連載第2回のコラムで「情報は鮮度が大事！」とお伝えしましたが、今回は、その最新情報を収集するのに役立つ情報発信源となっている団体とその活動を取り上げます。また、昨年11月に開催された「Internet Week 2012」でのIPv6関連のセッションについても報告します。

IPv6プロトコルをもっと知りたいなら

きちんとIPv6関連のプロトコルについて学ぶなら、RFC（Request For Comments、詳細は後述）文書にあたるのが正攻法です。しかし、IPv6に関連した文書は100をはるかに超える数が発行されています。たとえばNDP（Neighbor Discovery Protocol：近隣者発見プロトコル）に関するものは約100ページと、1つの文書が壮大なものもあります[注1]。

確かに、それらの文書をひとつひとつ読むのは一番正しい方法ですが、IPv6の専門家としてではなく、利用者として技術を押さえたいのであれば、そうした情報を凝縮したセミナーに参加して知識を吸収する方法もあります。IPv4アドレスの中央在庫やアジア太平洋地域の在庫枯渇以降、IPv6を扱ったセミナーの開催も徐々に増えています。

注1） RFC4861　http://tools.ietf.org/html/rfc4861

IPv6プロトコルは誰がどうやって作ったのか?

IPv4やインターネットの父（や母）については、いろいろな文書で取り上げられているのでご存じの方も多いと思います。IPv6に関する仕様は誰が、どういう議論で作りだしたのかご存じでしょうか。

インターネットに関する技術文書は、RFCというシリーズになっていて、現在6千を超える文書があります。IPv6に関する文書だけでも数百あります。このシリーズはIETF（Internet Engineering Task Force）という団体が発行管理をしており、世界中から毎日多くの文書が提出され、吟味されています。新しい通信インフラ技術上でIPパケットをどう扱うかというネットワークの話、そのネットワークを構築するにはどういう設定をすればいいかという運用の話、Webでの表現形式はどうすればいいかというアプリケーションの話など多岐に渡っています。そのため、現在では分野に応じたワーキンググループに分かれて、共通技術に落とし込むための標準化活動が行われています。

「IPv6に関する仕様は誰が作ったのか」については特定するのが難しい状況です。というのも、インターネット黎明期と違い、現在、IETFでプロトコルの標準化に関係する人は非常に増えています。各RFCには著者はいますが、ワーキンググループとして取り組んだ場合、メールなどのオンライン媒体や会合でのフェースto

フェースの議論に多数の人が関わります。

IPv6の場合、中心となったワーキンググループは「IPNG（ipngwg）」[注2]です。これが「IP Version 6 Working Group（ipv6）」と改名し、現在は「IPv6 Maintenance（6man）」に続いています。日本の技術者や研究者も多数参加しており、いち早くIPv6の参照コードを公開し、多くのプラットフォームで参照／利用されています。

IETFで発行される文書はいくつか種類があり、「Internet Standard」と呼ばれるものもありますが、ほかの標準化団体に比べると緩やかな運用がされています。誰でもいつでも文書を書いて投稿し、問題提起したり、議論に参加することが可能です。文書名である、「Request For Comments」（コメント求む）がIETFの姿勢を表していると言われています。

前回取り上げた「IPv6プロトコルの特徴」は、RFC1550という技術文書としてまとめられた「新プロトコルへの要求仕様」に基づいて出てきたものです[注3]。次世代プロトコルへの要求仕様は、5章の中で16個ほど挙げられています。要求仕様に対していくつかの提案があり、最終的に残ったSIPPが「IPv6」となりました。RFC1752[注4]に選考からIPv6誕生までが記録されています。

IPv6アドレス、誰でも勝手に使っていいのか？

インターネット技術の共通化や標準化はIETFが担っていますが、ここから出てきた128ビットのIPアドレスを誰でも勝手に使うと、割当ての重複がおきたり、何が不正で何が不正じゃないのかといった判断ができなくなります。IPv6アドレスの割り当て管理もIPv4同様に、世界中で統一されたルールで運営されています。

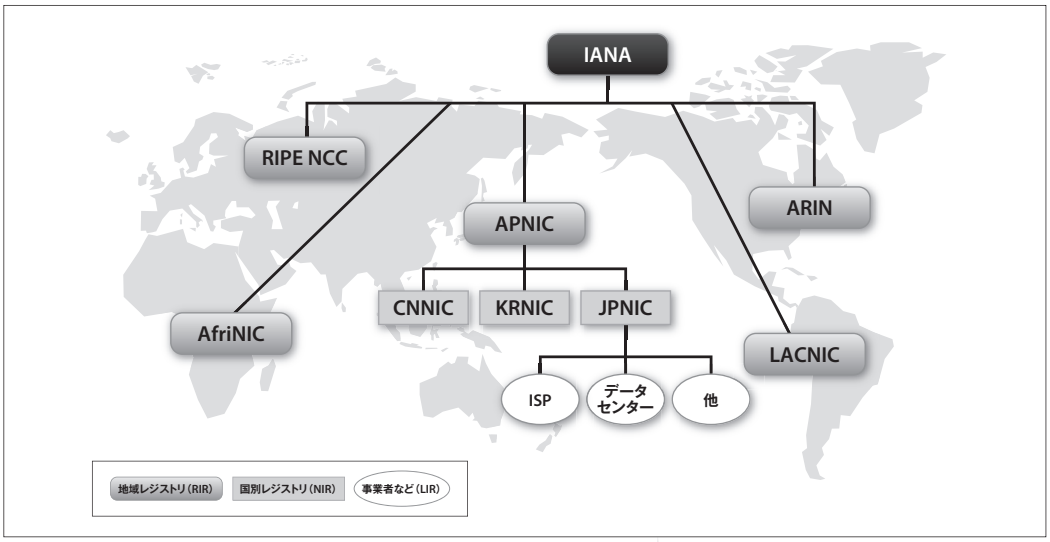
図1にあるように、管理組織体系はピラミッド型になっています。日本に関してはJPNIC（日本ネットワークインフォメーションセンター）が一元的に管理を担っています。共通の管理ルールや割り当てについての運用ルールは「アドレスポリシー」と呼ばれています。日本国内で議論し、アジア太平洋地域で議論し、必要があればさらにその上位組織（RIR間やIANA）というようにボトムアップで意見が提出できるしくみがあります。

注2） http://www.ietf.org/wg/concluded/ipngwg.html、現在は6manが後継となっている

注3） RFC1550　http://tools.ietf.org/html/rfc1550

注4） RFC1752　http://tools.ietf.org/rfc/rfc1752.txt

▼図1　IPアドレスの管理体系



たとえば最近では、プロバイダに依存しないIPv6アドレスを組織に割り当てる際の「マルチホーム要件の撤廃」がボトムアップで提案、議論され、実際に撤廃されました。日本では、ポリシーワーキンググループ[注5]が運営するオープンポリシーフォーラム[注6]で議論を行っています。ここには、年2回のJPOPMと呼ばれるオンサイトフォーラムと、IP-USERSというメーリングリストを用いたオンラインフォーラムの2つの議論の場があります。これらはIPアドレス管理について関心のある人は誰でも参加できます。議論の結果、アジア太平洋地域全体、あるいは全世界にとって共通したものとする必要がある場合にはJPNICやその上位組織であるAPNICなどと協力し、提案活動を行うサポート体制が構築されています。

たとえば、アドレスを割り振る際の分割単位について「まずISPなどには/32のIPv6アドレスブロックが割り振られ、ISPはそこから/48～/64のアドレスに分割してユーザ組織に割り当てる」といったことも、アドレスポリシーの議論がもとになっています。IPv6での運用はまだ始まったばかりであるため、初期割り振り、割り当てが議論の中心になっていますが、そのうち経路広告の単位の是正や、返却されたアドレスの扱いなどIPv4の運用の結果もたらされた議論がおきてくるかもしれません。

現在のIPv6アドレスの割り振りは、IPv4と同様でJPNICからISPに割り振りされたアドレスからISPがユーザへ割り当てるか、JPNICやAPNICから直接割り当てられたものを利用します。企業などで外部と接続はしないけれど内部のLANをIPv6で構築しておきたいという場合には、ULAというカテゴリのアドレスもあります。IPv6アドレスの種類と選び方については次回で取り扱う予定です。

注5） http://www.nic.ad.jp/doc/jpnic-01109.html
注6） http://www.jpopf.net/

▼写真1　Internet Week 2012の様子（写真提供：JPNIC）

Internet Week 2012 レポート

Internet Week[注7、注8]は、ハンズオン、チュートリアル、BoF、最新動向ディスカッションなどで構成される、インターネットに関する技術者の交流イベントです。商業的色合いが薄く、純粋にインターネットに関する技術を語る特色があります（**写真1**）。今回は、2012年11月19日～22日に秋葉原にて開催されました。Internet Week 2012でフォーカスされていたキーワードとしては、サイバー攻撃、IPv6、DNS、Open Flowが挙げられます。とくにIPv6については多くのセッションが割り当てられていました。ここでは、本連載に関連するIPv6のチュートリアルセッションについてレポートします。なお、各セッションの講演資料は、後日公開予定です。

これからはじめるIPv6

東日本電信電話（株）　安田歩氏

IPv4アドレス枯渇対応タスクフォースのセミナーでも講師を務めた安田氏によるIPv6の基礎に関する講義でした。IPv4アドレスの枯渇状況から、IPv6の主な特徴と課題、IPv6アドレスの基礎、アドレス自動設定、移行技術、アドレッシングとDNS、運用と非常に幅広く、通常は4時間かけて話される内容を2時間半に圧縮した非常に密度の濃い内容でした。安田氏が実運用の経験をふまえて話をされており、とくにアドレッシングに関して「他人に覚えてもらうアドレッシングも重要」というのが印象的で、この辺

注7） https://internetweek.jp/
注8） Internet Week 2012については、連載「Monthly from jus」（p.156,157）でも取り上げています。そちらにプログラム一覧を掲載していますので、併せてご参照ください。

はテキストには書かれていない部分で、受講の醍醐味の1つを味わえました。

「IPv6実践講座~トラブルシューティング、セキュリティ、アプリ構築まで~」

トラブルシューティング、セキュリティ、アプリ構築の3ジャンルを1つにまとめたセッションです。まず、IPv6トラブルシューティングでは、家庭ネットワーク／SOHO編、サーバ編、ISPネットワーク編の3つカテゴリに分けた講演が実施されました。

①家庭ネットワーク／SOHO編

日本電信電話(株)豊野剛氏

家庭内／SOHOからインターネットまでのネットワーク構成、接続モデルを確認したうえで、トラブル事例とその原因と思しき問題が解説されました。特筆すべきはIPv6がゆえにトラブルとなるケースは少なく、トラブルの多くはIPv4との共存に起因するということです。また、トラブルシュートにおいては、IPv4/IPv6のいずれかのみで通信すると状況が把握しやすいため、通信を切り替える手順を覚えておくとすばやくトラブルシュートが行えるようです。

②サーバ編

(株)クララオンライン　白畑真氏

クイック診断チャートを基にした問題切り分け方法が丁寧に解説されました。問題切り分けの観点としては、次の2つが挙げられます。

- サーバ／クライアント／ミドルボックス(リバースプロキシ、ロードバランサ、トランスレータ、CGNなど)のどこが問題か？
- どのプロトコル／サービスが問題か？

また、おもなチェックポイントは、フォールバック、DNS、Path MTUブラックホール、最近の実装やRFCなどの技術標準の変更のようです。OS、パッチによって挙動が異なるためとくに考慮が必要で、Windows Updateの適用状況も確認すべき、とのことです。

③ISPネットワーク編

NECビッグローブ(株)　川村聖一氏

IPv6アドレスの割り当てを中心に、川村氏の体験を交えて話されていました。運用で苦労しないポリシーを制定することが重要なこと、運用ツールはIPv6非対応のものが多く、作り込みが必要なケースがあるということが発表されました。ISPならではの話として、ユーザサポートにも言及されていました。一般ユーザを相手にする場合、確認するツールや、問題を絞り込んでくれるUIがあるとユーザに優しいようです。

◆ ◆ ◆

IPv6セキュリティについては、概要編、家庭ネットワーク／SOHO編、エンタープライズ／ISPネットワーク編の3カテゴリに分けた講演が実施されました。

④IPv6とセキュリティ(概要)

金沢大学　北口善明氏

「IPv6対応は、IPv6への移行ではなく、IPv4とIPv6との二重のネットワーク管理である」と前置きしたうえで、セキュリティ面の課題が仕様面、実装面、運用面に整理されて解説されました。とくに重要な点は、守るべき対象と同一のリンク内に侵入させないこと、仕様上明確でない上限値を実装上で設けること、デュアルスタックの挙動を理解すること、IPv6化しないセグメントもIPv6通信のモニタリングが必要なこと、これら4点のようです。

⑤家庭ネットワーク／SOHO編

NTTコミュニケーションズ(株)　山形育平氏

装置、端末、サービスともにIPv6 Readyになった現在において、家庭ネットワーク／SOHOを構築、利用する際に気をつける点、ISPや企業が個人向けサービスを行う上で気をつける点について解説されました。セキュリティで気をつけるべき点はIPv4とほとんど同じだが、IPv6の仕様やデュアルスタックによる影響などによりIPv4とは違う視点でのセキュリティの検討が

必要になる場合もあり、IPv6固有の仕様を理解し、IPv6固有の仕様をふまえたネットワークの検討を行うべき、とのことです。

⑥エンタープライズ／ISPネットワーク編

さくらインターネット(株)　大久保修一氏

ISPネットワークに関しては、ルータを守るために、アタックからの防御、負荷増大の抑制、統計情報の収集などいろいろな対策がとられているようです。エンタープライズ環境については、オフィスネットワークなどの内部ネットワークにもIPv6グローバルアドレスを使用するケースがあること、ステートフルインスペクション対応ファイアウォールの実装が必須であること、サービス監視、マネジメントをきちんと行うことが解説されました。

◆ ◆ ◆

IPv6プログラミングについては、BSD Socket APIを使用したアプリケーションのIPv6化の方法と、LL(Lightweight Language)のIPv6対応状況が取り上げられました。

⑦アプリケーションのIPv6対応概要

(株)リコー　大平浩貴氏

C言語によるBSD Socket APIを使用したアプリケーションのIPv6化について解説されました。クライアント、サーバのそれぞれについて、ソースコード例を交えながら実際の関数、構造体を用いたIPv6化の手法が紹介されました。アプリケーションのIPv6化はIPv6/IPv4両方のデュアルスタック対応をする必要があり、単純に関数を変更するだけではダメで、選択機構や並列受信機構などが必要となる場合があります。

⑧スクリプト言語とIPv6 2012

関根佳直氏

2012年11月時点におけるPerl、PHP、PythonのIPv6対応状況が解説されました。一言で「プログラミング言語がIPv6に対応している」と言うのは難しいため、ソケット、名前解決(DNS)、各種(L7)ネットワークプロトコル、IPv6アドレスの処理など、機能ごとに判断されていました。各言語の対応状況だけでなく、ソースコード例を交えながら実際に使用するライブラリ、クラス、メソッドが紹介されました。使用に注意が必要なものもありますが、いずれの言語もIPv6 readyの状況にあるようです。

IPv6各種移行／共存技術解説

既存ネットワークにIPv6を導入する方法として、各種共存技術の解説と実際の利用方法が1つのセッションで解説されました。

①IPv6共存・移行技術基礎

シスコシステムズ合同会社　印南鉄也氏

IPv4アドレス枯渇、IPv6導入の指針、構成モデルと移行のステップ、移行のための要素技術、セキュリティと幅広い内容が短時間に集約されていました。とくに注力されていたのが、移行のステップと要素技術の部分で、混乱しやすい内容がわかりやすく解説されました。重要なポイントは、IPv6化はIPv4枯渇とセットで考えることと、運用面でIPv6はIPv4ほど技術者が慣れていないということのようです。

②企業におけるIPv6導入のポイント

(株)インテック　廣海緑里氏

本連載の執筆者の一人による講演です。主に企業ネットワークにIPv6を導入する際に考えたい8つのポイントを紹介しました。8つのポイントに加えて重要なことは、企業では業務にとって何が最適かを選ぶこと、信頼できる筋の信頼できる情報を入手すること、情報をふまえて自分たちで考えることです。

③金沢星稜大学におけるIPv6導入事例

学校法人稲置学園 情報基盤センター　井上清一氏

実際に、組織ネットワークにIPv6を導入した事例が紹介されました。新棟の建設と合わせて情報システムの刷新を行い、その際にIPv6導入

を行った様子が話されました。IPv6導入のきっかけは、エンドユーザ側の環境の変化と話されていました。少人数のスタッフで0からIPv6化を行ったため、目的と導入範囲を絞り込んで取り組んだとのこと。また、組織内の取りまとめがたいへんだったようで、IPv6単品ではなく情報システムの刷新事業と合わせて行い、新機能・性能向上以外の提案も行ったようです。最後に、「エンドユーザに直接的なメリットをもっと！」と締めくくられていました。

IP Meeting 2012〜人のチカラ、インターネットのチカラ〜

IP Meeting 2012では、2012年の最新動向とそれをふまえた議論が行われました。IPv6に関しては「IPv6とIPv4の動向と対策」と題して、3つの講演と1つのパネルディスカッションが行われました。

①移転とアドレス売買

Geekなページ　あきみち氏

IPv4アドレス在庫枯渇問題をおさらいしたうえで、IPアドレスの返却、移転、売買について解説されました。とくに売買については、税務上の扱いや売り手を探す方法、IPv4アドレスブローカーなど一歩踏み込んで解説されました。今はアドレス移転／売買が活性化しつつあるように見えるが、いつまでも続かないと感じられているようです。

②World IPv6 Launch 〜その後〜

ISOC-JP　江崎浩氏

World IPv6 Launch（以下、W6L）のその後の状況について、総務省の研究会での議論や、IPv6普及・高度化推進協議会のアクセス網サブワーキンググループでの議論を、中心に解説されました。W6Lの次にやることとして、IPv6サービス提供のデフォルト化を唱えていました。

③IPv6/IPv4共存技術の動向

ソフトバンクテレコム　松嶋聡氏

後から後からいろいろ出てくる共存技術について、ステートフル／ステートレスとIPv6 over IPv4/IPv4 over IPv6という切り口で分類、マッピングされ、各技術の変遷をふまえてわかりやすく解説されました。とくに印象的だったのは、IETFでのコイントスの映像が再生されたシーンです。世界共通の技術仕様を決める際に、議論を重ねても結論が出ない場合には、コイントスという「運を天に任せる」方法が用いられるのは意外でした。

④枯渇後、IPv6とIPv4は共存できる環境が整っているのか

モデレータのあきみち氏を中心に、國武功一氏（㈱ビーコンエヌシー）、馬場達也氏（㈱NTTデータ）、佐藤良氏（㈱コナミデジタルエンタテインメント）の各パネリストが、IPv6対応で苦労するポイントについてディスカッションを行いました。「地雷を踏んだ経験談」の話で盛り上がり、パネリストの方々が苦労された経験が共有されました。

終わりに

今回は、情報発信源となっている団体およびその活動と、「Internet Week 2012」でのIPv6関連のセッションについて紹介しました。

次回は、いよいよ実行フェーズに入ります。まず、具体的な対応内容と注意点や落とし穴として、実行フェーズで気をつけたい事項を取り上げます。ご期待下さい。SD

連載を通じての質問やコメント、取り上げてほしいトピックを募集します。最終回などで取り上げて、追加解説したいと考えています。質問、コメントは編集部（sd@gihyo.co.jp）までお送りください。

ハイパーバイザの作り方

ちゃんと理解する仮想化技術

第6回

Intel VT-xを用いたハイパーバイザの実装「仮想CPUの実行処理・その1」

浅田 拓也（ASADA Takuya）Twitter @syuu1228

はじめに

前回までに、5回にわたりハードウェアが提供している仮想化支援機能を解説しました。

今回からは、実際にどのようにして仮想化支援機構を使い、ハイパーバイザを実装するのかを解説していきます。解説には、実装の具体例として現在FreeBSDへの実装が進められているハイパーバイザ「BHyVe（ビーハイブ）」を用います。

BHyVeとは

BHyVeはNetAppにより開発された、FreeBSDで動く新しいハイパーバイザです。2011年に発表され、実装が進められています。現在、FreeBSD10の新機能の1つとしてリリースすることを目指しています[注1]。設計はLinux KVMに似ており、FreeBSD版のKVMとも言えるでしょう。ただし、BHyVeは現在開発の初期段階であるため、現在のKVMに比べ非常に機能は限られています。現状、実用に用いるのは難しい開発版の機能となります。しかし、ハイパーバイザの実装方法を学ぶための教材としては、以下の二点が優れています。

1つは、最低限の機能のみ実装しているためKVMと比べるとコード量が非常にコンパクトであることです。もう1つは、デバイスエミュレーションなどを行うユーザランドプログラム（詳細は後述）がシンプルなものであることです。KVMではユーザランドプログラムに既存のエミュレータであるQEMUを流用しており、これにより既存OSとの高い互換性が得られていますがコードの見通しが悪くなっています。そこで、本連載ではKVMではなくBHyVeを用いて解説を行うことにします。

解説対象のバージョン

BHyVeは現在、リリース版が存在しません。また、開発の初期段階であるため、大きな変更が加えられることも予想されます。

そこで、本連載では現時点でのFreeBSD-CURRENTでスナップショットが公開されている最新リビジョンであるr245673を用いて解説を行います。r245673のインストールディスクは、次のアドレスからダウンロードできます。

```
ftp://ftp.freebsd.org/pub/FreeBSD/snapshots/amd64/amd64/ISO-IMAGES/10.0/FreeBSD-10.0-CURRENT-amd64-20130119-r245673-release.iso
```

r245673のソースコードは次のコマンドで取得できます。

```
svn co -r245673 svn://svn.freebsd.org/base/head src
```

注1）2013年1月にFreeBSD-CURRENTの開発ツリーにマージされました。

また、BHyVe用にコンフィグレーションされたゲストマシンのディスクイメージが配布されており、次のアドレスからダウンロードできます。

```
http://mirrors.nycbug.org/pub/BHyVe/r244024/ア
diskdev.xz
```

BHyVe の動作環境

BHyVeを試すには、Intel VT-xとEPT(Extended Page Tables)をサポートしたCPUが 必要です。

今のところAMDのCPUには対応していません。どのモデルのCPUがEPTをサポートしているかはark.intel.comで調べられます。 簡単な目安としては、Core i3, i5, i7ならばほぼすべてのCPUが対応しています。CeleronやPentiumなどの廉価版CPUでもEPT対応モデルが一部存在しています。 実機にインストールするのが最も確実ですが、最近のVMware(VMware Player・ VMware Workstation・VMware Fusion)ならばNested VM[注2]に対応しているため仮想マシン上で試すこともできます。

BHyVe が提供する機能

現状、BHyVeではゲストOSとしてFreeBSD/amd64のみをサポートしています。

ハードディスクコントローラとしては、標準的なIDEやAHCIコントローラのエミュレーションはサポートせず、準仮想化ドライバであるvirtio-blkをサポートしていま す。

NICコントローラとしても同様に、標準的なIntel e1000などのデバイスのエミュレーションをサポートせず、準仮想化ドライバであるvirtio-netをサポートしています。virtioは多くの場合、標準的なデバイスのエミュレーションと比較して高い性能が得られますが、ゲストOSにvirtioドライバをインストールし、/boot/loader.confの設定を行う必要があります。

システムコンソールとしては、標準的なビデオデバイスをサポートはサポートされず、PCI 接続の16550互換シリアルポートのみをサポートしています。PCI接続のシリアルポートをシステムコンソールとして使うのは非標準的なハードウェア 構成であるため、これについても/boot/loader.confの設定を行う必要があります。 また、ビデオデバイスをサポートしないため、X11を起動してGUI環境を表示することはで きません。

BHyVeがエミュレート可能なデバイスは上述の3種類だけですが、Intel VT-dを用いて実 機上のPCI・PCI Expressデバイスをゲストマシンへパススルー接続できます。 この他、割り込みコントローラのエミュレーション(Local APIC、IO-APIC)や、タイ マデバイスのエミュレーション、ハードウェア構成をゲストOSへ伝えるのに必要なAPIC などをサポートしています。 また、BIOSやUEFIなどのファームウェアをサポートしていないため、ディスクイメージ からブートローダをロードしてゲストOSを起動することができません。 このためにハイパーバイザの機能としてFreeBSDカーネルをロードしゲストマシンを初期 化するOSローダが実装されています。

BHyVe の構成

BHyVeは、カーネルモジュールとユーザランドプログラムの2つから構成されます。

カーネルモジュールvmm.koは、CPUに対してVT-x命令を発効するなど、ハードウェアに近い処理を行います。

ユーザランドプログラム/usr/sbin/bhyveは、ユーザインタフェースを提供し、ハードウェアエミュレーションを行います。

また、BHyVeには/usr/sbin/bhyveloadというツールがあります。前節で述べたとおり、BHyVeではディスクイメージ上のブートローダを実行できません。このため、ゲストカーネルをロードして起動可能な状態に初期化するゲストOSローダ(/usr/sbin/bhyveload)が付属します。

注2) ハイパーバイザ上でハイパーバイザを実行可能にする機能。

▼リスト1　BHyVe実行イメージ

```
# kldload vmm.ko
# ls /dev/vmm
# /usr/sbin/bhyveload -d diskdev -m 1024 guest0

Consoles: userboot

FreeBSD/amd64 User boot, Revision 1.1
(root@bhyve, Sat Dec  8 17:35:59 PST 2012)
Loading /boot/defaults/loader.conf
/boot/kernel/kernel text=0xd01c58 data=0x15e9a0+0x2bb490 syms=[0x8+0x14bbd8+0x8+0x1a7a89]
/boot/kernel/virtio.ko size 0x5a90 at 0x1810000
/boot/kernel/if_vtnet.ko size 0xd910 at 0x1816000
/boot/kernel/virtio_pci.ko size 0x6cc0 at 0x1824000
/boot/kernel/virtio_blk.ko size 0x6b08 at 0x182b000
|
  ______                    ____   _____ _____
 |  ____|                  |  _ ¥ / ____|  __ ¥
 | |___ _ __ ___  ___ | |_) | (___ | |  | |
 |  ___| '__/ _ ¥/ _ ¥|  _ < ¥___ ¥| |  | |
 | |   | | |  __/  __/| |_) |____) | |__| |
 | |   | | |    |    ||     |      |      |
 |_|   |_|  ¥___|¥___||____/|_____/|_____/    ```
                                             s` `.....---.......--.```   -/
 ?????????????Welcome to FreeBSD???????????? +o   .--`         /y:`      +.
 ?                                           ?  yo`:.           :o      `+-
 ?  1. Boot Multi User [Enter]               ?   y/             -/`   -o/
 ?  2. Boot [S]ingle User                    ?  .-                ::/sy+:.
 ?  3. [Esc]ape to loader prompt             ?  /                   `--  /
 ?  4. Reboot                                ? `:                      :`
 ?                                           ? `:                      :`
 ?  Options:                                 ?  /                      /
 ?  5. Configure Boot [O]ptions...           ?  .-                    -.
 ?                                           ?   --                  -.
 ?                                           ?    `:`              `:`
 ?                                           ?      .--          `--.
 ?                                           ?         .---.....----.
 ??????????????????????????????????????????

Booting...
# ls /dev/vmm

guest0
# /usr/sbin/bhyve -A -I -m 1024 -s 0:0,virtio-blk,diskdev10 ¥
-S 31,uart,stdio guest0

GDB: no debug ports present
KDB: debugger backends: ddb
KDB: current backend: ddb
Copyright (c) 1992-2012 The FreeBSD Project.
Copyright (c) 1979, 1980, 1983, 1986, 1988, 1989, 1991, 1992, 1993, 1994
  The Regents of the University of California. All rights reserved.
FreeBSD is a registered trademark of The FreeBSD Foundation.
FreeBSD 10.0-CURRENT #0 r243948: Thu Dec  6 14:03:56 UTC 2012
    root@kaos.glenbarber.us:/usr/obj/usr/src/sys/GENERIC amd64
……(省略)……
# ls /dev/vmm
guest0
# /usr/sbin/bhyvectl --vm=guest0 --destroy
# ls /dev/vmm
```

①カーネルモジュールをロードし、BHyVeを使用可能にする

②vmm.koがロードされると/dev/vmmディレクトリが作成される。この状態ではVMインスタンスが存在しないので、ディレクトリは空

③/usr/sbin/bhyveloadはVMインスタンスを作成し、BSDカーネルをディスクイメージから読み込みます。そしてVMインスタンスのメモリ領域へロードして起動可能な状態を作る。カーネルをロードするプログラムはFreeBSDのブートローダのコードをそのまま使っているため、ブート時に表示されるのと同じメニューが表示される。しかし、これはホスト側で実行されているプログラムでゲストマシンの起動は始まってない。引数-dはディスクイメージ、-mはメモリサイズ、最後の引数はVM名を指定している

④/usr/sbin/bhyveloadが作成したVMインスタンスは、/dev/vmm以下にデバイスファイルとして表示される

⑤/usr/sbin/bhyveは/usr/sbin/bhyveloadが初期化したVMインスタンスを実する。引数-mはメモリサイズ、-sと-Sはゲストマシンの PCI上に接続される仮想デバイス、最後の引数はVM名を指定している

⑥shutdownを行なってbhyveプロセスを終了した後も、カーネル側にVMインスタンスの状態が残る

⑦VMインスタンスを削除しゲストメモリを開放するには、これを削除する必要がある。/usr/sbin/bhyvectlの--destroyオプションを使い、VMインスタンスを削除する

⑧VMインスタンスを削除するとデバイスファイルも消去される

/usr/sbin/bhyveloadはFreeBSDブートローダをFreeBSD上で実行可能なプログラムに 改変し、ゲストマシンのディスクイメージからカーネルを読み込んでゲストメモリ空間へ 展開するようにしたものです。 /usr/sbin/bhyveloadを実行すると、FreeBSDのブート時に表示されるのと同じメニュ ーが表示されます。このため、一見するとゲストマシンの実行が開始されたように見えま す。 しかし、これはホストOSで/usr/sbin/bhyveloadが出力している画面で、ゲストマシン の起動は開始されていません。

このほか、ゲストマシンの実行とは直接関係しませんが、VMインスタンスの削除などを行う ためのVMインスタンス管理ツールとして/usr/sbin/bhyvectlが提供されています。

これらのユーザランドのプログラム群は、VM管理用のライブラリ(libvmmapi)を通してゲストマシンの初期化や実行をします。libvmmapiはmmapやioctlを発行し、vmm.koが提供するデバイスを操作します。

全体図を**図1**に示します。

BHyVe の使い方

BHyVeを試すには、ホストマシンのセットアップとゲストマシンのディスクイメージの準備が必要です。前述のアドレスを参照してください。

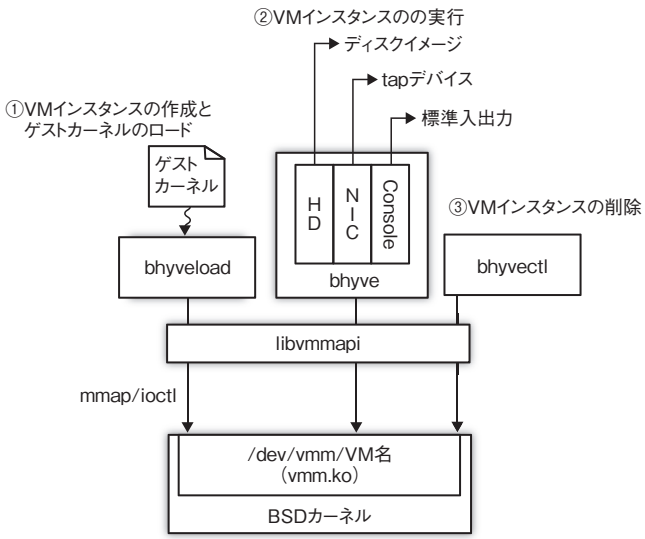
リスト1にシェルからBHyVeを用いてゲストマシンを実行する方法を示します。

使い方はKVMに似ていますが、VMインスタンスの状態がVM実行プログラムと分離されており、/dev/vmm/VM名に保持される点がKVMと異なっています。このため、OSローダがVM実行プログラムと別プログラムになっていたり、VM実行プロ グラムを終了とは別にVMインスタンスの削除コマンドを実行する必要が出てきます。

まとめ

今回は、BHyVeの概要と使い方について説明しました。 次回からいよいよソースコードの解説に移りたいと思います。SD

▼図1 BHyVeの構成



Emacs 64bit化計画!

使いやすいエディタ環境を作りませんか!

最終回 第7回

.NETコンポーネントの利用方法

太田 博志 *OOTA Hiroshi* ● TwitterID @h2oota　イラスト：黒崎 玄

GNU EmacsからKeePassを使う

さてWin64版のGNU Emacsを紹介する目的で始めた本連載も今回で最終回を迎えました。

前回の記事で今回はCOMの解説を行うと予告しました。COM利用の実践編として、ソフトウェアのプロジェクト管理とバグトラッキングを行うTracというツールのチケットシステムをCOMで実装したXMLRPCをGNUSのバックエンドとして使ってGNUSからアクセスすることを計画していました。筆者はメールをOperaのメールリーダーで読み書きしていますが、メールの件数が増えてきたせいもあり起動の遅さや検索性の悪さが目立ち、メールシステムの更新を考えています。この際メールの読み書きをGNUSで行うことを企み、GNUSのIMAPを設定してみましたが、設定を行う際にパスワードをプレーンテキストでファイルに書くことがどうしてもイヤだったので、脇道にそれてKeePass V2をGNU Emacs(Win64)から利用する機能を作成しました。KeePass V2はC#で記述されていて、その機能はMicrosoftの.NETフレームワークのコンポーネントとして外部プログラムから利用できます。外部プログラムの機能をパイプ以外で利用することはUNIX版Emacsではできないwindows版Emacsの大きな利点となります。

今回は予定を変更して、.NETコンポーネントをGNU Emacs(Win64)から利用するKeePassインターフェースモジュールを紹介します。

GNU Emacs(Win64)から.NETコンポーネントを使う

KeePass V2は.NETフレームワークにて開発されていて、その機能を他のプログラムから利用できます。今回はGNU Emacs(Win64)にKeePass V2のコンポーネントを利用するインターフェースを、ダイナミックライブラリサポート機能を利用し実装します。

.NETフレームワーク

.NETフレームワークとはMicrosoftが開発しているCLI(Common Language Infrastructure)を基盤とアプリケーション開発、実行環境の総称です。CLIでは高級言語で記述されたプログラムを中間言語(CIL)にコンパイルし、実行時にCLR(Common Language Runtime)がネイティブコードに変換して実行します。

CLRはJava実行環境をイメージするとわかりやすいと思います。Microsoftはコンポーネント開発をCOMから.NETフレームワークに移行してきていて、新しいコンポーネントは.NETフレームワークで提供されるようになっています。ネイティブコードへの変換はキャッシュされるので、最近のPCでは起動時のスピードも気にならなくなっています。CLRの管理下で実行されるプログラムコードはマネージドコードと呼ばれ、メモリ管理もガベージコレクション

を持つCLRが行います。CLRの管理外にあるプログラムコードはアンマネージドコードと呼ばれ、基本的には相互に直接呼び出せません。

C++/CLI

マネージドコードとアンマネージドコードの混在は基本的にはできませんが、膨大なアンマネージドコードの資産を利用するためにMicrosoftはC++を拡張して混在を可能にしました。前回の記事でC++/CLIを使い.NETフレームワークコンポーネントを利用するのは困難だと書きましたがC++/CLIを見くびっていました、簡単に両者を混在して利用することができます。訂正します。VC++のCOM拡張を利用せずに記述した前回のCOMインターフェースより簡単なくらいで拍子抜けしました。

Emacsとのインターフェース

C++/CLIはネイティブコードを生成します。Emacsとのインターフェースはextern "C"で宣言したCの外部名を持つ関数を介して行います。.NETコンポーネントのロードなどはC++/CLIのランタイムが全部行ってくれます。.NETコンポーネントを実装するファイルの検索で.exeと、.dllが区別されないことに注意が必要です。#usingでKeePass.exeを指定しましたが、同一ディレクトリにKeePass.dllがあるとそちらのロードを試み、異常終了します。

今回作成したdllをkeepassdll.dllという名前にしたのはそれを回避するためです。

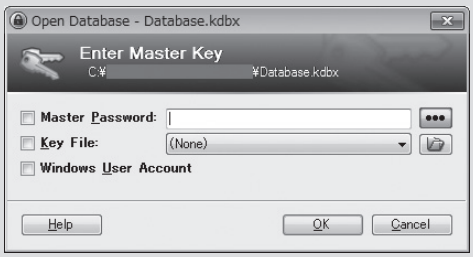
Column KeePass V2

昨今、マルウェアやフィッシング(phishing)の横行により、複数のサイトで同一のパスワードを利用するパスワードの使いまわしは厳に慎むべき行為になってきています。加えて、憶えやすいパスワードは辞書攻撃で瞬殺されるので、ランダムなパスワードを利用することも要求されていますが、人間の記憶力には限界があり、利用頻度の低いパスワードを憶えておくことは困難です。この矛盾する命題を解決するためにパスワードを管理するプログラムが開発されています。

KeePass V2[注1]はそのようなパスワードの管理を行うプログラムの1つで、オープンソースで開発されています。データベースは暗号化されていてパスワードを安全に保管することができます。スマートフォン用のプログラムも開発されているので、DropBoxなどのオンラインストレージ上にデータベースを置けば、バックアップを兼ねスマートフォンとのパスワードデータベースの共有を行えます[注2]。データベースの暗号鍵はパスワード、キーファイル、Windowsアカウント(データベース作成者以外は開くことができない)で保護されます(**図1**)。

マルウェアなどからの防御にも工夫が凝らされており、パスワードなどを入力するダイアログボックスを、通常のデスクトップとは別の一時的に作成したデスクトップ(セキュアデスクトップと呼ばれています)上に表示することもできます。セキュアデスクトップは通常のデスクトップとは切り離されてるので、スクリーンショットやキーロガーから保護できます。

▼図1 KeePass V2



注1) http://keepass.info/
注2) http://keepass.info/download.html のContributed/Unofficial KeePass Portsに一覧があります。筆者はMiniKeePassを利用しています。

マネージドオブジェクトの利用

マネージドコードはクラス名、メソッド名、引数型などの完全なインターフェース情報を持っています。C/C++ではこれらの情報はヘッダファイルに記述されていましたが、C++/CLIでは #using <KeePass.exe> と書くだけでコンポーネント中のインターフェース情報を利用できます。ソース中に#usingを記述するほかにコンパイラオプション/FUで指定もできます。

マネージドオブジェクトはC++/CLIで拡張されたハンドル型で参照します。マネージドオブジェクトはハンドルで参照しますが、ガベージコレクションのためにマネージドオブジェクトの参照は追跡されています。ハンドルがスコープを離れる(関数からリターンするなど)と参照が解放され、どこからも参照されていないマネージドオブジェクトは回収されます。EmacsLispにマネージドオブジェクトを渡すためにCLRが提供しているSystem.Runtime.InteropServices.GCHandleでガベージコレクタから保護されたハンドルを取得します。

GCHandleで取得したハンドルは明示的に解放する必要がありますが、これを格納しているEmacsLispオブジェクトの解放時に解放します。マネージドオブジェクトをサポートするためにC++の構文、演算子も拡張されています(**表1**)。

実際のコードを参照するとわかりやすいので、KeePassインターフェースの一部を掲載します(**リスト1**)。tovoidptrはマネージドオブジェクトのハンドルからGCHandleを生成し、C++のvoid*に変換します。fromvoidptrはその逆でC++のvoid*からマネージドオブジェクトのハンドルを得ます。find_group_childはKeePassのグループオブジェクト[注3]の下にある指定の名前を持つエントリオブジェクトを検索します。

▼表1　C++/CLIで拡張された構文

C++	C++/CLI	
クラス名 *	CLI クラス名 ^	変数定義
new	gcnew	
NULL	nullptr	
	safe_cast<T>(v)	キャスト
&	%	参照(アドレス)演算子

KeePassデータベースの保護

コラムで紹介したようにKeePassはデータベースの保護に気を配って作成されています。データベースのアクセス認証の他にも、パスワードなどの秘密情報はプロテクトストリングと呼ばれているオブジェクトで管理していて、復号された状態でのアクセスを最小限にするようKeePassは作られています。せっかくKeePassでパスワードを管理しているのにEmacsのインターフェースを追加することで、全体のセキュリティを損なうことがないようにすべきです。

データベースのアクセス認証をKeePassに行わせる

アクセス認証はKeePassのダイアログで行い、Emacsではデータベースパスワードを読みこまないようにします。Emacsからパスワードを与えてオープンするインターフェースは実験過程で作成したものがそのまま残っていますが、データベースの保護を考えるとこれの使用は特定のパスワードだけ格納するような限定したデータベースに限ったほうが良いと思います。

コラムで紹介したセキュアデスクトップも同様に利用するために、KeePassに実装されているものをそのまま利用しようと試みましたが、KeePassメインウインドウなしで利用するのは不可能なようでしたので、通常デスクトップから分離したデスクトップを生成する部分は独自実装としました。

セキュアデスクトップはEnumDesktop Window APIの対象外となるようDESKTOP_ENUMERATEを与えずに生成します。名前も乱数で生成したものを使用するので、マルウェ

注3) ディレクトリのようなオブジェクト、実際のパスワード等を格納するエントリオブジェクトを子として持つことができます。

▼リスト1　CLIオブジェクトの利用例

```
template<typename T>
static inline void *tovoidptr(T p)
{
    return (p == nullptr)
    ? NULL
    : static_cast<void *>(
        System::Runtime::InteropServices::GCHandle::ToIntPtr(
    System::Runtime::InteropServices::GCHandle::Alloc(p)));
}

template<typename T>
static inline T fromvoidptr(void *p)
{
    return
    safe_cast<T>(
        System::Runtime::InteropServices::GCHandle::FromIntPtr(
    System::IntPtr(p)).Target);
}

DLLEXPORT
void *find_group_child(void *group, const char *n)
{
    KeePassLib::PwGroup^ g =
    fromvoidptr<KeePassLib::PwGroup^>(group);
    System::String^ name = asString(n);
    System::Collections::Generic::IEnumerator<KeePassLib::PwGroup^>^ it
    = g->Groups->GetEnumerator();
    while(it->MoveNext()) {
        if (name->Equals(it->Current->Name,
            System::StringComparison::CurrentCultureIgnoreCase))
        return tovoidptr(it->Current);
    }
    return NULL;
}
```

アがこのデスクトップへアクセスすることは非常に困難になっています。

データフィールドのアクセス制限

KeePassデータベースではパスワードの一元管理を行っているので、メールボックス以外のパスワードも格納してあります。Emacsはスクリプタブルな環境であり、マルウェアが侵入する可能性を否定できません。不必要なパスワードにはアクセスできないようにしたいところですが、1つのマスターパスワードで複数のパスワードを管理するというKeePassのコンセプトと矛盾してしまいます。セキュリティと使い勝手は背反な場合が多くバランスが難しいところです。一方、今回の対象のGNUSの認証方式を考えてみると、メールボックスへの認証とSMTP認証になります。どちらもCRAM-MD5などHMACによるものとプレーンパスワードを利用する2種類の方式が用いられています。

HMACによるものは平文のパスワードをネットワークに流す必要がないプロトコルです。秘密鍵とメッセージを連結しそれのハッシュにて認証を行います。この方式では秘密鍵を認証モジュールの外に渡さずに認証データを生成できます。言い換えれば、EmacsLispからパスワードフィールドにアクセスする必要がないということです。

プレーンパスワードを用いるものはEmacs Lispにパスワードを渡す必要があります。認証モジュールにネットワークストリームを渡し、認証モジュールがそのストリームに出力するという方法を使えば渡さずに済みそうですが、ネッ

トワークストリームを偽造されたら盗まれるのでやはりダメです。

筆者が利用しているメールサービスのうち平文のパスワードが必要なものはGmailだけでした。Gmailでは二段階認証を用いているので、二段階認証に対応していないIMAPなどのプロトコルでは「アプリケーションパスワード」と呼ばれる固有のパスワードを生成して利用することになっています。これをKeePassのカスタムフィールドに格納し、そのフィールドへのアクセスを許可することにしました。これならば、EmacsLispから他のパスワードを読み出す必要がありません。

Emacs側インターフェース

GNUSは.netrcに基づいたファイルに保存されたパスワードへのアクセス機能をもっています。といえば聞こえはよいですが.netrcの実体は暗号化されていないテキストファイルです。GNUSではサーバとプロトコルの組(キー：K)に対してユーザ名とパスワードの組(値：V)を返す簡易なKVデータベースとして実装されています。自分の管理下にない既存のコードの修正は最小限にすることが、筆者のポリシーです。この簡易データベースの部分には触らずにパスワードフィールドが特殊な形式だった場合にKeePassを参照するようにしました。具体的には、パスワードフィールドがkeepass:で始まる場合は通常の処理を離れてKeePassを参照するようにします。

```
keepass:ファイル名?グループ名[&グループ名 ➥
[&グループ名]]&エントリ名[#visibleフィールド]
```

上記の形式にします。

ファイル名は.kdbxの拡張子を持つKeePassデータベースファイル名、グループ名、エントリ名はそれぞれKeePassのグループ名、エントリ名に相当します。最後のvisibleフィールドでEmacsLispからアクセス可能なフィールドを追加します。デフォルトはTitle、UserName、URL、Notesです。visibleフィールドはデータベースをオープンする際に指定し、オープン後には変更不可としています。こうすることでKeePassデータベースでパスワードを格納しているPasswordフィールドへのEmacsLispからのアクセスを保護しています。KeePassのエントリに対し、指定したメッセージに対するHMACを計算する関数と、指定したフィールドを取得する関数を定義しました。これをGNUSの認証システムに組み込む方法ですが、Emacsにはdefadviceという機能があり、関数呼び出しを奪い、定義と異なる処理を実行できます。

今回は、CRAM-MD5とプレーンテキスト認証の2方式に対応です。組み込む個所を調べる一般的な方法はありません。丹念にコードを追跡していくしかない地道な作業ですが、今回対象としているCRAM-MD5に対してはrfc2104-hashという関数、プレーンテキスト認証に対してはプロトコルごとに異なるのでIMAPだけの対応にしてimap-login-authという関数のadviceを作成します。これらはそれぞれ

```
(rfc2104-hash HASH BLOCK-LENGTH HASH- ➥
LENGTH KEY TEXT)
(imap-login-auth BUFFER)
```

と定義されています。GNUSの認証モジュールが.netrcに対してサーバ、プロトコルのペア(キー)を検索した値(ユーザ名、パスワードのペア)がrfc2104-hashではパスワードがKeyとして直接、imap-login-authではバッファローカルな変数imap-username、imap-passwordとして渡されます。

インストールと利用方法

❶KeePass 2.20.1[注4]をダウンロードしインストール

バージョン管理されている.NETコンポーネントはバージョンに非常に敏感です。プログラ

注4) http://keepass.info/news/n121004_2.20.1.html

ムは直観的なのでマニュアルがなくても利用できると思います。データベース新規作成の際にキーファイルやWindowsアカウントを指定できますが、キーファイルを指定した場合はそのファイルが必要になります。

Windowsアカウントを指定した場合はそのPCでデータベースを作成したユーザ以外はアクセスできなくなるので注意してください。keepass.exeをC:¥Program files¥GNU¥emacs 23¥binにコピーしてください。

❷KeePassdllを筆者HP[注5]よりダウロードする

zip中のdllをC:¥Program files¥GNU¥emacs 23¥binに、keepass.elをload-pathの通ったディレクトリにコピーしてください。

❸srfi-2.el[注6]をダウンロードしてload-pathの通ったディレクトリに置く

❹初期化ファイルに次を記述する

```
(add-to-list 'after-load-alist
     '(rfc2104
       (require 'keepass)))
(add-to-list 'after-load-alist
     '(imap
       (require 'keepass)))
```

❺~/.authinfoに以下のように記述する

KeePassデータベースはデータベース名でキャッシュします、cram-md5ではGNUSフィールドへのアクセスは不要ですが、Gmail用の設定のためにGNUSフィールドへのアクセスを追加しています。

```
# cram-md5の場合
machine localhost login ユーザ名 password ➘
keepass:KeePassデータベースファイル名 ➘
?Database/eMail/UserName=ユーザ名#gnus ➘
port imap
```

```
# AHTH-LOGINの場合(gnusというカスタム ➘
フィールドにパスワードを保存)
machine imap.gmail.com login ユーザ名@ ➘
gmail.com password keepass:KeePass ➘
データベースファイル名?Database/eMail/ ➘
UserName=ユーザ名#gnus port 993
```

❻~/.gnusにてGNUSの設定を行う

gnus-select-methodまたはgnus-secondary-select-methodsにサーバを追加します。

```
;; gmailの設定例
(setq gnus-select-method
      '(nnimap "識別名" ;; 実際のユーザ名 ➘
は.authoinfoで指定するので、ここでは任意で良い。
        (nnimap-address "imap.gmail.com")
        (nnimap-server-port 993)
        (nnimap-stream ssl)))
```

まとめ

連載の最終回となる今回は.NETコンポーネントの利用方法を紹介しました。Windowsを使用する場合.NETは避けて通れない技術です。GNU Emacs(Win64版)の可能性を示すことができたと思います。

筆者は今後もEmacsを使い続けると思いますし、使い続ける限り環境向上を追求していくことでしょう。本連載は筆者にとっても機能向上の追及のいいきっかけになりました。7回にわたる本連載にお付き合いいただきありがとうございました。

それではまたどこかでお会いできることを楽しみにしています。SD

注5) http://hp.vector.co.jp/authors/VA052357/keepass.html
注6) https://github.com/langmartin/site-lisp/blob/master/srfi-2.el

テキストデータならお手のもの

開眼シェルスクリプト

㈲ユニバーサル・シェル・プログラミング研究所　http://www.usp-lab.com
上田 隆一　UEDA Ryuichi　Twitter @uecinfo

第15回　シェルで画像処理 ――バイナリデータをテキスト化して扱う

はじめに

みなさん、ラーメンのおいしい季節、いかがお過ごしでしょうか。筆者は朝うどん、夜ラーメンという、太く短い人生を歩んでおります。

前々回、バイナリデータをシェルスクリプトでいじるという宣言をしました。何を扱うかは決めていなかったのですが、親しみ深いバイナリデータということで、ラーメンをシェルスクリプトで加工します。……すいません。ラーメンの画像をシェルスクリプトで加工します。

今回やることは、計算量のうえではとても効率が悪い方法です。しかし、画像処理に必ずつきまとう多次元配列やポインタが出てきません。データを端末で見ながらやると、結構簡単にできてしまいます(いや、慣れないと今回の内容は難しいかもしれませんが)。ポインタを気にしない画像処理で遊んでみましょう。

▼図1　今回の題材のラーメン写真(noodle.jpg)

難しいことはわかりやすく、わかりやすいことは面白く、面白いことは深く。――真島昌利

画像をテキストで表現できるppm形式

画像は我々が普段書いているようなテキストファイルと比べてサイズが大きいので、通常はバイナリ形式でファイルにします。ただ基本的には、ピクセル(画素)ごとにR(赤)、G(緑)、B(青)の値を記録したファイルならば、ディスプレイの上に画像として再現できます。

じつはテキストで画像を表現する形式は存在しています。ここでは、portable pixmap format(ppm)形式について紹介します。

ppm形式は、カラー画像の表現形式の1つです。ppm形式にはテキスト(アスキーコード)でデータを持つ形式とバイナリで持つ形式がありますが、本稿ではテキストの形式だけをppm形式として扱います。

説明のため、**図1**のようなJPEGの写真を準備しました。これをppm形式にしてみましょう。

画像の変換には、ImageMagickというツールが便利です。GUIアプリケーションの裏で使われていることもあるので、知らないうちに使っている人は多いかもしれません。環境によってはインストールが面倒だったりするのですが、Webなどで調べてインストールをお願いします。

Ubuntuなら**sudo apt-get install imagemagick**で問題なくインストールされます。この原稿で使ったのは、Ubuntu 12.04です。

インストールしたら、convertという、ちょっとそのネーミングはどうなんだという名前のコマンドでImageMagickの機能が使えるようになります。次のように、オプションの最後の方に入力ファイル名と出力ファイル名を書いて変換します。

```
$ convert -compress none noodle.jpg ↵
noodle.ppm
 ↓ファイルサイズがちょっと大き過ぎか・・・
$ ls -lh noodle.*
-rw-rw-r-- 1 ueda ueda 1.8M 12月 13 11: ↵
11 noodle.jpg
-rw-rw-r-- 1 ueda ueda  83M 12月 13 13: ↵
06 noodle.ppm
```

できたファイルをheadしてみましょう。こんなふうにテキストとして読めたらうまく変換できています。

```
$ head noodle.ppm
P3
2448 3264
255
112 14 13 112 14 13 113 15 14 114 16 15 ↵
112 14 13 111 13 12 111 13 12 111 13 12 ↵
113 15 14 115 17 16 113 15 （略）
```

ppmはこの出力のように、上数行のヘッダ部と、その下から始まる数字の羅列(ボディー部)で構成されます。ヘッダ部は最初の4個の数字で構成され、順に画像の種類(P3はテキストのppmを表す)、幅、高さ、ピクセルの値の最大値を表します。この画像は2448×3264ピクセル、256階調でテキスト形式で保存されているという意味になります。また、#記号があると、行末までコメント扱いされますので、何か処理するときは**sed 's/#.*$//'**などでコメント部分を除去します。

ボディー部には、画像の上の段から順番に、左から右に向かってR、G、Bの順にピクセル値が並びます。よく見ると3個ごとに似た数字が繰り返し出てくることに気づきます。改行とスペースが区切り文字になり、ヘッダも含めて改行はどこに入れてもよいことになっています。

ppm形式はテキストファイルですが、立派な画像ファイルでもあるので、Linuxのデスクトップ環境なら、GUI上でファイルをクリックするとJPEGと同様、画像が閲覧できると思います。

シェルスクリプトで画像処理

では、シェルスクリプトでこの画像をいじってみましょう。ここで扱うことはconvertのオプションで実現できることも多いので、興味がある方はmanを読んでみてください。本稿では、jpgからppmに変換するためにだけImageMagickを使います。

使い慣れたデータ形式にする

まずは、ppmのように数字が延々と並んでいるのは後処理がたいへんですので、次の形式のように5列のデータに変換しましょう。

```
縦の位置 横の位置 Rの値 Gの値 Bの値
・・・
```

コードは**リスト1**のようなものを書きました。コメントがあるとヘッダの行がずれてしまうので、まず7行目でコメントの行を除去し、全部数字を縦に並べた$tmp-ppmを作ります。座標を付けるときに画像の幅が必要ですので、13行目で$tmp-ppmの2行目から幅を取得しています。

15行目以降が、ピクセルの値を並べ直して座標をレコードに付加するコードです。先に**図2**で実行結果を示してから説明します。

まず、リスト1の15行目の**tail -n +5**は「5行目以降を出力」という意味になります。数字に+を付けると、その行数以降という意味になります。

17行目のawkは、読み込んだ数字を横に並べていって、3回に1回改行を入れるという処理です。printは文字列を出力後に改行を入れるので、**print ""**と空文字を出力すると改行の意

味になります。

18行目のawkは、各ピクセルのRGB値に座標を与えています。awkでは、ものの個数はなんでも1から数えます。NRは今扱っているのが何レコード目かという変数ですが、これも1からスタートします。これは直感的でよいのですが、数学的には面倒な処理を生む原因になります。18行目の処理では、0から行数をカウントするnという変数を作り、そこから各ピクセルが上から何行目、左から何列目に位置するかを計算しています。

ところで、この処理は大きな画像で行うと結構時間を食いますので、小さめの画像で試してから大きな画像を処理してみてください。これはシェルスクリプトでやると高速処理はまったく期待できません。ただまあ、スクリプト言語はどの言語もピクセルごとに読み出して処理するのは苦手なようです。

▼リスト1　ppmを正規化するシェルスクリプト（ppm2data）

```
01 #!/bin/bash -xv
02 # ppmを座標と画素値のレコードに変換
03 # written by R. Ueda / Dec. 13, 2012
04 tmp=/tmp/$$
05
06 #数字を1列に並べる
07 sed 's/#.*$//' < /dev/stdin     |
08 tr ' ' '¥n'                     |
09 grep -v "^$"                    |
10 sed 's/  *$//' >  $tmp-ppm
11
12 #幅（ヘッダ2行目の最初の数字）を代入
13 W=$(head -n 2 $tmp-ppm | tail -n 1)
14
15 tail -n +5 $tmp-ppm     |
16 #3個ごとに数字を1レコードにする
17 awk '{printf("%d ",$1);if(NR%3==0){print ""}}'  |
18 awk -v w=$W '{n=NR-1;print int(n/w),n%w,$0}'    |
19 awk '{print sprintf("%04d %04d",$1,$2),$3,$4,$5}'
20
21 rm -f $tmp-*
22 exit 0
```

▼図2　リスト1の実行結果（正規化後のデータ）

```
$ cat noodle.ppm | ./ppm2data > noodle.data
$ head -n 3 noodle.data
0000 0000 112 14 13
0000 0001 112 14 13
0000 0002 113 15 14
$ tail -n 3 noodle.data
3263 2445 199 132 90
3263 2446 198 131 89
3263 2447 199 132 90
↓ppmよりさらに巨大化
$ ls -lh noodle.data
-rw-rw-r-- 1 ueda ueda 158M 12月 14 10:58 noodle.data
```

▼図3　画像の切り抜き操作

```
$ awk '$1>"1000" && $1<"2764"' noodle.data > tmp
$ H=$(awk '{print $1}' tmp | uniq | wc -l)
$ W=$(awk '{print $2}' tmp | tail -n 1 | sed ➥
's/^00*//' | awk '{print $1+1}')
$ awk '{print $3,$4,$5}' tmp > body
$ echo P3 > header
$ echo $W $H >> header
$ echo 255 >> header
$ cat header body > hoge.ppm
```

画像を切り出す

さて、ここからは図2で作ったnoodle.dataを使って画像にいたずらしてみましょう。まずは基本として、画像の一部分を切り出してみましょう。シェルスクリプトでもよいのですが、あえて雑技団的な雰囲気を出すために**図3**のように端末でやってみました。画像の上から約1000ピクセル、下から300ピクセル分を削る処理です。

noodle.dataは第1列が縦の座標ですので、1行目で1001ピクセル目からのピクセルが抽出できます。2、3行目は画像の高さと幅を計算してそれぞれファイルに保存しています。なぜこうなるかは考えてみてください。5行目で画像のボディー部を作ります。座標を取り除けばそのままppmのデータとして使えます。あとはヘッダを1行ずつ書いていって、拡張子がppmのファイルに保存して一丁上がりです。

筆者の環境では、ファイルをクリックすると、**図4**のように画像を見ることができます……お腹がすいてきました。

見られない人は、次のようにconvertでjpgか何かに変換しましょう。

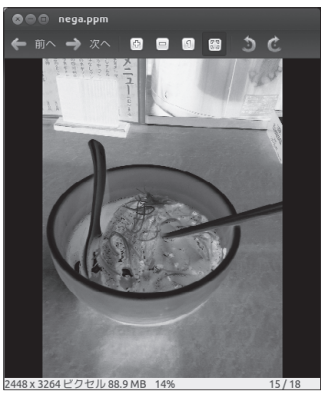
```
$ convert hoge.ppm hoge.jpg
```

ネガを作る

次に、色を反転させてみましょう。これは簡単で、RGB値それぞれを反転させればよいということになります。操作を**図5**に示します。これを実行すると、**図6**のような画像になります。カラーじゃないのが残念ですが、今度は淡青色スープにウミウシのようなチャーシューと黒髪のような白髪ネギを搭載した、たいへん食欲をなくすラーメン画像になります。

▼図4　図3実行後の画像（切り抜き画像）

▼図6　図5実行後の画像（色反転画像）



▼図5　色の反転操作

```
$ cat noodle.data | awk '{print 255-$3,255-$4,255-$5}' > body
↓もとのヘッダを付ける
$ head -n 3 hoge.ppm | cat - body > nega.ppm
```

画像を合成

次は、ラーメン画像に別の画像を合成してみましょう。偶然（嘘）、筆者の画像ディレクトリに、noodle.ppmと同じ大きさの図7のような画像curry.ppmがありました[注1]。これを**図8**のように処理します。

loopjはOpen usp Tukubaiのコマンドで、**図9**のように2つ以上のファイルの各レコードについて、キーが同じレコードを連結します。**num=1**は左から1列をキーするという意味です。キーはソートされている必要があり、あるファイルにあるキーのレコードがないと、0でパディングされます。

ですので、図8の2行目は画素の位置（左から2列）をキーにして、noodle.dataとcurry.dataを連結しているという意味になります。**図10**にtmpの最初の部分を示します。

注1） 素直に「ラーメン」としないのは性格上の問題です。

▼図7　合成する画像

▼図9　キーでファイルを連結するloopj

```
$ cat file1
001 aaa 123
003 bbb 234
$ cat file2
001 AAA
002 BBB
004 CCC
$ loopj num=1 file1 file2
001 aaa 123 AAA
002 0 0 BBB
003 bbb 234 0
004 0 0 CCC
```

▼図10　loopj実行後のデータ

```
$ head -n 3 tmp
0000 0000 112 14 13 255 255 255
0000 0001 112 14 13 255 255 255
0000 0002 113 15 14 255 255 255
```

▼図8　文字を画像に重ねる操作

```
$ cat curry.ppm | ./ppm2data > curry.data
$ loopj num=2 noodle.data curry.data > tmp
$ cat tmp | awk '{print $3*$6/255,$4*$7/255,$5*$8/255}' | sed 's/¥.[0-9]*//g' > body
$ head -n 3 noodle.ppm | cat - body > curry_noodle.ppm
```

3行目はnoodleとcurryのピクセルを比較して、curryの字のない部分(RGBそれぞれ値が255)については、noodleの値、字のある部分については画素が黒くなる演算をしています。たとえば**$3*$6/255**は$6=255なら答えは$3の値になるし、$6=0なら答えは0になります。3行目のsedは演算結果の小数点部分を削除する働きをします。この計算のような文字列処理が入るのは、シェルを操作しておもしろいことの1つです。最後、4行目でヘッダを付けて**図11**のような画像の完成です。

モザイクをかける

最後は、もうちょっと難しいことをしてみましょう。ブームに乗ってラーメンにモザイクをかけてみます[注2]。

まず、ラーメンの画像を10×10ピクセルごとに区切ってタイルにします。**図12**の操作でnoodle.dataの座標からグループのコードを作ります。tailの出力のように、たとえば(3263,2445)はタイル(32,24)ということを、各レコードの後ろに付加しておきます。

次に**図13**のように、各タイルの画素値を平均します。このデータがモザイクのレイヤになります。このコードでは、まずタイルの番号を左側に持ってきてキーにして、ソートし、Tukubaiコマンドのsm2で足し込んでいます。sm2のあとの出力は**図14**のようになります。

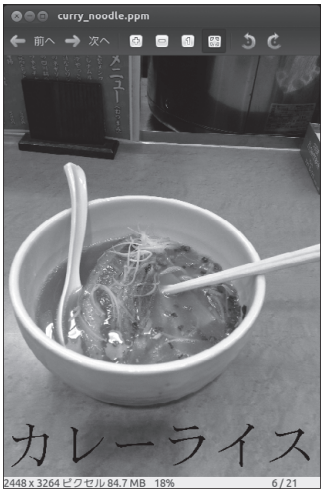
図13中の**sm2 +count 1 2 3 5**は、1、2列目をキーにして、キーごとに3〜5列目を足し込むという意味になります。**+count**を付けると、足し込むときにキーの数を数えておき、レコードの出力の際にキーの横に数を付加します。ですので、この出力の4、5、6列目を3列目で割ると、各グループの平均のRGB値になります。

sort -k1,2 -sの**-s**ですが、これはソートキーが同じレコードの順番を変えない「安定ソート」のオプションです。この処理では安定ソートは不要ですが、sortコマンドは安定ソートのほうが処理が早く終わるので経験的に付けています。

meanのレコードの一部を**図15**に示します。この部分の処理は元の画像が大きかったのでopen

注2) Nudiferで検索をしてみてください。

▼図11　図8実行後の画像(図7を合成した画像)



▼図12　座標をタイル化

```
$ awk '{print $0,substr($1,1,2),substr($2,1,2)}' noodle.data > tran
$ tail -n 3 tran
3263 2445 199 132 90 32 24
3263 2446 198 131 89 32 24
3263 2447 199 132 90 32 24
```

▼図13　モザイクをかけたときの画素値を計算

```
$ awk '{print $6,$7,$3,$4,$5}' tran | sort -k1,2 -s | sm2 +count 1 2 3 5 | awk '{print
$1,$2,$4/$3,$5/$3,$6/$3}' | sed 's/¥.[0-9]*//g' > mean
```

▼図14　sm2実行後

```
$ awk '{print $6,$7,$3,$4,$5}' tran | sort -k1,2 -s | sm2 +count 1 2 3 5 | head -n 3
00 00 10000 1096186 274854 214869
00 01 10000 1049205 268678 207120
00 02 10000 1048624 266316 212040
```

版のsm2だと10分程度、pypy[注3]でコマンドを高速化しても3分くらいかかってしまいました。awkでこの計算をすると、もっと速く処理できます。

モザイクのレイヤのRGB値が計算できたら、先ほど作ったtranファイルにmeanファイルを連結します。**図16**のようになります。

図16では、cjoin1というTukubaiコマンドを使いました。このコマンドはtranの第6、7列目のデータとmeanの左2列を比較して、meanの内容をtranに連結します。join1というコマンドもあるのですが、こちらはtran側が6、7列目でソートしていないと使えません。マスタ扱いされるmeanのほうは、cjoin1でもキーでソートされている必要があります。

delfは指定した列を消すコマンドで、すでに不要なタイル番号を消去しています。

tmpが作成できたら、もう少しです。**図17**のようにコマンドを打ちます。読むのがたいへんですが、要は画像の範囲指定をして、範囲内ならモザイクのRGB値、範囲外なら元の画像のRGB値を出力しているだけです。実行後の画像は**図18**のようになります。

注3） http://www.pypy.org/

▼図18　モザイクをかけた画像

▼図15　タイルの画素値

```
$ tail -n 3 mean
32 22 194 132 78
32 23 200 137 86
32 24 200 138 89
```

▼図16　画像の画素値とタイルの画素値を連結

```
$ cjoin1 key=6/7 mean tran | delf 6 7 > tmp
$ tail -n 3 tmp
3263 2445 199 132 90 200 138 89
3263 2446 198 131 89 200 138 89
3263 2447 199 132 90 200 138 89
↑座標、もとのRGB値、モザイクのRGB値
```

▼図17　領域を指定してタイルの画素値を反映

```
$ awk '{if($1>=1000&&$1<=2400&&$2>=100&&$2<=2000){print $6,$7,$8}else{print
$3,$4,$5}}' tmp > body
$ head -n 3 noodle.ppm | cat - body > moz.ppm
```

終わりに

今回はシェルでバイナリデータを扱うということで、画像処理をやってみました。しかし、よくよく考えてみると、バイナリデータを最初にImageMagickでテキストにしてしまったので、バイナリだからどうという処理は出てきませんでした。結局、相互に変換する道具さえあればよいということで、両者に本質的な違いはなく、シェルスクリプトで行うようなテキスト処理に落とし込むことができます。

ただし、JPEGのように圧縮効率のよいデータの形式と、テキストのようにベタなデータでは、サイズに100倍近い違いがありました。テキストを圧縮してもJPEGにはサイズの点でかないません。

一方で、catやgrepで中身が見えないJPEGに、EXIF情報[注4]のようなものが保存されてしまうと、いろいろ問題が起こりがちです。もしかしたら、テキストファイルで画像を持つことが普通になる日が来るかもしれません。

次回は、もうちょっと本格的な画像処理をやってみたいと考えています。SD

注4） 撮影日時やカメラの機種などの写真用のメタデータ。

第35回

少ないダウンロード数で大きく稼ぐアプリ開発術

スマートフォンの認知度を一般に広めただけでなく、ソフトウェア開発においても新しい波を作り出してしまったiOS。開発者たちは何を見、どう考えているのか。毎回入れ替わりでiOS向けアプリケーション開発に関わるエンジニアに登場いただき、企画・開発のノウハウやアプリの使いこなし術などを披露してもらいます。

藤田 武男　FUJITA Takeo
あぷまがどっとねっと　http://appmaga.net/

アプリでメシを喰う最も無名な開発者

2009年12月、iPhoneに魅せられ、気がつけばサラリーマンを辞めていました。当時の筆者には「アプリ作ってメシを喰う、家族を養っていくぞ！」という大きな夢はあったのですが、残念なことにプログラミング経験が少しもありませんでした。

それでも何とかなるさと、型落ちで安くなっていた白MacBookを購入し、参考書を片手にコードをガリガリ書いていたのですが、貯金はジョジョに減っていき、半年後には、その日のメシを喰うために運送業で働く日々が始まります。人生初の肉体労働に開発どころか白MacBookに触る気力もないまま時が流れ、MacBookが黄色く色づき始めたころには「このまま運送業でもいいかな」とそんな甘えが心の中で芽生えてきました……。

そんなある日、娘が「もうアプリ作らないの？」と言ってきました。娘にとっては「アプリを作る＝家で仕事をする＝いっぱい遊んでもらえる」という思考なのでしょう。筆者はこの言葉で「いったい何のためにサラリーマンを辞めたのか」。そんな思いがふつふつと湧いてきまして、昼は配達、夜は開発という2足のワラジを履いた生活を始めました。

そこからさらに半年後、「3000000RPG」[注1]というゲームアプリがヒットし、やっと食べていけそうな収入を得ることができ、運送業を卒業。昨年2012年はそのアプリを中心に何とか自作アプリ開発だけで家族を養うことができ、娘ともたくさん遊ぶことができました。

ヒットしたと言いましても、「ああ、あのアプリの作者なの!?」と言われるほど大ヒットしたわけではないですし、そもそも筆者のアプリは基本無料のものばかりです。なのになぜアプリ開発で喰っていけたのか？　筆者なりの「少ないダウンロード数でも大きく稼ぐアプリ開発術」を、この場をお借りしてお伝えします。何かの参考になれば幸いです。

アプリを作って収入を得る方法

大きく分けて次に示す3つがあります。

有料（売り切りモデル）

85円や170円などの小額でアプリを販売する方法です。Apple社には手数料として売上の30％を取られますので、85円の有料アプリが1本ダウンロードされれば60円程度、開発者の懐に入ることになります。

このモデルの例として、大手ゲームメーカー

注1）URL https://itunes.apple.com/jp/app/3000000rpg/id468079378?mt=8

の「有名大作RPG」、ある一定のダウンロード数が見込める「人気パチスロシミュレーター」などがあり、アプリでは比較的高いとされる1,000円以上の価格で売られています。また個人開発者なら便利なツール系のアプリを薄利多売で売り出し、成功している方もいますが、昨今の「ゲームは無料で遊ぶもの」というイメージのおかげで、ゲーム系個人開発者にはなかなか厳しい市場となっております。ちなみにアプリの価格はApple社があらかじめ段階的に決めており、たとえば10円や980円など開発者が自由に決められるわけではありません。また価格はリリース後も変更することが可能です。筆者も2本、有料アプリをリリースしていますが(85円と170円)、日ごとの売上報告に「売上：1」と記載されていると逆にビックリしてしまうくらい、ダウンロードされていません。

基本無料+課金

アプリは無料でダウンロードでき、基本最後まで遊ぶことは可能ですが、「課金」することで、たとえばゲームを有利に進めることができます。いわゆる「ソーシャルゲーム」と呼ばれるジャンルが、このモデルの代表例として挙げられます。また最近では「序盤まで無料」または「機能を絞って無料」、その先を遊びたければ、またはフル機能を使いたければ課金するという、感覚的には体験版で試遊し、そのままお金を払って使い続けてもらう方法も、一部メーカーやツール系個人開発者が導入しております。いきなり有料で買うよりは、ユーザとしても「お金を払う」という敷居が低いようです。

参考までに筆者の例を挙げておきますと、3000000RPGにも課金することによりゲームを有利に進める「ゲーム内通貨」機能を実装しています(85円、170円、250円の3段階の価格設定。価格が高いほどゲーム内通貨量が増える)。3000000RPGではピーク時、課金だけで1ヵ月35万円の収入を得ることができました。今でも1日500円～1,000円分くらいはほぼ毎日課金していただいております。一番の売れ筋は意外にも250円分です。これはあまり深く考えずに価格を設定したために、250円分がお得すぎるという点もありますが、逆に言えば気に入ってもらえれば250円という金額でもユーザはちゃんと払ってくれるのです。もし、3000000RPGが250円という有料モデルで販売されていたとしら、1本もダウンロードしてもらえなかったことでしょう。

無料+広告

無料でずっと使ったり遊んだりすることが可能ですが、広告バナーが画面のどこかに表示されます。ユーザが広告を見たり、タップしたり、広告されている商品を実際に購入したりすることにより、開発者は広告収入を得ることができます。前述のように、今のアプリ市場では「有料」で販売することがなかなか難しいため、ほとんどのゲーム系個人開発者はこのモデルを利用し収入を得ています。筆者もほぼこの方法でメシを喰っていますので、今回は、広告をクリックすることにより収入が発生するモデルを中心に、お話しします。

広告収入を得るためには

開発者自身が広告を出稿してくれる広告主を捜すのは稀なケースで、ほとんどの場合、開発者の代わりに広告主を集め、アプリ内に自動的にその広告を配信してくれる「アドネットワークサービス事業会社」(以下、広告業者)と契約します(**表1**)。契約と言うと大ごとのようです

▼表1　主なアドネットワークサービスの名称とURL

名称	URL
AdLantis	http://www.adlantis.jp/
Admob	http://www.google.co.jp/ads/admob/
AMoAd	http://www.amoad.com/
i-mobile	http://i-mobile.co.jp/
mediba	http://www.mediba.jp/
nend	http://nend.net/

が、たとえば無料メールサービスにユーザ登録する感覚で簡単に行うことができます(一部、身分証明書のコピーなどを求められる場合あり)。

契約後は、アプリを開発するたびにそのアプリ固有の広告IDなどを発行してもらい、各広告業者の自動広告配信システムを自作アプリに組み込み、Apple社の審査(約1週間)を経て、アプリが世にリリースされます。契約時も広告IDなどの発行時もほとんどの場合、広告業者による審査がありますが、よほどのことがない限りスムーズに契約し、承認発行してもらえます。もちろん複数の広告業者と契約することが可能ですので、最近では1つのアプリに対して、複数の広告業者の広告配信システムを導入するというのが主流です。このあたりについては、後ほど詳しく説明します。

余談ですが、Apple社の審査は「アダルト」に関してとくに厳しく、アプリそのものに「ポルノ」性はなくても、表示されている広告がそれを連想させるような場合、審査に落とされることがあります。どのような種類の広告を表示させるかを開発者自身がある程度設定で決められる広告業者もありますので、できればそのようなシステムがあるところと契約したほうが、Apple社の審査時に無駄な心配をしなくて済みます。

とあるアプリの広告収入

筆者の唯一のヒット作「3000000RPG」は、リリースから1ヵ月で5万5,000ダウンロードを記録しました。そのときに稼ぎ出した広告収入は**図1**のとおりです。

どの広告システムも同じように運用していたにも関わらず大きな差が出てしまいました。なぜこのような差が出てしまったのかは、後ほど詳しく解説します。

また、**図2**は、3000000RPGが2012年の1年で、広告業者1社から得た収入のデータです。

単純に1年分の全報酬を12で割ると、1ヵ月平均約12万円の収入を1本のアプリで得ていたことになります。さすがにこの収入で家族を養っていくのは困難ですが、お小遣いと割り切ればなかなかの数字ではないでしょうか。

図1のときは、リリース直後ということもあり、1ヵ月で約5万5,000ダウンロードされていますが、図2のときはダウンロード数もすっかり落ち着き、1年で合計6万ダウンロード、最近では1日50〜100くらいです。アプリはテレビなどで紹介されるなど特殊な要因がない限りは、リリース直後から2週間後までが一番売れる時期であり、その後は急激に売上が下がります。アプリをリリースしたことがある方ならご存じでしょうが、1日50〜100というダウンロード数は、売上的にかなり厳しい数字です。にも関わらずなぜ月々12万円の収入が得られるのか？ そのために必要な3つの要点を紹介します。

少ないダウンロード数で食べていくためにやるべき3つのこと

広告をつねに表示させておく

アプリに広告業者から提供される「自動広告配信システム」を組み込んでも、肝心の「広告そのもの」がなくなってしまうと、もちろん広告が表示されません。これは広告主が決めた予算に達したため、その広告が表示されなくなってしまったからです。その場合、別の広告主のものが次から次へと表示されれば何ら問題はない

▼図1　リリース後30日分の広告収入の比較

A社：66万4,000円
B社：8万円
C社：5,000円

▼図2　広告に関する1年分のデータ

広告表示回数(インプレッション数)：2,194万3,653回
広告クリック数：14万3,821回
報酬額：144万6,337円

のですが、取り扱っている広告の量は広告業者に依存します。

ここで先ほどの図1「リリース後30日分の広告収入の比較」を再度確認してみましょう。A社からは約66万4,000円、B社からは約8万円、C社からは約5,000円の広告収入がありました。アプリに組み込んだ広告配信システムが、各社のサーバに「広告を表示してくれ」という信号を送ったところ、A社はほぼ100%広告を返しましたが、B社は5%、C社は1%にも満たない数字になりました。つまり広告がなかったのです。

B社はともかく、ほぼ収入が得られなかったに等しいC社の枠のところに、素直にA社の広告を用意しておけば、「捕らぬタヌキの皮算用」ですが単純計算で2倍の収入が得られたことになります。悔やんでも悔やみ切れません。いまだに夢でうなされます。

当時はあまり広告収入に関しての情報がなく、また筆者自身そんなにダウンロードされるとも思わなかったので、あまり深く考えずに広告業者を選んでしまったのが大失敗でした。今ではアプリ開発者同士が、Twitterやブログなどでそのあたりの情報を交換し合っていますので、ぜひ参考にしてください。

また、筆者はダウンロード数の99%が日本ですのであまり意識していませんが、グローバルなヒットを狙いたい場合は、「国内のみ」「海外もOK」と広告業者により違いが出るため、自分がどの国で勝負するかによっても選択肢が決まってきます。広告配信システムの対応という点では、最近、アプリの開発環境もさまざまものが出てきており(cocos2d、Unity、coronaなど)、広告業者が提供するシステムがその開発環境に対応するかなどにも注意が必要です。このあたりも、その開発環境を利用している方がブログや勉強会などで情報を公開してくれていますので、該当する場合は必ずチェックしてください。

すべての条件を満たしている広告業者を見つけるのはなかなか難しく、最近では前述のとおり、1つのアプリに複数の広告業者の配信システムを導入するのが主流となっています。アプリ利用者が日本ならA社、海外ならB社の広告を表示させる、C社の広告が表示されなくなったら自動的にD社の広告を表示させるといったプログラムを組んでいる方もいます。また、そのようなシステムを簡単に導入できるツールなども公開されています。

広告のタップ単価を検討する

広告をつねに表示させる準備ができたら、次は「クリック単価」を意識します。基本的なクリック単価はやはり広告業者に依存します。クリック単価は0円から30円まで幅広く存在します(**表2**)。

表2は、3000000RPGの2012年1月と2月の業者別クリック単価です。前述のとおり、C社の広告がほぼ表示されないのでどこか他に良い広告業者がいないか情報を探していると、「当社と契約してくれ！」という営業メールがいろいろ届き始めました。ランキングの上位に入ると、各広告業者からたくさん連絡がくるようになります。どこがよいかなどは実際に使ってみないとわからないだろうということで、7社の広告会社と契約し試してみました。

1円から10円まで幅広く存在するのが見てもらえると思います。この結果を受け、筆者は単価が高く、表示率も申し分ないD社とE社に絞ることにしました。

ここからは余談です。ある日、ほぼ同じタイミングでD社とE社から「広告の表示回数を増やしてほしい」という営業メールがきました。E社は単純に「当社の広告表示回数を増やすこ

▼表2　クリック単価の比較

業者	2012年1月	2012年2月
A社	5.5円	5.2円
B社	システム的に単価のみの計算不可能	
C社	6.1円	1.7円
D社	10円	9.8円
E社	9円	8円
F社	6.3円	4.5円
G社	7.6円	8.3円

とは可能ですか？」という内容。対してD社は「あなたE社の広告使ってますよね!? その枠を当社にしてもらえませんか！ 御礼にクリック単価の高い広告を優先的に表示させますよ！」という、より具体的な提案でした。皆さんならどう判断されたかはわかりませんが、筆者はE社の枠にD社の広告を表示することにしました。その効果かどうかはわかりませんが、一時期は20円という高単価の広告収入を得られましたし、今でも平均的に10円以上という、クリック単価では高単価を維持してくれています。

さらなる余談ですが、その後E社の単価は見事に落ち込み、単価2円ほどになっていたので、間違った判断をしていたらとっくに別の仕事をしていたと思います。

話がそれてしまいました。10円玉1枚と5円玉1枚のどちらがほしいと言われれば、「面倒なのでどっちでもいい」と答えがちです。しかし、「10円玉を1万枚」と「5円玉を1万枚」のどちらがほしいか、10万枚ならどうか、と問われれば、ほとんどの方が10円玉を選ぶと思います。クリック単価は日々の積み重ねです。1時間ごとにクリック単価をチェックし、より良い条件の広告業者の広告を優先的に表示させたり、営業に直接電話して他社の単価を材料に交渉したりする開発者もいます。筆者はそこまでの努力はしていませんが、チリも積もれば山となるのです。3000000RPGに初めからD社の広告を表示させておけば、もっともっとおいしいゴハンが食べられたのです。

広告の画面内の配置を考える

広告はほとんどのアプリで画面の上下いずれかに寄せて表示されます。実は画面の上にあるか、下にあるかで、クリック単価が変わります（**図3**）。

画面の上のほうにあるとなかなか広告をタップするのが難しく、逆に言うとそこまでしてタップしてくれるならよほどその広告に興味がある、広告主にとっては「見込み客」ということになります。そのため、一般的には画面の下よりも上のほうが、広告単価は高いようです。筆者自身はその恩恵をあまり受けていませんが、今ランキングチャートをにぎわしているゲーム開発者のほとんどが「上」に配置しているのを見ると、そちらのほうが収益が良いようです。

広告を上に配置すると、「ゲームを遊んでいる間に間違ってタップすることがなくなる」、いわゆる「誤クリック」「ミスタップ」を防ぐという意味でも、高単価になる理由の1つとなります。筆者は開発の他にアプリ紹介サイトも運営しており、ミスタップをわざと狙っているような配置のアプリもときおり見られます。そのような配置は広告主にとってはまったくうまみがないので、広告単価の低下、最悪の場合には広告配信の停止という措置がとられます。

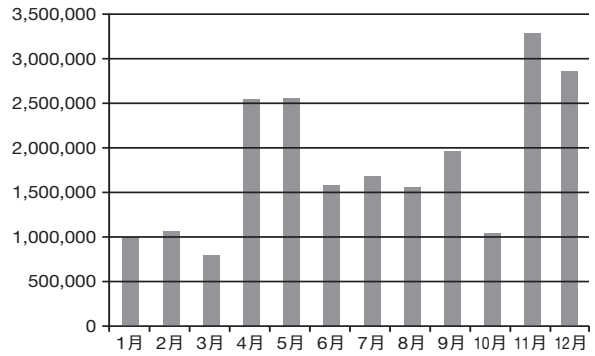
筆者も駆け出しのころ、たくさんクリックしてもらったほうが稼げると考えたあまりに、クリック率約50%というとんでもなくひどい広告配置のゲームアプリをリリースし、翌日に広告業者から広告停止勧告があったことを覚えています。筆者の場合は勧告がありましたが、業者によっては即停止というところもあります。

誤クリック狙いは広告の停止だけで

▼図3　広告を上（左側）と下（右側）のどちらに表示するか

▼図4 「3000000RPG」の2012年の広告表示数の推移

月	インプレッション数
1月	983,270
2月	1,064,605
3月	801,124
4月	2,556,491
5月	2,555,847
6月	1,577,876
7月	1,678,444
8月	1,569,700
9月	1,969,310
10月	1,037,524
11月	3,285,550
12月	2,863,912



※リリース後トータルで14回のバージョンアップ(バグ修正含む)を行い、うち4月と11月は追加要素を用意した。

はなく、ユーザにとっても楽しく遊べない大きな原因となり、長く遊んでもらえず、そしてストアのレビューに低い点数をつけられて新規ユーザも獲得できないという、負のスパイラルに落ち込みます。たくさんタップしてほしいという気持ちはわからなくはないですが、そのような配置は避けましょう。

広告業者によっては、誤クリックに関してかなり厳しく判断するところもあります。たとえば子ども向けのアプリの場合、子どもはイロイロ触るため、どうしても広告のクリック率が上がってしまいます。その数字のみを判断され、広告を停止された開発者もいます。そのような場合、広告はタイトル画面だけにし、ゲームプレイ画面中には広告を表示させないなどの配慮もときには必要です。

最後に

広告関係以外で、できればやっておきたいこととして「定期的なバージョンアップ」があります。それを行うことにより「プレイを止めてしまったユーザ」へメッセージを送り、戻ってきてもらえるキッカケを生み出すことができます。**図4**を見てもらえれば効果的なバージョンアップを行うと、広告表示回数が飛躍的に伸びる、つまりユーザ数が戻る、結果として広告収入が安定するというウハウハの方程式が見ていただけると思います。

本連載では異色の記事になってしまい、皆さんのお口にあうかかなり不安なのですが、何かの参考になれば幸いです。インターネットを検索すれば「2ヵ月で700万円！」「3週間で1千万円！」という景気のイイ話も聞こえてきますが、筆者くらいの数字のほうが身近に感じてもらえる気がして、今回このような記事を書かせていただきました。もし、アプリでいくら稼げるのかなど興味がありましたら、筆者が運営している「あぷまがどっとねっと」[注2]というサイトにて、毎月筆者のアプリのダウンロード数と広告収入を公開しています。こちらもあわせてよろしくお願いいたします。最後にお金の話らしく、ちゃっかり宣伝させていただきました。SD

注2) URL http://appmaga.net/

● 藤田 武男(ふじた たけお) あぷまがどっとねっと

「Takeo Fujita」名義でこれまで50本近くのアプリをリリース。プログラムもデザインも何もできないプランナーとして2年ほどゲーム会社勤務経験あり。現在はアプリ開発よりもiPhoneアプリ情報サイト「あぷまがどっとねっと」の運営に全力投球。Twitter @appmaga_info

第35回 JUnitによるテスト自動化の基礎を知っておこう

モバイルデバイス初のオープンソースプラットフォームとして、エンジニアから高い関心を集めるGoogle Android。いち早くそのノウハウを蓄積したAndroidエンジニアたちが展開するテクニックや情報を参考にして、大きく開かれたAndroidの世界へ踏みだそう！

日本Androidの会 神戸支部
高木 基成　TAKAGI Motoshige
趙 文来　CHOU Bunrai

はじめに

世の中は、ソフトウェアにより人々の作業が自動化され、便利になっていると言えます。しかし、ソフトウェアの開発自体は人が地道に作業しなければなりません。とくにテストは品質を担保する重要な作業にも関わらず、単調な部分が多い領域です。この分野の負荷を軽減する方法としてツールによるテスト自動化があります。今回はAndroidアプリの開発におけるテスト自動化について紹介します。

テスト自動化ツール

テスト自動化を支援するツールはいくつかあります。その中から無償で公開されているものを**表1**にまとめています。現在、Androidには開発工程のさまざまなフェーズにおいて、テストを自動化する方法が検討されています。

本稿執筆時における状況を分析すると、主力となるソフトウェアが選ばれつつあり、整理が進んでいるように思えます。たとえば、非常に注目されていたNativeDriverは、従来のメンバーによる開発が停止したようです。また、AndroidMockは、mockitoがAndroidに対応したことやEasyMockがAndroid対応を予定している[注1]ことを理由に開発を停止するようです。

テスト自動化ツールを利用する開発者としては、なるべく長く使えるソフトウェアを選びた

注1）URL http://jira.codehaus.org/browse/EASYMOCK-108

▼表1　テスト自動化ツール一覧

製品	概要	更新日（執筆時点）	URL
Robotium	ユーザシナリオに沿ったUIテストを中心に自動化できる	2012年11月	http://code.google.com/p/robotium/
scirocco	Robotiumを拡張し、テストレポートの集約や画面のキャプチャを自動化できる	2011年12月	http://code.google.com/p/scirocco/
NatvieDriver	WebDriverと呼ばれる自動化ツールのAPIに準拠してAndroid、iOS、WindowsのUI操作を自動化できる	2011年 8月	http://code.google.com/p/nativedriver/
Robolectric	エミュレータや実機でのテストではなく、開発機のJava仮想マシン上で高速にテストを実行できる	2011年 2月	http://pivotal.github.com/robolectric/
Android Mock	EasyMockと呼ばれるソフトウェアをベースにしており、モックオブジェクトを使い外部連携を簡素化してテストを自動化できる	2012年 5月	http://code.google.com/p/android-mock/
mockito	Android Mockと同じ領域をカバーしており、よりシンプルにテストコードを実装できる	2012年10月	http://code.google.com/p/mockito/
MonkeyTalk	オープンソースで無料で利用できるが、商用サポートが用意されており、iOS、Android、HTML5などのテストを自動化できる	2012年12月	https://www.gorillalogic.com/monkeytalk
monkeyrunner	Androidの開発環境に同梱されており、Pythonを使ってUI操作や画面のキャプチャを自動化できる	—	http://developer.android.com/tools/help/monkeyrunner_concepts.html

いと思います。したがって、採用しようと思っているプロジェクトの活動状況は注意深く見ておきましょう。

今回は、Androidの開発環境に始めから用意されており、開発が停止されるリスクが少ないJUnitを紹介します。JUnitは個々のソフトウェア部品のユニットテストを対象としていますが、実装を工夫すればさまざまな用途に利用できるため、基礎技術として学んでおいて損はないはずです。では、趙氏にバトンタッチをしてJUnitの具体的な使い方を解説してもらいます。テスト自動化の基礎を一緒に学んでいきましょう。

テストプログラムの開発

ここでは、JUnitの使い方を学ぶにあたって、簡単なサンプルをテスト対象として、自動化の実装方法を解説します。始めに、簡単にAndroid関連のJUnitアーキテクチャを紹介します。続いて、サンプルの動作とJUnitのテストコードを解説します。

AndroidのJUnit

Androidのテストフレームワークは、アプリケーション(以下、アプリ)の各機能をテストできるように設計されており、いくつかのツールを提供しています。JUnitのAPIを基本とし、Instrumentation[注2]フレームワークとAndroid専用のテストクラスによって拡張されています(**図1**)。単純にJavaオブジェクトのユニットテストを実行したい場合、JUnit本来のTestCaseクラスを利用します。Activityなどの Android固有の部品のテストには、拡張されたTestCaseクラスを使うことになります。テスト対象アプリもテストアプリもAndroid端末上で動きます。

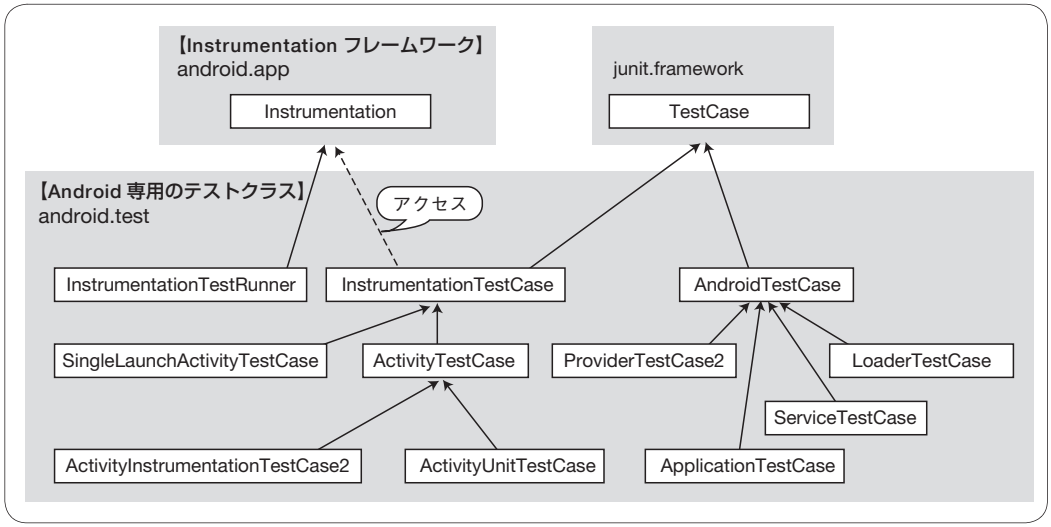
続いて、Instrumentationフレームワークと Android専用のテストクラスの概要を簡単に紹介します。いずれもandroid.testパッケージに含まれるクラスです。

■AndroidTestCaseクラス

Androidアプリのリソースにアクセスする必要がある場合や、マニフェストファイルのパーミッションをテストするときに利用できます。モックのContextを挿入することも可能です。キーイベントの送信などはできません。

注2) AndroidにおけるInstrumentationは、システムとアプリのコミュニケーションをモニタできます。また、"横取り"の制御もできます。Instrumentationを有効にすることで、通常のライフサイクルのAndroidコンポーネント(Activityなど)を独立させて制御できます。

▼図1 Androidテストフレームワークの API



■ApplicationTestCaseクラス

アプリケーションオブジェクトの前処理、後処理のテストを実行できます。マニフェストファイルの<application>要素が正しくセットアップされているかどうかを検証する際に役立ちます。

■ProviderTestCase2クラス

ContentProviderをテストするための基底クラスです。モックのContextを表すIsolatedContextクラスとモックのContentResolverを表すMockContentResolverクラス(android.test.mockパッケージ)を利用することにより、独立した環境下でテストを実行できます。

■ServiceTestCaseクラス

Serviceをテストするためのクラスです。MockApplicationクラスとMockContextクラス(いずれもandroid.test.mockパッケージ)の利用が可能です。getServiceメソッドで対象のServiceを取得する前に、startServiceやbindServiceといったメソッドを呼び出して、Serviceのライフサイクルを開始する必要があります。

■InstrumentationTestCaseクラス

テストを実装する中でInstrumentationを使用したい場合、InstrumentationTestCaseクラスまたはそのサブクラスを使用する必要があります。Activity用のテストケースクラスはこの基底クラスを拡張しています。

■ActivityTestCaseクラス

Activityをテストするための基底クラスです。サブクラスにはActivityUnitTestCaseクラスとActivityInstrumentationTestCase2クラスの2つがあります。

■ActivityUnitTestCaseクラス

独立した単一のActivityをテストするためのクラスです。Activityを開始する前に、モックのContextまたはApplication(あるいはその両方)を挿入できます。

■ActivityInstrumentationTestCase2クラス

API Level 3で非推奨となったActivityInstrumentationTestCaseクラスの新しいバージョンです。複数のActivityの機能テストを行う目的で設計されています。モックオブジェクトの挿入はできません。getActivityメソッドを実行すると、テスト対象のActivityのライフサイクルが開始します。

■SingleLaunchActivityTestCaseクラス

1つの環境で複数のテストに対応し、その環境を変えたくない状態で単一のActivityをテストするのに便利なクラスです。ただし、setUpメソッドとtearDownメソッドの呼び出しは、メソッドごとではなく、1回だけになります。モックオブジェクトの挿入はできません。

テストケース作成の流れ

テストケースは通常のJUnitと同じように、次の流れで開発します。

①用途に合わせたクラスを選んで継承する
②通常のJUnitと同じように、テストデータの配置などの前処理をsetUpメソッドに実装し、不要なログファイルの削除などの後処理をtearDownメソッドに実装する
③テストしたい内容をtestではじまるメソッド名で実装。ユーザの操作などを実装し、入力された内容やファイルの内容を取得してassertメソッドを使って結果を確認する

テスト対象アプリの概要

それでは、早速テストケースの実装に入りましょう。まずは、テスト対象のサンプルを紹介します。今回のサンプルは、簡単なユーザ登録処理を行います。

- 「登録情報入力」画面(図2)でユーザ情報を入

力し、「登録」ボタンを押すと、「登録情報確認」画面に遷移する。入力エラーが発生した場合、エラーメッセージが表示される
- 「登録情報確認」画面(図3)に登録したユーザ情報が表示される

テストケースの解説

本稿で解説するサンプルは、本誌Webサイトの「本書のサポートページ[注3]」からダウンロードできます。

サンプルには2つのプロジェクトが含まれています。「Sample」がテスト対象プロジェクト、「SampleJunitTest」がテストプロジェクトです。これらのプロジェクトをEclipseにインポートしてください。SampleJunitTestプロジェクトを実行すると、Sampleプロジェクトに対するテストが自動的に実行されます。

SampleJunitTestは、JUnitのテストケースを実装したプロジェクトです。SampleJunitTestプロジェクトでは、ActivityInstrumentationTestCase2クラスを利用した画面の機能テストとInstrumentationTestCaseクラスを利用した結合テストをそれぞれ1ケースだけ実装しています。結合テストのシナリオは、次の流れで実施しています。なお、今回は、Android 4.0.3(API Level 15)を指定したエミュレータで稼働を確認しています。

①「登録情報入力」画面を起動する
②次のユーザ情報を入力する
- [ユーザID]、[パスワード]、[パスワード確認]にそれぞれ情報を入力する
- [パスワード表示]のチェックボックスをオンにし、[性別]を「女」に、[年齢]を「40世代」に変更する

③[登録]ボタンを押して「登録情報確認」画面に遷移する
④「登録情報確認」画面の表示が「登録情報入力」画面の入力内容と一致しているかどうかを確認する

注3) URL http://sd.gihyo.jp/archive/2013/201303/support

◀図2 「登録情報入力」画面

図3 「登録情報確認」画面▶

■UIの操作

画面の部品を取得して値の確認などを行うためには、Androidのアプリ開発ではお馴染みのfindViewByIdメソッドを利用して対象となるUIオブジェクト取得します。取得したUIオブジェクトのメソッドを呼び出すことで、属性情報などを取得することも可能です。

sendkeyメソッドを使えば画面に対する入力処理を実行できますが、使われているIMEによって入力が異なる可能性があるため、注意が必要です。また、日本語の入力にも対応できません。この場合、setTextなどのメソッドを使えばよいでしょう。

Spinnerのように構成が複雑なUIオブジェクトは、KeyEventクラスのKEYCODE_DPAD_DOWNやKEYCODE_DPAD_UPのキーイベントと組み合わせて操作することができます(**リスト1**)。

■画面遷移の操作

Androidでは、画面遷移はIntentというしくみを使い、別の画面(Activity)を起動すること

で実現しています。いろいろなパターンのIntentを発行することで、画面遷移の正当性を確認できます(**リスト2**)。

▼リスト1　UIの操作(RegisterTest.java)

```
public void test_register() {
    // ……省略
    registerActivity.runOnUiThread(new Runnable() {
        @Override
        public void run() {
            // [ユーザID]を入力する
            useridElem.setText("tanaka");
            // [パスワード]を入力する
            pwdElem.setText("123456");
            // [パスワード確認]を入力する
            pwdCfmElem.setText("123456");
        }
    });
    // ……
    // [パスワード表示]のチェックボックスをオンにする
    TouchUtils.clickView(RegisterTest.this, dspPwdElem);
    // [性別]は「女」を選択する
    TouchUtils.clickView(RegisterTest.this, womanElem);
    // [年齢]を選択する
    TouchUtils.clickView(RegisterTest.this, ageElem);
    //  4番目の「40世代」を選択する
    for (int i = 0; i < 4; i++) {
        sendKeys(KeyEvent.KEYCODE_DPAD_DOWN);
    }
    sendKeys(KeyEvent.KEYCODE_ENTER);
    // ……省略
```

▼リスト2　Intentの指定による画面遷移(RegisterUnitTest.java)

```
public void test_mofify() {
    UserParcelable user = new UserParcelable();
    user.setSex("女");
    user.setAge("30世代");
    user.setPassword("password");
    user.setUserid("testid");
    Intent intent = new Intent();
    intent.putExtra("userinfo", user);

    setActivityIntent(intent);
    Activity registerActivity = getActivity();
    // ……省略
}
```

▼リスト3　結合テストの画面遷移(RegisterTest.java)

```
    // リスト1の続き
    //「登録情報確認」画面のActivityのモニタを追加する
    monitor = getInstrumentation().addMonitor(
            RegisterConfirmActivity.class.getName(), null, false);

    // [登録]ボタンを押して画面を遷移する
    TouchUtils.clickView(RegisterTest.this, registerElem);
    Activity confirmAct = getInstrumentation().waitForMonitorWithTimeout(
            monitor, 2000);
    // ……省略
}
```

結合テストのシナリオを考えると、表示されているActivityから次のActivityに遷移するように操作する必要があります。このようなテストケースを実装したい場合、InstrumentationTestCaseクラスとActivityMonitorクラスを利用することで遷移後のActivityを取得できます(**リスト3**)。

■メニューの操作

画面に表示されている部品のテストだけではなく、端末のハードウェアキーなどの動作も確認しなければなりません。たとえば、[MENU]ボタンの押下によるオプションメニューの表示はInstrumentation.invokeMenuActionSyncメソッドで、画面の長押しによるコンテキストメニューの表示はInstrumentation.invokeContextMenuActionメソッドで操作できます(**リスト4**)

テストの自動化における注意点

ここでは、JUnitを使ってテストを自動化する際に、初学者がつまずきやすいと思われるポイント(AndroidTestCaseとApplicationTestCaseの実装例：リスト1参照)をいくつか紹介します。皆さんが実装するときの参考にしてください。

■UIスレッド

Androidは描画やUIの操作を、「UIスレッド」という特別なスレッドで実行しています。Androidのテストは、UIスレッドとは別のスレッドで実行されるため、UIに関する処理はrunOnUiThreadメソッドを呼び出して実行する必要があります。runOnUiThreadメソッドを呼び出した後、Instrumentationクラスのwait ForIdleSyncメソッドによってUIスレッドの処理が完了するまで待ってから、後続の処理を行うようにします。

ただし、ユーザの操作をエミュ

レートするTouchUtilsクラスを利用する場合には、UIスレッドを意識する必要はありません。

■TouchUtilsクラス

TouchUtilsクラスを使えば、ドラッグ、スクロール、長押しなどの操作をエミュレートできます。また、UIスレッドを意識しないで済むため、テストの実装が少しだけ楽になります。

■スリープ

テスト対象のアプリケーションにおいて画面遷移に時間のかかるスレッドを利用している場合、テスト側でThread.sleepメソッドを呼び出してスリープしなければなりません。

■assertメソッドのエラー

JUnitのassertメソッドでエラーが発生したときには、tearDownメソッドで起動したActivityを終了すべきです。エラーとなったActivityが原因でテストを継続できなくなる可能性があります。

■警告ダイアログ

AndroidのJUnitでは、直接警告ダイアログを操作できるAPIを提供していません。そのため、android.view.WindowManagerImplやandroid.view.WindowManagerGlobalなどの隠されたクラスを利用する必要があります。

◆ ◆ ◆

Androidが拡張しているJUnit用のクラスは、前述のとおり目的に合わせて用意されています。そのテストケースの特徴と制限事項を確認したうえで、利用する機能を選んでいきましょう。

▼リスト4 オプションメニューのテスト（RegisterUnitTest.java）

```
public void test_menu() {
    Activity act = getActivity();
    sendKeys(KeyEvent.KEYCODE_MENU);
    boolean success = getInstrumentation().
  invokeMenuActionSync(
            act, R.id.version, 0);
    // ……省略
}
```

最後に

今回はテスト自動化ツールの使い方を紹介しました。Androidのテストに関しては、さまざまなアプローチが試されています。たとえば、発売されている機種が多すぎるため、リモート環境に実機を揃えて、開発者が共通で利用するようなサービスも公開されています（Testdroid、AppThwack、cloudmonkey、Scirocco Cloud、Remote Testkitなど）。開発のフェーズやテスト対象アプリケーションが抱えている課題に合わせて、最適なソリューションを選んでいくとともに、事例を積極的に共有できればとよいなぁと思っています。

最後になりますが、テストの自動化について体系化された知識を得たい方は、TABOK（Test Automation Body of Knowledge）を読んでみてはいかがでしょうか。TABOKは、米国ATI（Automated Testing Institute）が策定した自動テストに関する知識をまとめたものです。テスト自動化を効率よく進めるための手助けになることでしょう。SD

高木 基成（たかぎ もとしげ）日本Androidの会 神戸支部

株式会社サンモアテック勤務。トレンドテクノロジーを企業内システムに取り込むために頭を悩ませている。数千台のモバイルデバイスと、数十Tバイトのファイルが目下の対戦相手。趣味は勉強会。日本Androidの会 神戸支部、Google Developer Groups神戸などのコミュニティで活動中。良き家族、同僚、友人に支えられています。皆さん、いつもありがとう。

趙 文来（ちょう ぶんらい）日本Androidの会 神戸支部

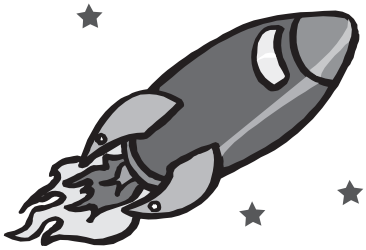
株式会社楓（フォン）勤務。Javaによる開発を楽しんでいる日々。Java EEを使った業務アプリの開発やOSSの検索エンジンの構築などを経験している。2010年からAndroid関連の開発に従事している。趣味はOSSの解析と家族との散歩。今は、日本語を習得するためのアプリケーションを開発しようと画策中。

新連載

Debian Developer　やまねひでき　henrich@debian.org

パズルゲームとディストリビューション開発

Debian Hot Topics



ディストリビューションの開発とは

みなさん、こんにちは。筆者らが参加しているDebian Projectでは「Debian」というディストリビューションを開発しています。本連載では、Debianの開発を通じて得た知見(あるいは愚痴)や、みなさんの興味を引きそうな開発状況のトレンドなどの話題をお伝えしていきます。短いページ数ではありますが、内側から見た情報をできる限り伝えていければと考えています。よろしくお付き合いくださいませ。

さて、本誌読者の中には、Linuxディストリビューションを仕事や趣味で利用されている方が多数いらっしゃることでしょう。ですが、「ディストリビューションの開発に関わっているよ！」という方は、(筆者が感じるかぎり)残念ながら少数派ではないか、と思います。

そこで今回は、普段何気なく使っているディストリビューションが「どんなふうに開発されているの？」というところをざっくりとですが、説明させていただきます。

「絵画」と「パズル」

まずは、ディストリビューション開発の「雰囲気」あるいは「性格」について、みていきましょう。ディストリビューションといえばオープンソースソフトウェア(以下、OSS)の集合体であるわけですが、「OSSそれ自体」の開発と「ディストリビューション」の開発はかなり異なっているなー、というのが筆者の見解です。では、どんなふうに違うのでしょうか？

自由に開発するOSS

例えてみると、OSSの開発は絵を描くことに似ています。エディタという名前の真っ白なキャンバスに向かい、思い思いのプログラム言語やフレームワークという好みの画材を使って、自分の頭の中にある姿を目の前に紡ぎ出していきます。テストコードというラフスケッチを繰り返す人もいれば、いきなり油彩絵具で色を載せていく人もいるでしょう。難解でトリッキーな構成を好んで描いては捨てる人もいれば、単純なモチーフについてひたすら細かく描写するのを好む人もいます。コードという画材で、自分の「理想の形を描く」のがソフトウェア開発の楽しみでしょう。

利用するOSや開発言語によって差はあるでしょうが、「なければ作る」「あったとしても気に入らなければ再実装する」「気に入らなければ捨てる」というのが基本的な姿勢になります。一言で言えば自由気まま、というところでしょうか。

ルールのもとで開発するディストリビューション

対してディストリビューションは「あるものを使う」のが基本です。車輪の再発明は好まれず、既存のソフトウェアの組み合わせで問題を解決しようとします。結果として、その開発はパズ

ルゲームの様相を呈します。数千あるいは数万ものソフトウェアについて、既存システムとの整合性を確認しつつビルドとインストールに最適なオプションを探してMakefileやautotools(automakeやautoconfなど)、Rakefile、setup.py、build.xml、pom.xmlなどの中をさまよい、ライブラリ間の依存関係を確認して突き合わせるなどの試行錯誤が続きます。それは、一片一片を組み合わせるパズルのようです。

また、ディストリビューションには数百、数千の人間が関わります。半ば必然の結果として無秩序のままでいることは許されず、ある一定のルールの中で各自のソフトウェアを組み合わせられるようにしよう、というコンセンサスが生まれます。この一定のルールというのが「パッケージングポリシー」であり、その結果として「パッケージ」という単位でソフトウェアがパズルのピースとして量産されるわけです[注1]。

しかし、ルールが決まっていくつものパッケージが生み出されていくにつれ、中にはピースを組み合わせてもうまく噛み合わない部分が出てきます。ルールを作っているのは神様ではないので、最初からすべてを見通した万能のルールなど作れないのですね。そんな場合にはルールを見直して、今まで曖昧(あいまい)だった部分に定義が追加されます。そして、新たなルールに合わせて既存のパズルのピースを調整しなおして、再度当てはめていくことになります。

ポリシーを維持するしくみ

Debianでの作業を具体的に考察してみましょう。さまざまなメーリングリストやIRCチャンネルなどでのコミュニケーションを通じて開発は進められます。その間に問題の指摘や改善提案が追加されていきます。ある程度概要がまとまると、それがdebian-policyメーリングリストで議論され、その結果がパッケージの構成や動作についての定義を記述した「Debianポリシーマニュアル」[注2]という文章に反映されます。

このDebianポリシーに従ってパッケージメンテナはパッケージを作成／維持していきます。このポリシーは(当たり前ですが)英語という自然言語で書かれているので、パッケージメンテナの解釈によってどうしても結果にブレが出てしまいます。その対策として、Debianでは「lintian」というツールをPerlで実装しています[注3]。

このツールは、パッケージの中のポリシーに沿っていない部分を見つけてくれるので、これを使ってパッケージのビルド時に機械的にチェックを行い、問題を見つけて各パッケージメンテナが修正していくのです。流れをまとめると「議論」→「ポリシーの更新」→「lintian実装」→「チェック」→「パッケージ修正」というサイクルです。図にしてみるとまるでDebianの渦巻のようですね(**図1**)。

ポリシーの遵守が品質を高める

ちなみに、Debianはパッケージの品質の高さを売りの1つにしていますが[注4]、それはポリシーの定義とそのポリシーを自動的に確認していくためのツールの存在に加え、このサイクルをきちんと回しながら、あまり行儀の良くないパッケージを洗い出して修正する作業を行って

注1) このルールは世の中に1つではなく、Red HatのRPM、Debianのdeb、Gentooのebuild、FreeBSDのports、NetBSDのpkgsrcなど、さまざまな人が、いろいろなかたちでルールを実装しています。「ディストリビューションによってパッケージ形式が異なるのは面倒だ、統一しろ！」という方もいらっしゃるようですが、各団体がそれぞれのポリシーとそれまで作成したパッケージという資産があるため、はっきり言ってこれは無理難題です。「パッケージ」をあなたの好きなエディタや言語、フレームワークなどに言い換えてみれば、前述の考えがナンセンスであることがわかるでしょう。

注2) URL http://www.debian.org/doc/debian-policy/

注3) 一時期、lintianの開発が停滞したころ、「lintianの開発が止まってるのはPerlで書いて可読性が低いからだ」などといって「linda」というPythonで実装されたチェックツールも作られたのですが、lintianの開発が盛り返してきてlindaはお亡くなりになりました……。

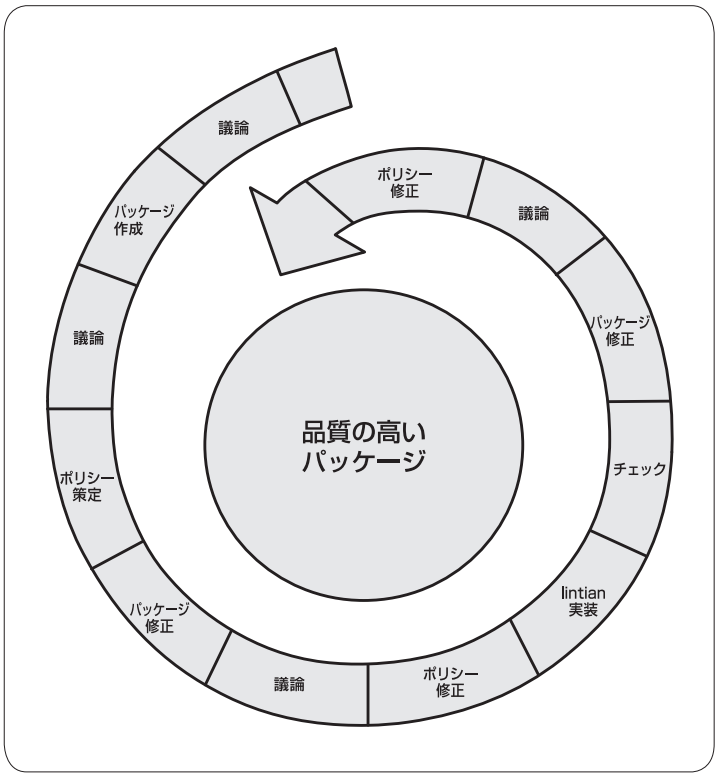
注4) あくまでも「パッケージ」の品質であって「ソフトウェア」そのものではない、というところに注意してください。言葉遊びに聞こえるかもしれませんがこの点は重要です。Debianではソフトウェア自体のクオリティも高く保とうと努力していますが、その品質は開発元(upstream)に大きく左右されます。そして、ソフトウェアの品質を簡単に測る術は残念ながら存在していません。

いるからです(イテレーティブである、と表現してもよいかもしれません)。

では、ほかのディストリビューションの場合はどうなのかと言うと、広く使われているのはRPMですが、Fedoraなどはパッケージングガイドラインはきちんとしたものがあるものの[注5]、それを自動的にチェックするためのツール「rpmlint」はガイドラインの変更とリンクして開発されていないので[注6]、「うまくこのサイクルを回せていないのでは……」というのが筆者の見方です。

なお、lintianによるチェックはあくまでもチェックでしかないので、(メンテナとしては行儀がよろしくないですが)無視してアップロードしてしまうことも可能です。一応、パッケージのアップロード時にサーバでもlintianを使ってチェックを行い、あまりに明白で致命的な問題については、アップロードをrejectしてリポジトリにパッケージが追加されないようになっています[注7]。まぁ、そんな問題があるようなパッケージをアップロードする前に気づけよ!というところですが、フェイルセーフ重要、ということで……。

▼図1　ポリシーの変更とパッケージ開発の流れ



「頭の体操」のおもしろさ

自由気ままに絵を描くのもおもしろいですが、パズルゲームにはパズルゲーム特有の「頭を捻る」楽しみがあります。ちょっと趣向の変わった知的パズルで遊んでみたいな、という方はぜひ一度Debianの開発に目を向けてみてください。easyモードから激ムズハードモードまで、さまざまな問題があなたのために用意されていますよ。

最後に

次回も筆者が担当させていただきます。以降のネタはぶっちゃけ今のところまだ未定ですので、何か取り上げてほしいものがありましたら気軽にお寄せください。本連載では、みなさんからのご意見、ご感想などを首を長〜くしてお待ちしておりますので、編集部(sd@gihyo.co.jp)宛にお送りいただくか、あるいは「何か一言だけ」という方は@gihyosdや@debianjp宛にいただければと思います。SD

注5) URL https://fedoraproject.org/wiki/Packaging/Guidelines/ja　日本語版で大体問題はないのですが、ここ2、3年ほど和訳の更新が追随されていないので、最新版を見たい方は英語版をどうぞ。
注6) URL http://rpmlint.zarb.org/cgi-bin/trac.cgi/log/trunk
注7) URL http://ftp-master.debian.org/static/lintian.tags

JBossって何?

大溝 桂
Kei Omizo
レッドハット(株)
JBossサービス事業部
ソリューションアーキテクト

今回のテーマ

はじめまして。レッドハットでミドルウェア製品のソリューションアーキテクト(SA)をしている大溝と申します。なにわ通信からバトンを受けて、ふたたび恵比寿通信をお届けします。レッドハットというとLinuxと連想されることが多いのではないでしょうか？　当社ではRed Hat Enterprise Linuxだけではなく、JBoss Enterprise Middlewareと総称されるミドルウェア製品も提供しています。そうとは言っても「JBossって何？」という疑問もあるかと思いますので、今回はこの疑問を解決していきます。

JBossの名前の由来

EJB＋oss＝EJBoss → EJBoss－E＝JBoss

JBossは1999年にOSS版のEJBコンテナとして登場しました。当初はEJB + ossでEJBossという名称でしたが、Sun Microsystems社との商標の問題が発生したため、"E"をとって"JBoss"となりました。その後、2001年にJBossを開発・サポートするJBoss社(JBoss Group LLC)が設立され、2006年にRed Hat社がJBoss社を買収し、2007年にRed Hat社がサポートするエンタープライズ版が発表されました。JBossという名称は、最初はアプリケーションサーバの名前だったので、JBossというとコミュニティ版のアプリケーションサーバであるJBoss ASを連想されることが多いかと思います。私も以前はそうでした……。JBossという名称は、JBoss Enterprise Application Platform、JBoss Enterprise BRMS、JBoss Enterprise SOA Platform……などRed Hat社がサポートするエンタープライズ向けのミドルウェア製品のブランド名にもなっています。

名前の語感

JBossというのはアプリケーションサーバだけではなく、いろいろなミドルウェアがあるんだ!!と、新発見？した方の中には、名前だけは聞き覚えがあった方もいらっしゃるのではないでしょうか？　それは、きっと名前がシンプルなので、どこかで見聞きしたことをうっすらと覚えていたのでしょう。JBossなんて初めてという方も、この機会にぜひ覚えてください。では、なぜJBossという名前が覚えやすく、記憶に残るのかを語感をテーマに、考えます。

「語感」を広辞苑で調べると、

①言葉の与える感じ。言葉が持っているニュアンス・響き
②言葉に対する感覚

と定義されています。

JBoss(ジェイボス)という単語からどのような印象を受けるでしょうか？　JはJavaの頭文字ですし、Bossはボス・上司・親分などの意味で頼りがいがあり力強い感じがするのでJavaに関係する強力な何かだ！？と考えずにはいられません。

『名前力――名前の語感を科学する』によると、

▼図1　JBossロゴ（https://community.jboss.org/wiki/TrademarkGuidelines）

J音については「Z/J音は、舌をふっくらと使う振動音。このため、Z/J音の名前は、呼ぶ人にも呼ばれるあなたにも豊かさや高級感を感じさせます。さらに、Z/J音は、息を上の前の歯茎にぶつけ乱気流を起こします。Jの乱気流のパワーは、まるで下町のお祭りのような庶民的なにぎやかさ[注1]」。

B音については「口先で破裂するB/P音は、にぎやかな威嚇音。強く前向きなので、アグレッシブ・パワーを感じさせる音なのです。さらに、唇の快感は、食と成長のイメージにつながるため、成長・繁栄・繁殖・倍増などの印象と密接に関係しています[注2]」と解説されています。JもBも、にぎやかさやパワーを感じる音なのだそうです。

さらにロゴを見てみるとJBossという文字とドットの組み合わせになっています（**図1**）。JBossを構成するアルファベットはすべて丸みを帯びた形状をしていますし、さらにドットが弧を描いて配置されていますので、尖った部分が1つもなくてキュートな印象です。にぎやかで強く前向きなパワーがあり、それでいてキュートって、人気者の予感がしますよね[注3]。

JBossって何?

JBoss Enterprise Middlewareは、JBossコミュニティで開発されているOSS製品にレッドハット社がエンタープライズ向けの改良とサポートを提供している製品群です。**図2**に示すように、アプリケーション開発から運用監視まで10種類の製品群があります。アプリケーションサーバだけではなく、いろいろな機能を実現するためのミドルウェアが提供されているのです。

- **アプリケーションの開発とデプロイ環境**
  - JBoss Developer Studio（アプリケーション開発環境）
  - JBoss Enterprise Application Platform（アプリケーションサーバ）
  - JBoss Enterprise Web Server（Webサーバ）
- **サービス実装**
  - JBoss Enterprise Data Services Platform（データの仮想化）
  - JBoss Enterprise Business Rule Management System（ビジネスルール管理、ビジネスイベント管理、複合イベント処理）
  - JBoss Enterprise Service Oriented Architecture Platform（アプリケーションやサービスの統合）
  - Red Hat Enterprise MRG Messaging（メッセージング）
  - JBoss Enterprise Portal Platform（ポータルサーバ）
- **ビッグデータソリューション**
  - JBoss Data Grid（インメモリデータグリッド）
- **運用管理**
  - JBoss Operations Network（JBossアプリケーション環境の運用監視）

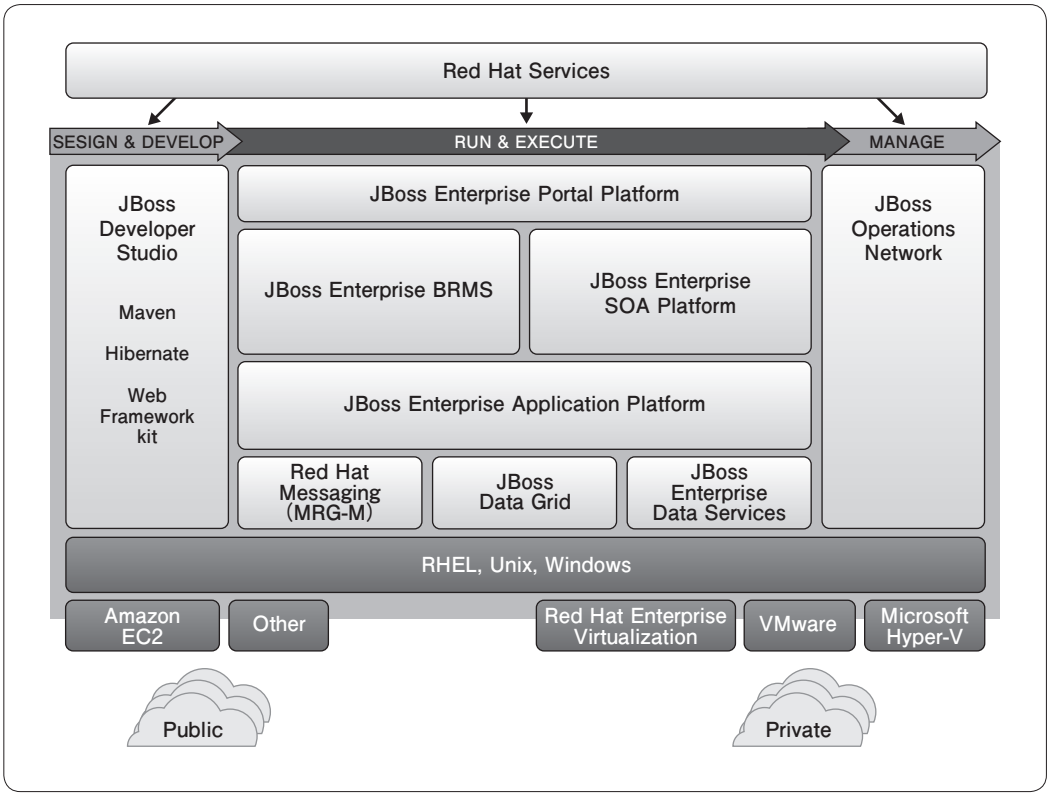
製品の詳細は当社Web[注4]で紹介していますので、こちらもぜひ参照してください。

JBoss ASプロジェクトの名前が変わる!!

JBossという名称は、JBossのコミュニティ、

注1）『名前力――名前の語感を科学する』黒川伊保子著、イースステージ、2009年4月、173ページ
注2）同著、190ページ
注3）ロゴに含まれる"by Red Hat"という文字は、オープンソースプロジェクトに基づいているが、エンタープライズ向けにレッドハット社が品質を担保するためのテストを実施し、パッチを適用していることを表しています。
注4）http://jp.redhat.com/products/jbossenterprisemiddleware/

▼図2　JBossの製品群



Java EE対応のアプリケーションサーバであるJBoss AS、また、商用のJavaEE対応アプリケーションサーバであるJBoss Enterprise Application Platformに使われています。製品とコミュニティの名称を分けるため、JBoss ASプロジェクトの新名称を選定中です[注5]。

・2012年10月1日～14日：新名称の公募
・2012年11月15日～2012年11月30日：新名称の投票
・2013年初頭：新名称の発表

というスケジュールになっていますが、2013年1月時点では結果は発表されていません。どんな名称になるか、皆さんもチェックしてください。

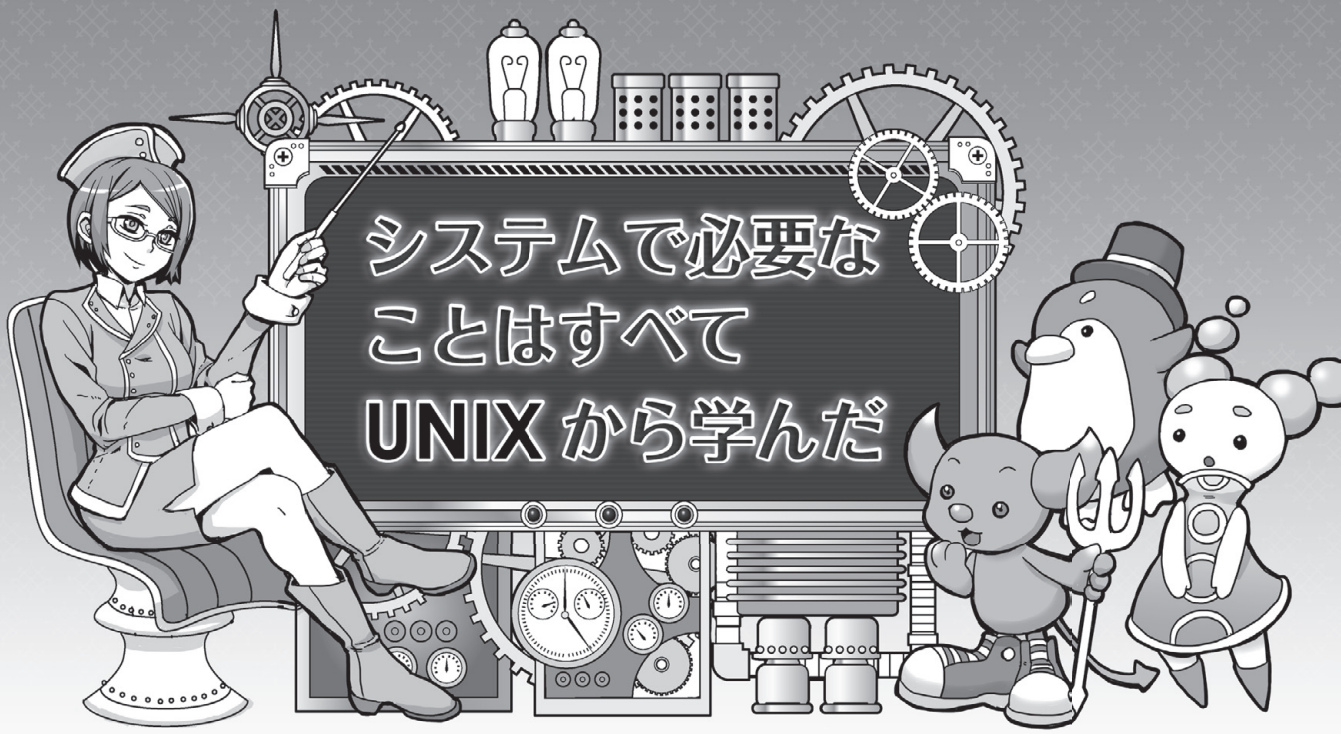
注5）http://www.jboss.org/vote

番外編「女性エンジニアからみた恵比寿」

恵比寿といえばレストランの激戦区で、代官山や広尾方面にも足を伸ばすとランチを食べられるレストランがたくさんあります。女性的にうれしいのは食後にコーヒー（カフェラテが選べたらなおうれしい）やデザートが付くお店です。ミドルウェアと格闘する合間に、美味しいランチでほっと一息できる、恵比寿はそんな場所ですごく気に入っています。

もっとJBoss

今回は「JBossって何？」というテーマでお届けしました。次回もミドルウェアの紹介を予定しています。SD



第9回 「エンジニアのおもちゃ?」

仮想化やデータセンタの利用が当たり前になり、コンピュータ、とくにサーバの運用がシステム管理者の手元から離れる時代になってきた。システムの運用コスト低減や消費電力の削減、自然災害時のリスク低減など多くのメリットがあるのだが、エンジニアの立場からすると手元にコンピュータがないのは心情的にはなんとなく寂しい。そんななか、エンジニアが手元に置いて遊べることを目的としたコンピュータがいくつもリリースされてきている。

有限会社エムブイシステムズ　水越 賢治　MIZUKOSHI Kenji　　イラスト：髙野涼香

仮想化が流行り

世の中仮想化が流行である。サーバの運用ではデータセンタでレンタルサーバを借りるときには仮想サーバの1つを借りるVPSが当たり前となり、物理サーバを1台丸ごと借りることのほうが特別のこととなった。一般的なWebやメールなどのインターネットサーバでは、現在のハードウェアではリソースをすべて使い切ることは少ない。このため仮想化してハードウェアを共有することでリソースを有効利用することが考えられた。これによってレンタルサーバではコスト削減ができるし、消費電力も削減できる。VPSの1ヵ月の利用料が500円以下などというプランもあり、自前でハードウェアを用意するよりはるかに安い。電気代を自分で払うより安いかもしれない。

ネットワークの仮想化も当たり前になりつつある。データリンクレイヤではVLANによって仮想経路を構成するのはもはや常識というか必須となった。IPレイヤでも大規模キャリアや広域にネットワークを構成する大企業ではMPLS(Multi-Protocol Label Switching)で 仮想経路を構成する方法が広く用いられている。さらにはOpenFlowのようなネットワーク機器と経路を統一的にモデル化して管理するしくみも登場した。今年はOpenFlowがキャリアなどで本格的に導入される年になるだろう。

自分のコンピュータをend to endで接続してネットワークを構成できることが魅力で、爆発的に普及したインターネットだが、最重要インフラとして社会のあり方まで変えてしまった。つながることが楽しい時代から安定性やセキュ

リティ対策、効率的運用が重要になってきていて、お楽しみの要素が少なくなってきている。とくにハードウェアの分野は著しい。今やルータやサーバのトラブルによるサービスの停止は即座に大問題になる。ベテランエンジニアでもルータやサーバの管理操作には細心の注意を払う必要があり、運用現場では「わくわく感」より「ピリピリ感」のほうが強い。仮想化でハードウェアの実在感が薄まり、効率化やコスト削減で自由に「おもちゃにできるコンピュータ」を用意することも難しくなってきた。

そんななか、デジタルガジェット、それもソフトウェアを開発したり、ちょっとしたハードウェアを操作するためのガジェットがいろいろ登場している。もともとは学生の教育用にと企画されたものが多いが、飛びついたのは学生だけでなくコンピュータを散々触ってきたはずのベテランたちも多い。

開発キット

コンピュータやネットワークの仮想化が進む一方で新しいデジタルガジェットが登場してきている。小さくて安価なコンピュータだ。その目的はもっぱらプログラムを書いて実行して楽しむ、あるいはコンピュータの動作を理解するための勉強に使うものだ。プログラムを書いて実行するだけならPCでできるのだが、あえてハードウェアも用意している。ワンチップのCPUにメモリとUSB I/F、Flashメモリスロットなどを搭載した小型のボードコンピュータだ。筆者と同年代のベテランコンピュータユーザだと、ボードコンピュータというとTK80などを思い出すかもしれない。

プリント基板むきだしのコンピュータは昔から開発キット、あるいは評価ボードなどの名前で販売されていた。最近のものとは違い、CPUメーカーや組込み機器メーカーが、製品に組み込むコンピュータを開発するメーカーに対して、ターゲットとなる機器のハードウェアを開発テストするために使用したり、ターゲット機器用のソフトウェアを開発デバッグする用途に販売していた。プロフェッショナル用途に販売するものだったので価格が高かったり、一般には流通していなかったりして、一般のソフトウェアエンジニアやホビーユーザには縁遠いものだった。

SAKURAボード

ルネサスが一般向けの開発キットとして発売しているの「がじぇっとるねさす」プロジェクト(略称がじぇるね)のSAKURAボードだ。正式名称はGRリファレンスボードだが、SAKURAの愛称で呼ばれることが多い。RX63シリーズの32bit CPUを搭載しArduino互換のプロセッサボードで、EthernetやUSBポートのコネクタが搭載されたものと、コネクタ類がついていないモデルがある。

SAKURAボードはそれ自体ではソフトウェアの開発ができず、他のPCなどでコンパイルしてオブジェクトをロードする必要がある。従来からある開発キットのクロス開発の手法だ。しかしもっと簡単に開発ができるように、SAKURAボードでは開発環境をクラウドで提供するという方法が提供されるようになった。具体的にはWindows PCのWebブラウザで開発サイトに接続し、ブラウザ上でコードエディット、コンパイルなどを行う。出来上がったバイナリは一度PCにダウンロードしたうえで、USBで接続したSAKURAボードに書き込む。この方法はWindows PC上に開発環境を構築する必要がないので、手軽に開発を始められるし、出先などで急遽プログラムの変更が必要になっても、現地にあるPCで開発、修正ができるという強みもある。さらにGUIで部品を組み合わせてプログラムを作ることができるAndroidアプリケーションも提供されている。ホビーユーザが取り組みやすい配慮がなされている。

がじぇっとるねさす

http://japan.renesas.com/products/promotion/gr/

変わったところでは、SAKURAボードに小型のLED表示パネルを付け、組込み用Rubyのmrubyを搭載した開発実験キットが発売されている。コンパクトな実験キットでRubyが簡単に動作させられるのはおもしろそうだ。

EAPL-Trainer

http://www.ilc.co.jp/commodity/eapl-trainer-mruby/

RetroBSD

PIC32マイコンで動作するBSD UNIXがあり一部で話題になっている。BSD UNIXというとVAX11用の4.2BSDが有名だが、16bitミニコンPDP11用のBSD 2系もあり、かつては広く使われていた。BSD 2系の最終版2.11をPIC32に移植したのがRetroBSDだ。ハードウェアの制限からWebサーバなどとしての使用は難しいが、PICのI/Oを使ってのデータロガーがUNIXツールを使って開発できる。能力が低い、メモリが少ないなど言われているが、PDP11よりはすべての性能が上回っている。うまく用途を考えれば実用になるかもしれない。

RetroBSD

http://retrobsd.org/

Raspberry Pi

最近注目を集めているのがRaspberry Piというボードコンピュータだ。オープンソースソフトウェアを活用した教育用コンピュータとしてケンブリッジ大学のEben Uptonらが始めたプロジェクトだ。子供たちが手が届くようにと低価格にし周辺機器は既存のものを使えるようにしている。ハードウェアの仕様も公開し誰でも作れるようになっている。2011年から実際のボードが出荷し始め、Linuxが実用的に動作することから話題になり一時期は入手困難になったほどだ。2012年末頃からは順調に出荷が始まっている。

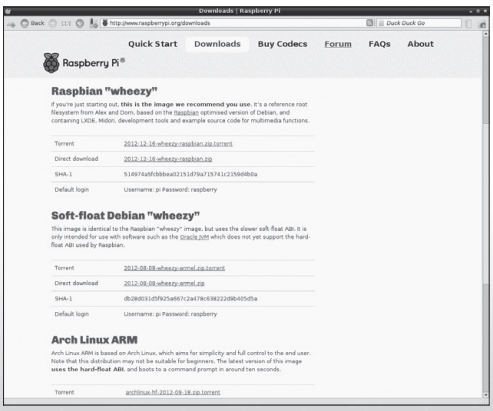
Raspberry Pi Official Site

http://www.raspberrypi.org/

●Raspbian "Wheezy"

Raspberry Piを動作させる方法はオフィシャルサイトにも情報が出ているし、本誌にも何度か紹介されている。OSはDebian GNU/Linuxをベースにした“オフィシャル”ディストリビューションのRaspbian "Wheezy"(**図1**)とベテランユーザ向けのArc Linuxなどがある。これ以外にFreeBSDやNetBSD、RiscOSなども稼働している。現在のところ環境が一番整っているのはWheezyなので最初はこれから始めるのが良いだろう。GUIが標準で用意されているだけでなくFirefoxなどのツール類も標準で搭載されているし、ソフトウェアもDebian GNU/Linuxのお作法apt-getで簡単にインストールできる。日本語環境だとフォントの問題からメインメモリが不足しがちだったがType

▼図1　Raspbian Wheezy



Bで512MBのメインメモリになり、そこそこ使えるようになった。もっとも本格的にワープロソフトウェアを動かしたければPCを使うべきだが。

FreeBSD for Raspberry Pi

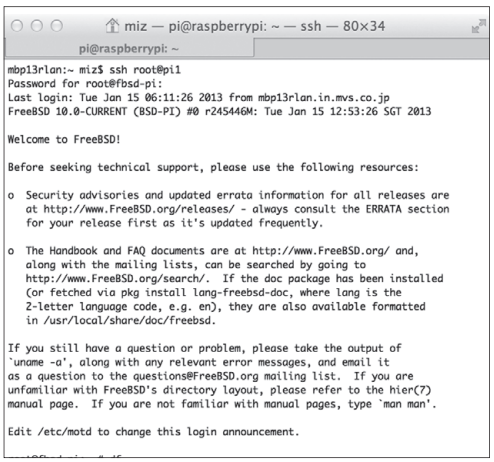
BSD系は従来からNetBSDがARMプロセッサ用に存在していて、FreeBSDでもNetBSDのカーネルをベースにしたARMアーキテクチャ版が存在していた。Raspberry Piがリリースされたころから開発が活発化している。

FreeBSDのARMアーキテクチャはFreeBSD 8から存在しているがそれなりに動作するのはFreeBSD 9からだ。Raspberry Pi用のFreeBSDはFreeBSD 10-currentつまり開発の最先端版で行われている。日々進化しているバージョンなので実際に動かす場合はできる限り最新が良い。可能ならFreeBSD10-currentの最新ソースからPC上でクロスビルドするのが最善だが、コンパイル環境構築に手間と時間がかかる。SDカードに書き込むイメージも公開されているのでそちらを使っても動作テストはできる(**図2**)。

Raspberry Pi版のFreeBSDは現時点ではまだいろいろと制限がある。まずグラフィックチップのサポートが完全でないため、ブート後のコンソールはシリアルポートが優先になっている。Raspberry PiのGPIOにはシリアルポートも搭載されているのでこれが使えるのだが、入出力は3.3VのTTLレベルなので通常のRS232Cポートに接続するためには、レベル変換回路が必要になる(MAX232などワンチップICがあるので簡単に作れる)。Raspberry Piのシリアルポートは送受信とアースの3線で、ハードウェアフローコントロールがないので高速な入出力はできない。

ほかにもSDカードスロットがDMAモードで動作していないためWheezyなどに比べるとディスクI/Oが遅い。またARM版のportsサポートもまだ十分でないため、x86系では問題

▼図2　FreeBSD for Piのコンソール



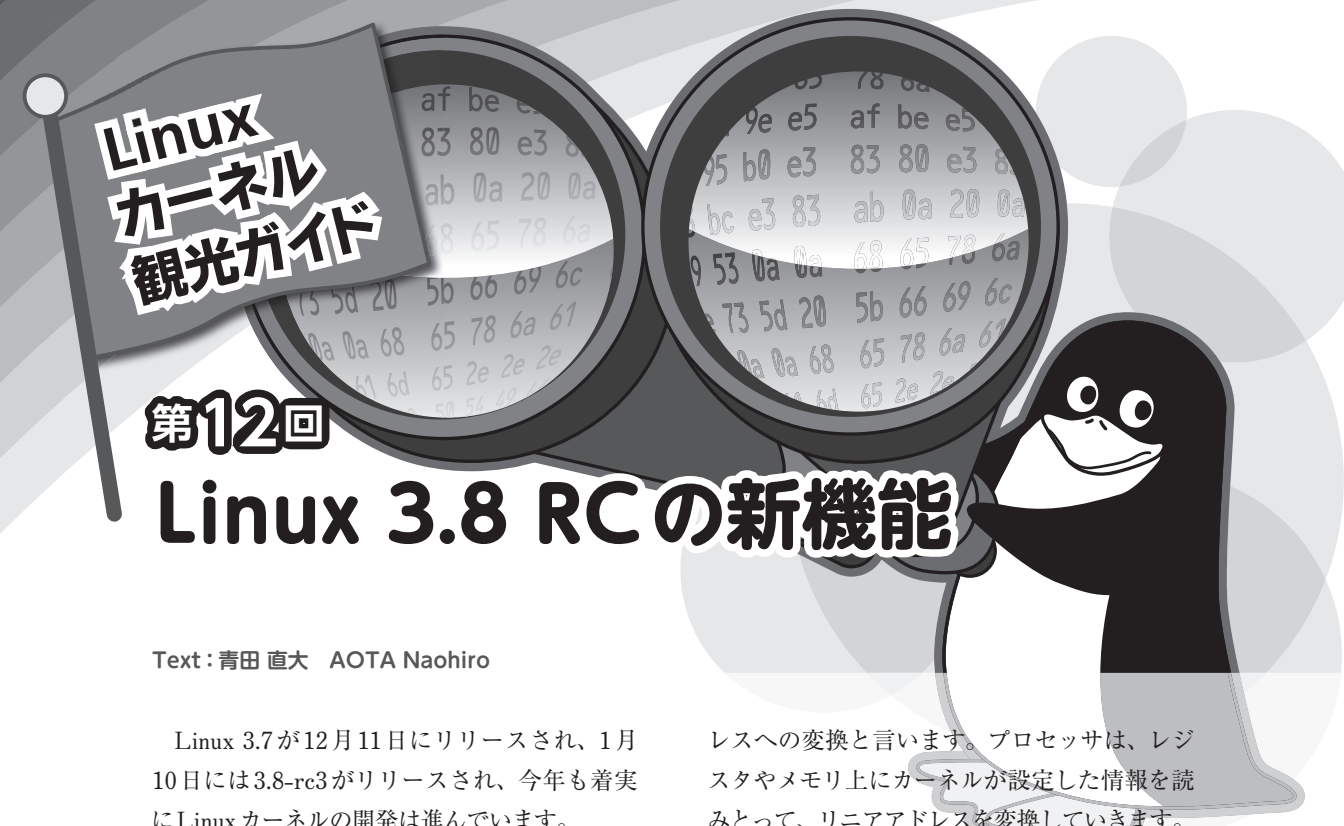
のないアプリケーションがARMのportsではコンパイルできなかったり実行時にcoreをはいてしまうこともある。このあたりの制限事項は日々改善されているので早晩解決すると思われるが、読者のみなさんも興味があったら開発、テストに参加してみていただきたい。

▼ FreeBSD for Raspberry Pi

`http://www.raspberrypi.org/archives/3094`

●Raspberry Pi日本ユーザ会

Raspberry Piの日本のオフィシャルユーザコミュニティも立ち上がっていて昨年末から活動を開始している。Raspberry Piのオフィシャルサイトに日本語でのフォーラムができているほか、財団のコアメンバとも密に連絡を取ってドキュメント類の翻訳などを始めている。3月には財団関係者の来日に合わせて日本でイベント開催の予定もあるようなので、興味のある方は情報フォローしてみてはいかがだろうか。現在はまだ日本のコミュニティ独自のWebサイトはないがオフィシャルサイトでのフォーラムやGoogle Groupsでアクティブに活動している。Japanese Raspberry Pi Users Groupで検索してほしい。積極的に活動してくれるメンバを募集しているとのこと。SD



第12回 Linux 3.8 RCの新機能

Text：青田 直大　AOTA Naohiro

Linux 3.7が12月11日にリリースされ、1月10日には3.8-rc3がリリースされ、今年も着実にLinuxカーネルの開発は進んでいます。

Linux 3.7での変更はこれまでの連載で、すでにいくつかは紹介していますが簡単におさらいしてみると次のようなものが挙げられます。

- ARMのマルチプラットフォームサポート
- 64bit ARMのサポート
- TCP Fast Openのサーバ側のサポート
- perf traceの追加
- moduleの署名のサポート

LKML（the Linux Kernel Mailing List）ではいつも通り次のバージョンのカーネルの開発が進んでいます。今回はLinux 3.8 RC版に現在コミットされている機能を紹介します。

リニアアドレスと物理アドレス

x86上で実行されるプログラムはそれぞれ固有の仮想メモリ空間で動作します。つまりプログラムが32bitのアドレス……たとえば0x12345678番地にアクセスしても、これは実メモリ上の0x12345678番地にアクセスするのではなく、プロセスごとに実メモリ上に割り当てられた領域の中のどこかへと変換されてアクセスされます。このことをリニアアドレスから物理アドレスへの変換と言います。プロセッサは、レジスタやメモリ上にカーネルが設定した情報を読みとって、リニアアドレスを変換していきます。

プロセッサはまず、cr3レジスタから、「ページディレクトリ」の物理アドレスを読み取ります。リニアアドレスの上位10bitが、ページディレクトリ内のエントリを示すオフセットになっています。このエントリには、「ページテーブル」の物理アドレスの上位20bitが入っています。ここで上位20bitとなっているのは、各ページの大きさが4KBであり、ページの先頭アドレスが4,096の倍数であるため、下位12bitはすべて0になっているからです。リニアアドレスの次の10bitは、今度はページテーブル内のオフセットになっています。これもページディレクトリと同じ構造になっており、「ページ」の物理アドレスの上位20bitが入っています。このアドレスに残りの下位12bitを足すとアクセスされる物理アドレスになります。

このようにリニアアドレスを変換するだけで、数回のメモリアクセスが必要になってしまいます。そこでリニアアドレスと物理アドレスとの対応をメモリよりも速い「TLB（Translation Lookaside Buffer）キャッシュ」に貯えておきます。こうして最近アクセスされたアドレスであればTLBキャッシュを使い、すぐに物理アドレスが取得できるようにしています。

Hugepage

前節の内容でもわかるとおり、各ページのサイズは4KBになっています。昔のプログラムであれば、4KBでもそれなりに十分であったかもしれません。しかし、今ではより大きなメモリを一度に使うプログラムも一般的になっています。たとえば、4MB程度のメモリ領域にランダムにアクセスするようなプログラムでさえも、TLBキャッシュミスが起こりやすくなります。そこでPentium以降のプロセッサでは、4MBより大きなページサイズ(Hugepage)を使うことができるようになっています。この場合ページディレクトリ内のページ長フラグが1になっており、リニアアドレスのページディレクトリ以外の22bitが4MBのページの中のオフセットになります。

Hugepageを使うと、TLBキャッシュミスの軽減だけでなく、メモリの節約にもなります。たとえば、2GBのメモリを使っている場合のことを考えてみましょう。ページサイズがすべて4KBであれば、ページディレクトリ1個とページテーブル512個を使用します。一方でページサイズがすべて4MBであれば、ページディレクトリのエントリだけで足りてしまうので、差し引き2MBのメモリが節約できることになります。

この機能をユーザ空間から使うには、まず/proc/sys/vm/nr_hugepagesと/proc/sys/vm/nr_overcommit_hugepagesの値を適切に設定します。nr_hugepagesはHugepageの個数を示します。この数だけのHugepageがカーネル内部で予約され、Hugepage以外の用途に使われず、しかもメモリが足りなくなってもスワップアウトされることはありません。nr_overcommit_hugepagesは、nr_hugepages以上のHugepageが要求されたときに、最大いくつまでのHugepageの追加を許可するかの値です。こうやって確保されたHugepageは"surplushuge page"と呼ばれ、プログラムが使い終わった時点で解放されます。

こうやって確保したhugepageを、次の3つのいずれかの方法でプログラムから使用します。

- hugetlbfsをmountし、そこにファイルをmmap()する
- MAP_HUGETLBのフラグつきでmmap()する
- SHM_HUGETLBのフラグつきでshmget()する

詳しくはLinuxカーネルのソースコードのtools/testing/selftests/vm/の下にサンプルコードがありますので、それを参照してみてください。また、LD_PRELOADでライブラリを読ませたりすることで、malloc()の挙動を置き換えて、コンパイルし直さずににHugepageを使わせることができるlibhugetlbfsというライブラリもあります。

Hugepageの使用状況は/proc/meminfoから知ることができます。たとえば、2MBのHugepageを500個確保して、Hugepageで256MBの領域を使うプログラムを動かすと**図1**のようになります。

HugePages_T otalがnr_hugepagesで設定した500に、HugePages_Freeが500-128で372になっていますね。また、hugepageのサイズがHugepagesizeにあるとおり2MBになっています。

Transparent Hugepage

前節のようにユーザランドからhugepageを使うのには次の2つのデメリットがあります。

- プログラムの書き換え、またはlibhugetlbfsを使う必要がある
- hugepageにしか使えない領域を事前に確保している

▼図1　Hugepageで256MBの領域を使うプログラムの動作

```
% echo 500 |sudo tee /proc/sys/vm/nr_hugepages
% cat /proc/meminfo|grep Huge
AnonHugePages:   4667392 kB
HugePages_Total:     500
HugePages_Free:      372
HugePages_Rsvd:        0
HugePages_Surp:        0
Hugepagesize:       2048 kB
```

こういった問題に対応し、アプリケーションが「透過的に」hugepageを使うことができるようにするためのカーネルのシステムが“Transparent Hugepage”です。

Transparent Hugepageは次のことを行います。

- 大きなメモリ領域が確保されたときにhugepage化を試みる
- hugepageが確保できるようになったときに、通常ページの領域を自動的にhugepage化する

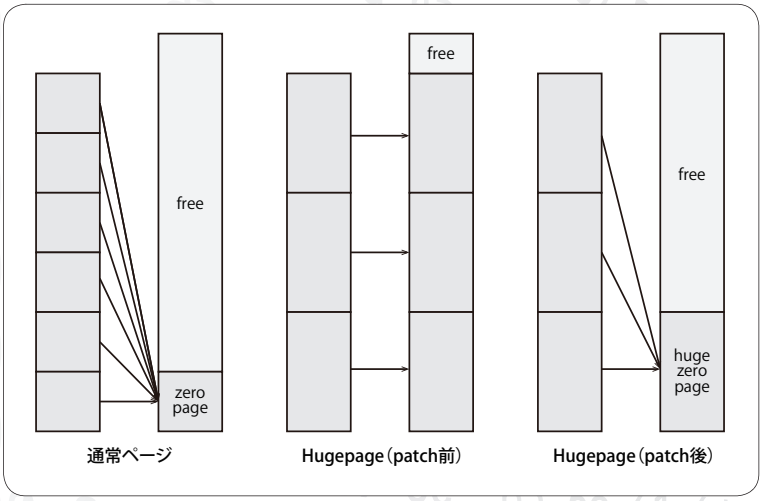
まず、前者について見てみましょう。Transparent Hugepageには2つの動作モードがあります。/sys/kernel/mm/transparent_hugepage/enabledをalwaysにするとどのような領域でもhugepage化の試みが有効になります。しかしこの場合、メモリが無駄に使われる可能性もあります。たとえば、2MBの領域を確保したものの、結局最初の1byteしかアクセスしなかったというシナリオを考えてみましょう。Transparent Hugepageを使っている場合、その最初のアクセスで2MBの領域が実際にメモリ上に確保されます。一方でHugepageにしていなければ、実際に割り当てが必要であったのは4KBだけであったはずです。そこで、先ほどのenabledをmadviseにすることでアプリケーションがMADV_HUGEPAGEのフラグを付けて、メモリ領域をmadvise()したときにだけHugepage化が試みられるようになります。

次に後者を見てみましょう。Transparent Hugepageが有効になっていると、khugepagedというカーネルスレッドが実行されます。これはシステムのメモリ全体を定期的に少しずつスキャンして、デフラグしたり、Hugepage化したりするスレッドです。khugepagedの挙動は/sys/kernel/mm/transparent_hugepage/khugepaged/下で調整できます。khugepagedは毎回、pages_to_scanの数のページをスキャンし、scan_sleep_millisecsに設定されたミリ秒数だけsleepします。ただし、Hugepageの確保に失敗した場合、alloc_sleep_millisecsに設定されたミリ秒数だけ次のHugepageの確保を遅らせます。

▼図2　LKMLのpatch投稿の抜粋

```
posix_memalign((void **)&p, 2 * MB, 200 * MB);
for (i = 0; i < 200 * MB; i+= 4096)
    assert(p[i] == 0);
```

▼図3　zero page



huge zero page

さて、ここからが3.8で入ったTransparent Hugepageの改良点についての話になります。ここまでで紹介したTransparent Hugepageはアプリケーション側がとくに何も考えなくとも、カーネル側でHugepageを使わせてパフォーマンスや消費メモリを改善してくれるものです。しかし、ここに1つ見落としがあったのです。

図2はLKMLのpatch投稿[注1]からの抜粋です。

このコードは200MBのメモリ領域を（Hugepage

注1）http://lwn.net/Articles/515526/

化がうまく作動するように2MBにアラインメントしつつ)確保し、その全領域が0であることを確認しているだけのコードです。この部分が実行されたあとにプログラムの実メモリ消費量を見てみると、Transparent Hugepageが使われていない場合であれば400KBほどであるのに対して、Transparent Hugepageが有効にされている場合200MBが消費されてしまいます。これではもともとの目的が達成できていませんね。これはなぜなのでしょうか。

まずTransparent Hugepageが使われていない場合に、200MB確保したにもかかわらず400KBほどしか実メモリが消費されていないことに注意してください。これはzero pageというもののおかげです(**図3**)。Linuxカーネルは実メモリの割り当てを、アプリケーションからのアクセスがあるときまで遅延します。新しいページに前に使っていたアプリケーションのデータが残っていると、悪意のあるプロセスに情報漏えいしてしまう可能性があるので、まず新しいページは0で埋められなければなりません。

ここでzero pageという0で埋められた読み込み専用のページを用意します。アプリケーションが最初にメモリを読み込もうとしたときは、このzero pageへと関連付けを行います。その後、書き込もうとすると、このページは読み込み専用なので、ページフォルトが発生し、カーネルに制御が移ります。この時点まで実メモリの割り当てを遅延させているので、400KBほどしか消費されていない、というわけです。

ところが、Hugepageの場合にはzero pageがこれまで存在していませんでした。そのため、上で見たように実メモリ消費が200MBとなっていたわけです。ということで、3.8ではhuge zero pageというHugepage版zero pageが追加されています。とはいえ、2MBもずっとhuge zero pageのために確保しておくのは困ってしまいます。ですので、こちらは(通常のzero pageと違って)参照カウントされており、誰も使わなくなった時点で解放されるようになっています。

sparse file

仮想化のディスクイメージファイルなどによく使われるsparseファイルというファイルがあります。これはディスク上に実際には割り当てられていない領域を持つファイルのことです。たとえば、

```
$ truncate -s 100T foo
```

とすると一見、100TBのファイルができたように見えます(もちろん筆者は100TBものディスクなんて持ってません)しかし、実際にはまったくディスク上に領域は割り当てられておらず、書き込みがあったときに随時領域が割り当てられます。

hole punching

sparseファイルに関連してhole punchingというものがあります。これはファイルの指定した部分に、あとから「穴を開けて」データをディスク割り当てから外して、sparseなファイルを作るためのAPIです。FALLOC_FL_PUNCH_HOLEフラグを付けてfallocate()を呼び出すと、指定したoffsetからlenの長さの部分を割り当てから外してしまいます。

実際にhole punchingを見てみましょう。util-linuxのfallocateコマンドに、-p(= --punch-hole)というオプションがあるので、それを使います(**図4**)。

-oでoffsetを2MBにして、-lでlengthを3MBと設定しています。実際にls -lhsの結果の一番左側、割り当て容量の部分が10MBから7MBになっていることがわかりますね。

SEEK_DATAとSEEK_HOLE

さて、こうしてあちこちに穴のあいたファイルを扱うことを考えてみましょう。巨大で多く

の場所が割り当てられていないファイルの場合、もしもどこが割り当てられているのか、どこが割り当てられていないかを知っていれば効率的に処理をすることができます。これを実現するのが、lseek()のSEEK_DATAとSEEK_HOLEです。Linux 3.8では、このサポートがext4に追加されました。

リスト1のようなコードで、ファイルfooのデータのある場所と「穴」のある場所を表示できます。先ほどのfooに対して実行してみると、次のように表示されます。

▼リスト1　fooのデータと穴のある場所を表示

```
#define _GNU_SOURCE
#define _FILE_OFFSET_BITS 64
#include <stdio.h>
#include <unistd.h>
#include <sys/types.h>
#include <sys/stat.h>
#include <fcntl.h>
int main()
{
  int fd = open("foo", O_RDONLY);
  off_t offset = lseek(fd, 0, SEEK_DATA);
  off_t end = lseek(fd,0,SEEK_END);

  if(offset != 0){
    printf("0x%012lx: hole n", (off_t)0);
  }

  while(offset >= 0 ){
    offset = lseek(fd, offset, SEEK_DATA);
    if(offset < 0) break;
    printf("0x%012lx: data n", offset);
    offset = lseek(fd, offset, SEEK_HOLE);
    if(offset < 0 || offset == end) break;
    printf("0x%012lx: hole n", offset);
  }
  close(fd);
  return 0;
}
```

▼図4　hole punchingの様子

```
$ dd if=/dev/zero of=foo bs=1M count=10
10+0 レコード入力
10+0 レコード出力
10485760 バイト (10 MB) コピーされました、0.00505831 秒、2.1 GB/秒
$ ls -lhs foo
10M -rw-r--r-- 1 naota naota 10M  1月 23 21:51 foo
$ fallocate -p -o 2M -l 3M foo
$ ls -lhs foo
7.0M -rw-r--r-- 1 naota naota 10M  1月 23 21:51 foo
```

```
% ./hole
0x000000000000: data
0x000040000000: hole
0x000080000000: data
```

このように先ほど開けた穴の位置が確認できます。

その他のLinux 3.8の変更

Linux 3.8では、ほかにも以前紹介したフラッシュフレンドリーなファイルシステム“f2fs”がさっそく取り込まれていたり、ext4がinodeインラインデータをサポートしていたり、さらにbtrfsにディスクの交換をより便利にする機能も入っていたりと、ファイルシステム関連の変更がおもしろいリリースとなりそうです。

UEFI secure bootにむけて

最後にまだまだ提案中・議論中の“secure boot”に向けた機能について解説します。UEFI(Unified Extensible Firmware Interface)とはBIOSを置き換える目的で開発が進められている、新しいOSとファームウェアのやりとりのための仕様です(**図5**)。そしてこのUEFIに、OSが起動する前の攻撃(たとえばあらかじめrootkitが仕込んであるOSを起動するなど)を防ぐことを目的としてsecure bootというものが制定されています。

さてsecure bootとはどのようにしてboot前のセキュリティを保証しているのでしょうか。まず、secure bootには2種類の鍵(「プラットフォーム鍵」と「鍵交換鍵」)と2つの署名データベース(「許

可データベース」と「ブラックリストのデータベース」)とがあります。

プラットフォーム鍵は、そのマシンの所有者が管理(=秘密鍵を持ち)するもので、鍵交換鍵は、OEMやOSベンダが管理する鍵です。許可データベースにその署名が掲載されているバイナリが実行可能となっています。

プラットフォーム鍵が設定されていないときのことをsetupモードと呼びます。この状態からまず最初にマシンの所有者が鍵を作り、その公開鍵をプラットフォーム鍵として登録し、userモードへと移行します。そして、OSベンダなどが配布する公開鍵を、プラットフォーム鍵の秘密鍵を使って署名して鍵交換鍵として登録します。署名データベースに実行バイナリの署名を登録するには、プラットフォーム鍵あるいは鍵交換鍵のどちらかで署名がされている必要があります。OSベンダの鍵交換鍵を登録したことで、OSベンダを信頼して、そのベンダが署名した実行バイナリの署名を追加してもらうことができるようになります。

先ほども述べたとおり、署名データベースにないバイナリ(=OSなど)であれば起動できなくなるので、これで悪意のあるOS・ブートローダを防ぐことができるようになります。

少し話はそれますが、FSF(Free Software Foundation)などがsecure bootに対して懸念しています。これはOEMがプラットフォーム鍵を事前に設定し、なおかつプラットフォーム鍵をリセットしてsetupモードに強制的に戻す機能や、secure bootを無効にする機能を提供しない可能性があるからです。もしそうなれば、そのマシンはデフォルトのOS以外のOSを起動できなくなります。

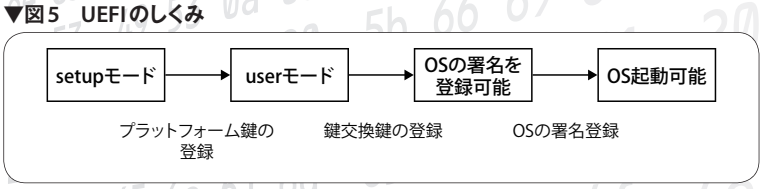
さて、このsecure bootをLinuxで実現するにあたって問題とされているのが、kexecというシステムコールです。kexecはカーネルを読み込んで、そのカーネルをBIOS、UEFIを通さずに直接起動するもので、より速い再起動ができるなどの利点があります。直接UEFIを通さずに任意のカーネルが読み込めるということは、UEFIのsecure bootの効果を台なしにしてしまいます。kexecに制限をかけなければ、認証されたLinuxでkexecを使って不正なバージョンのOSを起動するという攻撃が横行し、ブラックリスト入りされてしまうのではないかという危惧もあり、secure bootでのLinuxをどのように制限し、どのように実装するかが議論されています。 その中で今おもしろいものがELF executable signing andverification[注2]です。これはELF実行バイナリの".signature"セクションに署名を追加し、バイナリ実行時にその署名を確認できなければ、プロセスの実行を拒絶するというものです。鍵は今のところ、Linux 3.7で追加されたモジュールの署名に使われたものを使用することが想定されているようです。

まとめ

今回はLinux 3.8に追加される機能の中から、huge zero pageとext4に入ったSEEK_HOLE、SEEK_DATAと、現在議論中のUEFI secure bootに向けた新機能の議論について解説しました。SD

注2) http://thread.gmane.org/gmane.linux.kernel/1422530

▼図5　UEFIのしくみ



March 2013

NO.17

Monthly News from jus

Japan UNIX Society

日本UNIXユーザ会　http://www.jus.or.jp/
波田野 裕一　HATANO Hirokazu　tcsh@tcsh.csh.sh
高野 光弘　TAKANO Mitsuhiro　takano32@jus.or.jp

人のチカラ、インターネットのチカラ

Internet Week 2012

■Internet Week 2012
【日時】2012年11月19日（月）～22日（木）
【会場】富士ソフト アキバプラザ

今回は、2012年11月に行われたInternet Week 2012のレポートをお送りします。Internet Weekは日本ネットワークインフォメーションセンター（JPNIC）が主催するイベントで、インターネット関連の最新技術動向を学び、議論するプログラムが多数実施されます。jusはこのイベントに後援団体の1つとして参加しており、プログラムの企画や告知などで協力しています。

今年のInternet Weekは4日間に渡って合計33本（同時開催イベント2本含む）のプログラムを行い（**表1**）、約2200名が参加しました。レポートではその中から2本を選び、内容を紹介します。

■OpenFlowチュートリアル＆ハンズオン
【日時】2012年11月21日（水）　13:00～18:30
【会場】富士ソフト アキバプラザ 6階
セミナールーム3

このプログラムは、トラフィックをコントロールできるOpenFlowスイッチについて、データパス（スイッチ）に対してどのような概念で設定を行えばよいかなどを解説するものです。とくに今回は「作ってわかるOpenFlow—Tremaチュートリアル＆ハンズオン」と題し、Trema開発チームの鈴木一哉さんと高宮安仁さんからチュートリアルおよびハンズオンというかたちで説明が行われました。TremaはOpenFlowスイッチのコントローラを記述したり、スイッチのシミュレーションを行ったりできます。参加者には事前に仮想化機構向けのイメージが準備され、Tremaが動作するプラットフォームをもたない参加者でも気軽にハンズオンに参加できるかたちになっていました。

■実際にRubyを書きながらTremaを学ぶ

まず、OpenFlowのフレームワークのPOX、NOX、Floodlightについての紹介が行われ、それらと比較することで、プログラミング言語Rubyで記述できるTremaがいかに簡潔であるかを理解できました。チュートリアルでは、Tremaのコントローラの記述に使うRubyについての簡単な説明から、OpenFlowスイッチのコントローラの記述の紹介、それから具体的な利用シーンなどが紹介されました。ハンズオンでは、単純にスイッチを起動するだけのプログラミングから、MACアドレス学習型スイッチをどのようにして記述するかまでを実践することで、Tremaの具体的な使い方を実感しながら理解できました。

■今後のTremaの動向

なお、現行のTremaはOpenFlow 1.0に対応し、Ruby 1.8での記述ができるというものになりますが、開発中の次期TremaはOpenFlow 1.3や、Ruby 1.9での記述にも対応する予定という話も聞くことができました。これからも楽しみな技術です。

■IP Meeting 2012
～人のチカラ、インターネットのチカラ～
【日時】2012年11月22日（木） 9:30～17:30
【会場】富士ソフト アキバプラザ 5階 アキバホール

最終日には、この1年のインターネットの最新動向を伝え、今後に向けた議論を行い、1年間を総括する「IP Meeting」が午前と午後に渡って開催されました。

■2012年話題となったテーマ、今後の課題について大いに語り合う1日

前半は「Internet Today!」と題して、「IPv6とIPv4の動向と対策」「インターネット運用の動向／ホットトピック」の2つのテーマについて講演と議論が行われました。1つめの「IPv6とIPv4の動向と対策」は9時30分開始とエンジニアには朝早い時間にもかかわらず早々に会場の席が埋まるほどの盛況ぶりで、IPv6導入に対する注目度の高さをあらためて感じました。2つめの「インターネット運用の動向／ホットトピック」では、「人為的ミスによる大規模障害」「セキュリティ・インシデント」など2012年にとくに運用現場において話題となったトピックについて振り返りと考察が行われました。

後半はInternet Week 2012のキャッチフレーズである「人のチカラ、インターネットのチカラ」をテーマとしたパネルセッションが行われました。日本のインターネットを外の視点で見たときに、その強みは何なのか、そして課題は何なのかを、鉄道会社、大学、海外のエンジニアの視点で指摘し議論する意欲的な内容が展開されました。レイヤ間での分業が進んだ結果、我々は「インターネット屋」としてインフラを支えあっていくだけの気概が薄まっているのではないか、という意見が印象的でした。

1日に渡る長いプログラムでしたが、参加者の皆さんにもインターネットの現状とチカラ、これからの方向性について思いを馳せる良い機会になったのではないでしょうか。SD

▼表1 Internet Week 2012 プログラム

■11月19日

9:15～11:45	インターネット資源管理の基礎知識（ドメイン名／DNS／IPアドレス）	CSIRTの今とこれから	エンジニアも知らなきゃならない財務会計
13:00～15:30	第23回 JPNIC Open Policy Meeting ＜ポリシーワーキンググループ主催＞	BYOD時代のスマートフォンリスク管理	ルーティングチュートリアル
16:00～18:30		標的型攻撃の現状と対策	肌で感じる！インターネットルーティングセキュリティ
18:45～20:15	サイバー攻撃のこれからを考える夕べ		Peering in Japan

■11月20日

9:15～11:45	これからはじめるIPv6	インターネットをめぐる国際的な規制の動向	クラウド基盤運用術
13:00～17:00	IPv6 ハンズオン ネットワーク編 ～AlaxalA製ルータを使って～		
13:00～15:30	IPv6実践講座 ～トラシュー、セキュリティ、アプリ構築まで～	インターネットの決めごとの作り方を学ぼう	パケットフォワーディングを支える技術
16:00～18:30		第35回 ICANN報告会	
18:45～20:15	ISOC-JP BoF		地方在住エンジニアを盛り上げましょう！ BoF

■11月21日

9:15～11:45	DNSSECチュートリアル	IPv6各種移行・共存技術解説	サービス事業者に関連する法的問題の実例とサイバー犯罪の実態2012
9:15～13:15	IPv6 ハンズオン ネットワーク編 ～CISCO製ルータを使って～		
11:45～13:00	親の心子知らず？ 委任にまつわる諸問題について考える ～ランチのおともにDNS～		
13:00～15:30	DNSDAY	OpenFlow チュートリアル＆ハンズオン	意外と知らないソーシャルメディアの落とし穴
16:00～18:30			—
14:00～18:00	IPv6 ハンズオン サーバ編		
18:45～20:15	日本DNSオペレーターズグループBoF		

■11月22日

9:30～11:50	IP Meeting 2012 ～人のチカラ、インターネットのチカラ～	・Internet Today! 1) IPv6とIPv4の動向と対策 2) インターネット運用の動向／ホットトピック ・テーマセッション 3) ～人のチカラ、インターネットのチカラ～ 4) クロージング	—
11:50～12:50			Tier1ネットワークの高可用性・高効率性を追求したグローバルオペレーション
12:50～17:30			—
18:00～20:00	懇親会		

第35回 Ubuntu Monthly Report

LibreOffice 4.0の新機能

今回は、すっかり定番になったLibreOfficeの最新版である4.0の新機能を紹介します。

Ubuntu Japanese Team　あわしろいくや　AWASHIRO Ikuya　ikuya@fruitsbasket.info

Why 4.0?

LibreOffice 4.0の機能もそうですが、4.0というバージョニングがまず気になるかなと思います。もともとは3.7という順当なバージョンでリリースされることになっていました。現在までのタイムベースリリースでは、メジャーバージョンアップ版のタイミングは毎年2月と8月だったのですが、バージョンを上げるのに決まった時期でなくてはならないということはない（その時期まで待つ必要はない）、ということで4.0に上がることになりました。

メジャーバージョンアップとして特筆する機能が実装されたわけではありませんが、これまでのバージョンアップと同じように新機能が盛り込まれています。そして一部以前のバージョンとの非互換は発生しています。具体的には、削除された機能があります。

削除された機能で真っ先に目につくのは、Microsoft Office 95以前の形式で保存ができなくなったことです。現在は、こんな古いバージョンは使われておらず、その割には日本語で保存すると文字化けしてしまうという重大な不具合が修正されないまま残っていたので、誤ってこの形式で保存してファイルが読めなくなった、という事故が発生しなくなるというメリットが生じることになります。

ほかにもStarOffice[注1] 5.x以前の形式ではまったく読み書きができなくなりましたが、日本では販売されておらずユーザもほぼいないと思われるので、実害はないでしょう。また、ついにPPC Mac用のバイナリが提供されず、（Mac）OS X版のサポートは10.6からになりました。

あとは目に見えない部分の変更で、API、とくにUNOのAPIで非公開になったり、削除されたものが多数ありますが、もう使われなくなったものばかりのようなので、こちらもあまり気にすることはなさそうです。

というわけで、結論としてはバージョンは上がっているもののこれまでのバージョンアップとあまり違いはない、と思っていただければいいかと思います。

ちなみにApache OpenOffice（AOO）も4.0がリリースされることになっており、こちらはルック&フィールが変更になるという違いが発生することになりますが、今回はとくに取り上げないので興味のある方は筆者によるgihyo.jpの新春特別企画『LibreOffice/Apache OpenOffice ～2012年の出来事と2013年の新バージョンリリース～』[注2]をお読みいただければと思います。

今回のリリースの特徴

今回のリリースで特徴的なのは、いくつかの機能をAOO 3.4から取り込んでいることです。後述する

注1）ここでいうStarOfficeはOpenOffice.orgの前身となったもので、NECのグループウェアのことではありません。

注2）http://gihyo.jp/lifestyle/column/newyear/2013/libo-aoo-prospect

Calcの関数がそうですし、Draw/Impressの[角の形状]もそうです。正規表現のエンジンが変更になったのもそのせいです。とくに関数は興味深く、もともとはオラクル時代にEike Rathke氏によって実装されており、CWSというブランチでは開発が完了していました。ただしこれはリリースされないまま終わり、Apache License 2.0（AL2）になってCWSのソースコードは取り込めなくなりました。しかし、オラクルの人[注3]がこのブランチをAL2にしてAOOの開発版に取り込みました。さらにそれをLibreOfficeに取り込んだのです。取り込んだのはRed Hatに転職したEike Rathke氏でした。

Writer

範囲指定してコメント

これまでも[挿入]-[コメント]でコメントを入力できましたが、単語や文など範囲を指定することはできませんでした。しかし、4.0では範囲を指定してから[挿入]-[コメント]を実行すると、範囲が明示されるようになりました。今までいまいち使い勝手が悪かったコメント機能ですが、これで実用的になったといえます。ただし長過ぎると指定できないうえ、DOCファイルとの互換性はなく、またPDFエクスポートすると無効になってしまうので、相互運用性の観点ではまだまだ修正が必要のようです。

最初のページだけ異なったヘッダ／フッタを入れる

論文などの長文を書くような場合、通常最初のページは表紙にするので、ヘッダ／フッタは入れません。しかし、これまでのWriterでこれをやろうとすると[スタイルと書式設定]の機能を駆使する必要があり、マスターしていないとけっこうたいへんでした。今回この機能が実装されたので、最初のページだけヘッダ／フッタを入れない場合、あるいは別の内容にする場合でも簡単に対応できるようになりました。とはいえ、表紙のあとに入れた目次にもヘッダ／フッタを入れたくないといった場合には[スタイルと書式設定]を使用する必要があるので、長文を執筆する機会が多い場合はマスターすることも考えていただければと思います。

Logo

Logoとはいえ Wordアートのようなものではなく、Logoというプログラミング言語をWriterで使用できる機能が追加されました。[表示]-[ツールバー]-[Logo]で表示できます。プログラム勉強用兼デザイン用とのことですが、よく使う点線や破線なども簡単に描くことができます。今のところ日本語で読めるドキュメントがないので取っつきにくいですが、遊ぶのにも実用にも耐えるなかなか興味深い新機能です（**図1**）。

Calc

新規関数のサポート

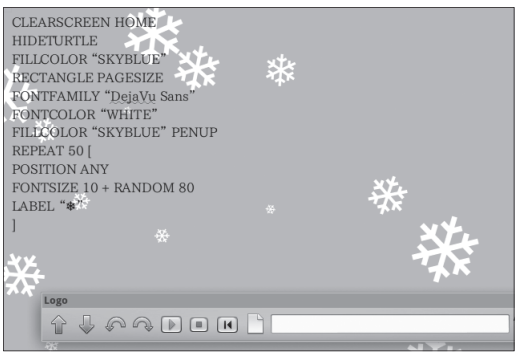
新たにAVERAGEIF、SUMIFS、AVERAGEIFS、COUNTIFS、IFERROR、IFNA[注4]関数がサポートされました。

Calcの関数に関して今回少し詳しく調べてみましたが、Excel 2007と比較するとほとんどの関数をサポートしているといってもいいところまできていま

注3） ライセンスを変更できるのは著作権者のオラクルだけなので、こういった手順を経る必要があります。

注4） これだけExcel 2013からの新関数です。

図1　コマンドを入力し、Logoツールバーの再生ボタンを押したところ



す。バイト数を返す関数（FINDB、LEFTB、LENB、MIDB、REPLACEB、RIGHTB、SEARCHB）のほか、YEN、DATESTRING、NUMBERSTRING、PHONETICといった日本語でしか使用しない関数ぐらいしかサポートしていないものはありません。バイト数を返す関数はODF 1.2に定義されているにもかかわらずサポートされていないので、いずれ実装されるものと思います。日本語関連関数においてはそもそもバグ登録すらもされていないので、誰かが実装する必要がありそうです（**図2**）。

条件付き書式の仕様変更

3.6では［書式］-［条件付き書式設定］は［条件付き書式設定］と［管理］の2つだけでしたが、4.0では［書式］-［条件付き書式］となり、［条件］［カラースケール］［データバー］［管理］の4つになりました。［カラースケール］と［データバー］は、以前はとても使いにくかったのでこれはうれしい変更です。そればかりか、条件も多数追加されており、上位10%や平均より上といった指定ができるようになったので、活用する場面が増えました。

図2　新たに追加されたAVERAGEIFS、COUNTIFSは［数学］に分類されている

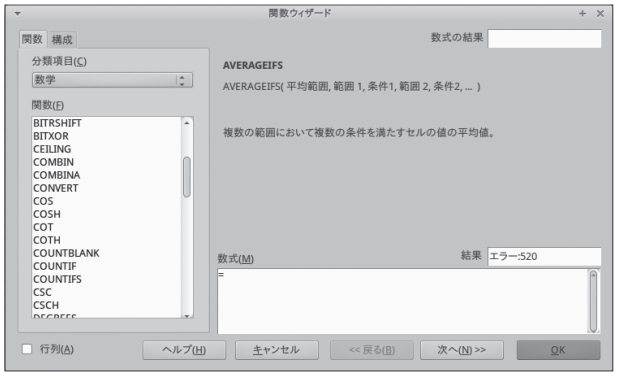


図3　［角の形状］の隣に［頂点の形状］が追加された

ファイル読み込みの高速化

Calcはバージョンアップごとにファイル読み込みの高速化が図られていますが、今回も例外なくODSファイルの読み込みが高速になっています。

Draw/Impress

頂点の形状

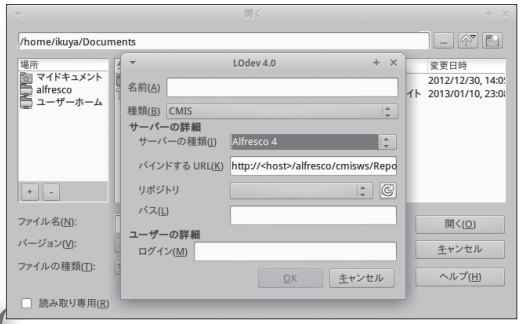
四角や丸などのオブジェクトを作成し、［線］の［線］タブを開くと、これまで［角の形状］だったのが［角と頂点の形状］となり、頂点の形状を選択できるようになりました（**図3**）。

その他全般

CMISサポート

CMISはContent Management Interoperability Servicesの略で、端的にいえばドキュメントサーバの規格です。これに対応するためLibreOffice標準のダイアログが大幅に書き直されました。［ツール］-［オプション］-［LibreOffice］-［全般］の［LibreOfficeダイアログを使用する］にチェックを入れると、このダイアログを使用できます（**図4**）。パスの右横にあ

図4　LibreOfficeのダイアログでは、CMISほかいくつかのプロトコルでリモートのリポジトリにアクセスできるようになった



る[...]をクリックするとサーバに接続するためのダイアログが表示され、[種類]に[CMIS]があります。これを選択すると[サーバーの種類]が表示され、ここから定番のAlfrescoから著名なSharePoint 2010まで、いくつかのCMISサーバを選択できます。今回はあまり詳細には取り上げませんが、いずれこの連載かgihyo.jpのUbuntu Weekly RecipeでCMISサーバとの連携について触れてみたいと思います。

ファックスとレターウィザードをPythonで再実装

LibreOfficeは2011年から[注5]2年連続でGoogle Summer of Code(GSoC)のプロジェクトに採択されており、ここで開発されたコードがさまざまな形でLibreOfficeに取り込まれ、新機能や改善事項として我々も恩恵を受けています。実は今回もいくつかGSoCの成果があったりするのですが(後述のPublisherファイルのインポートもそうです)、これもそれです。ファックスとレターウィザードを動作させるのにJava仮想マシンが必要でしたが、これがPythonに移植されたことによってJavaの依存が少なくなりました。これはとても価値があることだと思います。

画像の圧縮

なぜかWriterにはないのですが、CalcとDraw/Impress[注6]で貼り付けた画像を右クリックすると[画像の圧縮]というメニューが表示されるようになりました。画像の解像度やサイズを小さくしてファイルサイズを軽減するための機能です(**図5**)。大きな画像を貼り付けて縮小するとどうしてもファイルそのもののサイズが大きくなるうえ、縮小だと画質がいまいちになることもよくあるので、この機能であらかじめ圧縮しておくときれいに表示ができていいのではないでしょうか。

注5) その前は組織がなかったので、LibreOfficeができてからずっと……ともいえます。

注6) 内部的にはDrawとImpressは同一で、見た目を変えているだけです。

テンプレートマネージャー

[ファイル]-[新規作成]-[テンプレート][注7]で、これまでの[テンプレートとドキュメント]に代わり[テンプレートマネージャー]が起動するようになりました(**図6**)。ほかのPCにあるテンプレートや、CMISのテンプレートを開くことができるばかりか、サムネイルがちゃんと表示されるので使い勝手も向上しています。

Microsoft Publisherファイルのインポート

リリースのたびに新しくインポートできるファイ

注7) 以前は[テンプレートとドキュメント]でした。

図5 [画像の圧縮]では、こと細かなオプションを設定して可能な限りクオリティを保ちつつファイルサイズの削減ができる

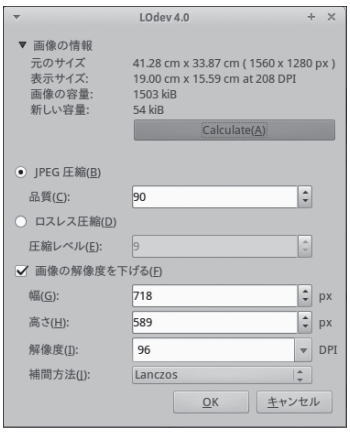
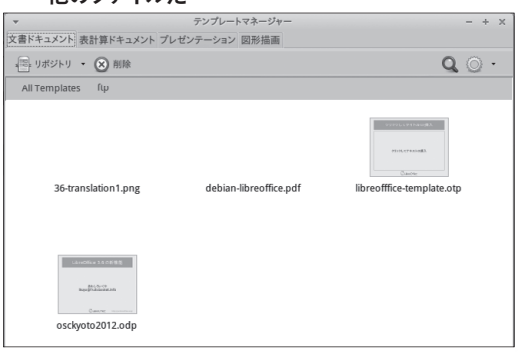


図6 [テンプレートマネージャー]は使いやすくなっただけではなく、FTPなどのプロトコルにも対応した。これはFTPサーバに置いてあるテンプレートとその他のファイルだ



ルを増やしているのですが、今回対象になったのはMicrosoft Publisherでした。やはりこれも日本ではあまり使われておらず、そもそもMicrosoftのアプリケーションとしては相対的に安価で、かつテンプレートがあって初めて価値があるものなので、あまり使う機会はなさそうな気がします。

Visioファイルのインポート強化

3.5で実装されたVisioファイルのインポート機能ですが、これがこのたび強化され、すべてのバージョンのVisioのファイルが読み込めるようになりました。ぜひともお手元のVisioファイルを読み込ませてみてください。

Firefox Personaのサポート

FirefoxのPersonaはいわゆるテーマ機能ですが、驚いたことにこれがLibreOfficeでも使用できるようになりました。[ツール]-[オプション]-[LibreOffice]-[個人設定]を開き、[独自のPersonaを使用]にチェックを入れて[Personaを選択]をクリックします。[FirefoxのPersonaを開く]をクリックするとWebブラウザでPersonaの配布サイトが開きます。インストールしたいPersonaのURLをコピーし、[Persona URL]の下の欄にペーストしてください。オプションを閉じると適用されます(**図7**)。

PDFインポート/Presenter Console/Python Scripting Providerの本体取り込み

タイトルのとおりですが、これまで拡張機能として用意されていたこれらの機能が本体に取り込まれました。Ubuntuはバージョンに応じてこれらの拡張機能を別途インストールしたり、あらかじめインストールされていたりまちまちでしたが、4.0からはそのようなことがなくなります。

あいまい検索の仕様変更

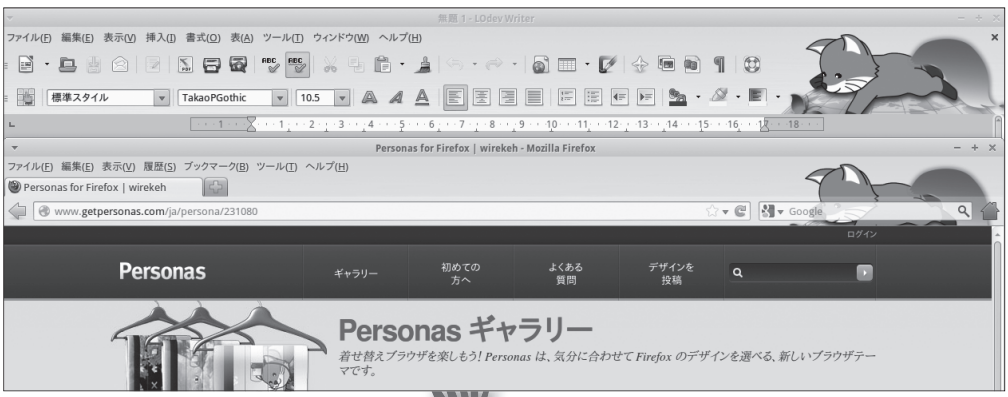
正規表現のエンジンが変更され、バグが少なくなり、高速化されました。ただしそれにあたって日本語のあいまい検索の挙動が変更されている部分があるかもしれない、とのことです。

Ubuntu 13.04とLibreOffice 4.0

Ubuntu 13.04とLibreOffice 4.0のリリーススケジュールを見ると、13.04は4.0.0ないし4.0.1でリリースされそうです。最近のパターンだと必ずしも最新版のLibreOfficeに追随するわけではなく、たとえば12.10では3.6.2のままです。ですので、どの程度バージョンアップされるのかはわかりませんが、何らかの事情で最新版が使いたい場合はPPA[注8]の使用を検討するのがよさそうです。SD

注8) https://launchpad.net/~libreoffice

図7 LibreOfficeでFirefox Personaの[wirekeh]を表示したところ。下はFirefoxで、まったく同じものが表示できていることがわかる



Text=Bart Eisenberg　E-mail jaysteller@hotmail.com　Translation=嶋崎正樹 SHIMAZAKI Masaki

最終回

これまでを振り返って―Software Designerの名言集

Bart Eisenberg

ライター

20年間、ありがとうございました

2010年、日本を訪れる計画を立てていたとき、本誌元編集者の金箱賢太郎氏から「今年はSoftware Designの創刊20周年の記念に読者向けのイベントを計画しています。創刊時から連載していただいているBartさんは今では唯一の創刊時メンバーです」とメールをいただきました。

20年……？　信じられません。連載開始時はまだ若く40歳でしたが、今や還暦を越えました。変化の激しい業界について記事を書くため、努力が欠かせませんでした。編集部から毎月送られてくる本誌は書斎の本棚をいくつも埋め尽くしています。創刊時からの執筆陣で残っているのは「私だけ」になりました。

以前は「Pacific Connection」という連載で、技術やトレンドを紹介する記事を書いていました。編集部からは自由に書いてよいと言われ、トピックは多岐にわたりました。P2Pのファイル交換、パラレルコード、JavaBeans API、スロットマシンの科学、Appleクローン市場の終焉、初期のAjax、BitTorrent、iPhone、Android、RSS、OpenStreetMap、Wikipediaなどなど。その後、もっと人物に光を当てようと新連載が企画されました。それがコードの裏の開発者を紹介する「Software Designer」です。欧米の業界のプロを、日本の同業者、つまり読者の皆さんに紹介する連載です。キャリア、考え方、若いころの着想のヒント、トレンド予想、アドバイス、愛する仕事における課題や想いなどを書こうと思いました。

最初のインタビューでは幸運にも、米国立気象局の環境モデリングセンター所長Steve Lordの話が聞けました。10代のころのロングアイランド・サウンドでのヨットレースの経験から天気予報に関心を持ったこと、アルゴリズム、ハードウェア、チームが作成した数理モデルを支えるコーディングスキル、改良で予報の精度が高まること、それらを学ぶために必要なことなどを話してくれました。この最初のインタビューが、その後40回以上もの取材のモデルになりました。

取材で最も難しいのは取材のアポイントを取ることです。著名なCEOでもすぐに快諾する人がいれば、何ヵ月も待たせる人もいます。断られたり返信がないこともありました。これは日本への関心度に関係していたようです。日本でのビジネスやその計画がある、あるいはマンガやアニメ、スシが好き。そういう人のほうが取材に応じてくれやすい感じでした。私はコーヒーを飲みながら会話するようなインタビューを考えていましたが、実際には相手が遠く離れた場所にいることもありました。シリコンバレーのように近くても私の自宅からは車で数時間かかっ

たり、ヨーロッパ、ニュージーランド、日本などの外国だったり。そのため、自宅の書斎で携帯電話やSkype(私のお気に入りです)を使ってインタビューをすることになりました。

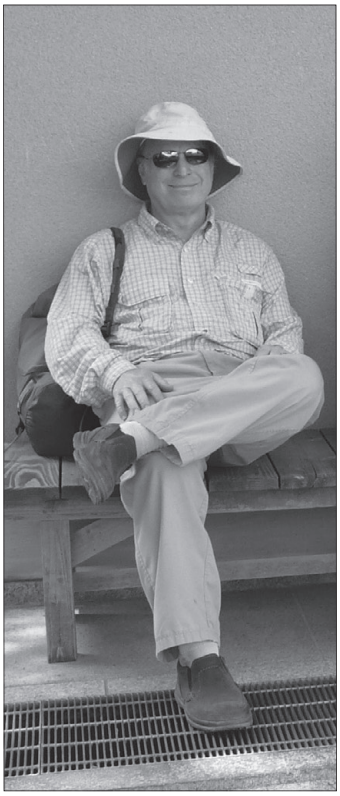
インタビューはすべてMP3で録音しました。妻のSusanがそれを聞いてテキストに起こしてくれました。それをベースに序文とQ&Aという形式の草稿を作り、インタビュー相手に送ってチェックしてもらいました。内容の書き直しや削除を恐れて相手にチェックしてもらわないライターもいます。私の場合は書き直しや削除はなく、結果として相手の素顔をより正確に伝える記事になったと思います。

今回各記事を読み直してみて、洞察力に富んだ話、楽しい話、皮肉な話、暴露話など、いろいろと良い話があったと改めて思いました。最終回の今回はそのような話をいくつかテーマに分けて取り上げてみます。誰もが雄弁に語るわけではありませんが、いくつか箴言があったように思います。それも取り上げました。

2012年夏、シエラネバダにハイキングに出かけたとき、この連載を締めくくる時期がきたと不意に感じました。創刊号から参加している最後の1人として、引き際だと思ったのです。62歳になり、30年間のキャリアを築いてきた仕事の多くを私はすでに辞めています。本連載を終えるのは勇気のいることですが、私は書き手として時間をかけて何かを生み出したいと思うのです。

Rubyの開発者、Matzこと、まつもとゆきひろ氏は、「プログラマの幸福感」について語ることで有名です。今までのインタビューを1本の糸でつなぐものがあるとしたら、おそらくこの言葉に尽きると思います。皆さんにも幸福であってほしいと思います。これまで本当にありがとうございました。

▼2010年の来日時、金沢にて



仕事としてのソフトウェア開発と教育

— Vanessa Hurst(Girl Develop It 共同創設者)

ソフトウェア開発のすばらしいところは成果物自体がものを言う点です。今ではGitHubやBitbucketなどのプラットフォームでコードを共有できるため、開発者の能力を実証できます。Webサイトを構築して公開すれば、開発の仕事に就くことができます。新興企業のコミュニティでは、職歴が浅くスキルアップが必要な人でも、仕事ができると証明できれば重宝されます。

— Steve Lord(米国気象局環境モデリングセンター所長)

米国の大学の場合、環境科学の教育プログラムでは、ブラックボックスとしてのシステムの使い方だけでその中身は教えません。コードを実行して気象予報ができる人は大勢います。入力する内容がわかれば、システムの周辺を変更するだけで済むからです。しかし、より正確な予報結果を出すには、コード内の処理を変更しなくてはなりません。それができるプログラマを探すのは難しくなってきています。そのように教えていないからです。

— Brian Kernighan(プリンストン大学教授「K&R」の「K」)

どの大学でも、学生が授業中にいろいろな作業に取り組んで時間を浪費しています。あまりにもやることが多いため、教師の話を聞いていません。その対策も難しいです。私も「講義に集中する時間」を作ろうとしてきました。その一環として無線通信

を盗聴できるパケットスニッファ（Wireshark）を動かし、無線やネットワークプロトコル、プライバシーに関する技術を教えようと試みました。これは誰が自分を盗聴しているかわからない、同じ教室に盗聴者がいるかも、という教訓にもなります。

—Telle Whitney（アニタ・ボルグ女性と技術研究所 所長兼CEO）

若い女性にとって、コンピュータやエンジニアリングはクールではありません。コンピュータサイエンスを学ぶ能力は十分でも、女性は地に足の着いた選択をします。急成長する産業の8割がITを活用していても、学士号があれば給与の良い仕事に就けるのだとしても、女性はITを職業として選択しません。仕事はすべてアウトソーシングされる、1日中コンピュータの前に座らされるなど、ありもしない話を鵜呑みにする人もいます。女性の多くはIT業界に悪いイメージを抱いているのです。

—Daniel Jackson（MIT教授）

経験豊かなプログラマは忍耐強く物事に当たります。近道をして手痛い目に遭うことが多かったからでしょう。経験が少ない開発者ほど、ものを作るときのワクワク感を楽しみます。ものを作る行為はすごくおもしろいですから。でも、そのワクワク感だけですべてを見てしまいがちになります。なので、学部生に、より大きな視野で物事をとらえ、設計をきちんと行うこと、つまり最初に単に開発するだけでなく、適切な構造を用意して適切な機能をサポートしていくことが重要であると理解してもらうのがとても難しいです。

—Chris Timossi（ローレンス・バークレー国立研究所 元ソフトウェアエンジニア）

将来について考えている学生は、自分が選んだ分野の基礎を理解したうえで大学を卒業したいと思うでしょう。専門分野が多様化し変化も速いために、仕事をするには研究分野が1つだけでは十分と言えなくなっているからです。たとえば、粒子加速器の制御システムに関心があるなら、ハードウェア／ソフトウェアのインテグレータではなく、機械工学や電気の技術者として仕事をするほうがよいかもしれません。

—Chad Fowler（『情熱プログラマー』ほか執筆者）

たいていの人は自分のキャリアについても、「偶然のプログラミング」的なことを行っていると思います。勉強すべきものが見つかるのは、たぶん優れた教師に影響されてのことでしょう。その業界において市場がどのような状態か、キャリアがどのようになるかなど、実際には知らないはずです。仕事に行き、数十年が経ってから振り返ってみて、自分がどこを目指すべきか実際には決断を下していないことに思い当たるのです。

→ コーディングの本質

—David Heinemeier Hansson（Ruby on Railsの開発者）

プログラミングは創造的な作業であり、それだけを半日も続けてできるものではありません。少なくとも私の脳みそは、ピークの状態をそんなに長い時間保てるようにはできていません。たとえば、100%の状態で7時間続けて作業する代わりに、90%の状態で12時間続けるというわけにはいかないのです。生産性は急激に落ちていきます。でも、ピークの状態で作業しているときにしか、最終的な成果をすっかり変えるようなブレークスルーは起きません。

—Andrew Stone（Stone Design創設者兼CEO）

私はいわば「サムライ」型のコーダです。コーディングが大好きなので、作業にとりかかると、「フロー状態」になってしまいます。作業に没頭して時間の感覚がなくなるんですね。早朝4時に起きて、家族が起きるまでの数時間にコーディングをして、子どもたちを学校に送り、1時間ほどヨガをしたあと、ふたたびフロー状態に戻ったりします。ふと気づくと、夕方だったりします。

—Adam Wiggins（Heroku共同創設者兼CTO）

物理的な世界では、エンジニアの仕事は仕様書に沿ってプロジェクトを完成させることです。

ですが、ソフトウェアの世界では、たとえば橋を作り始めたら必要なのはボートだったということが、作業の途中でわかる場合もあります。開発者として、私たちはその切り替えができます。ソフトウェアとはまさに「ソフト」なものだからです。私からすれば、だからプログラミングは楽しいのです。

— Matthew Mullenweg
（WordPress創設者兼開発者）

（ジャズとソフトウェア開発は）どちらもかなり練習をしなくてはなりません。毎日練習が必要です。ミュージシャンとして大成する人は練習のときにはひどい音を出していたりします。耳をふさぎたくなるほどの音です。不得意なパートを練習するからなんですね。聴衆に向かってすばらしい演奏ができることは魅力的ですが、だからこそ、出来の悪い部分を練習しなくてはなりません。それが自分をさらに高める唯一の方法です。

→ ソフトウェアの隠し味

— Adam Wiggins

Android や BlackBerry 対 iOS、Windows対Appleなどの論争を見ると、違いを突き詰めていけば結局は幸福感を得られるかどうかの問題だったりします。技術を使うときにその技術をどう感じるかという問題です。Appleのコンピュータを購入しても、明細書には「幸福感」を表す記載はありません。しかし、幸福感はあるはずです。Appleの重要な売りの1つがそれ、一貫したユーザ体験です。

— Warren Spector
（Junction Pointクリエイティブディレクター）

私はもともと映画が大好きです。本もよく読みます。物語が大好きなんです。1970年代後半に、「Dungeons and Dragons」をプレイしていたライターやコミック作家などのグループと偶然知り合いました。このゲームは、他人の探検ストーリーを体験するのではなく、自分でストーリーを作る点が実におもしろく、その考え方が魅力的で、このような仕事をしたいと思うようになりました。10年後、ビデオゲーム業界に入りましたが、当時は、プレイヤーがゲームを完了するイベントや操作の順番をデザイナーが考えるようなゲームばかりでした。私はそれに対抗し、プレイヤーが自分のストーリーを体験できるように、「Dungeons and Dragons」で のロールプレイングの体験をビデオゲームに持ち込むことを目標

▼阿蘇山にて（2010年）

にしたのです。

→コラボレーション

— まつもとゆきひろ（Rubyの開発者）

Rubyの動きには2つの側面があります。1つは言語の構文と機能、もう1つは人間同士の関係です。人がお互いに「いい人」として振る舞えば、コミュニティも大いに助かると私は思います。「いい人」であることが、寄贈ライブラリ、Ruby on Rails、RubyGemsパッケージシステムなどのエコシステム全体を向上させます。すべては人間同士のある種の関係から生じたものです。このように、私たちには健全な発展を遂げてきた技術的側面とソーシャルな側面があります。Rubyの魅力の1つは、その両方がそろっている点にあるのでしょう。

— Bruce Tate
（『JavaからRubyへ』ほか執筆者）

Matzはいつも楽しむことについて語っていますが、それは言葉以上に広い意味を持っていると思います。彼が作り上げたコミュニティと哲学は、親切心と美徳を持って生きることを示しています。優れたプログラマ、そしてあらゆる分野の優れた人たちがその価値を見出せるでしょう。

— Mark Billinghurst（HITLab所長）

（研究者の加藤博一氏と私は）異なるスキルを持っていました。加藤氏はコンピュータビジョンの研究に優れており、私はインターフェースの設計を得意としていました。2人とも研究熱心で、目標の達成に必要なら時間はいくらでもかける性格でした。文化的な違いはそれほど大きな要因ではないと思います。ニュージーランドは日本とのつながりが深く、彼も家族と一緒に何度か訪れています。私もとても楽しい時間を過ごせました。日本人はニュージーランドを気に入ると思います。どこか日本に似ているところがあります。山の多い島国ですし、海に囲まれているし。ただ、人口は日本ほど多くありません。

— David Weekly
（Hacker Dojo創設者）

（Hacker Dojoは）自宅でも職場でもない第三の場所が求められた結果できた場所と説明するのが一番でしょう。仕事抜きでも込みでもいろいろな人と交流が図れ、何かを作ったりアイデアを交換したりできる場所です。昔から、カフェやサロンはそのような場を提供してましたが、スターバックスは仕事をするには良い場所でも、コミュニティは形成されません。ノートPCを持っている人に近づいて何をしているか尋ねても、「俺はコーヒーを飲んでるんだ。放っておいてくれ」と言われるのがオチでしょう。

— Ward Cunningham
（Wikiの開発者）

Wikiで興味深い点の1つは、Wikiでは通常の共同作業とは手順が逆転することです。普通は共同作業では、レビューを繰り返してからコンテンツの公開に踏み切ります。Wikiの場合、まず公開してからレビューが繰り返されます。これは、コンテンツの公開とレビューのサイクルに要する時間を短縮するという意味で、コラボレーションシステムそのものに大きな影響を与えます。先に公開してからレビューを行うようにすると、物事がスムーズに進みます。

ジャズバンドを組んでいる知り合いは、最初のコードを鳴らした段階で皆がどう感じているかわかると言います。物事がどのように行われているかはすぐ伝わります。このように即時的な感覚はごく自然です。人間は1秒未満で物事を感じ取るようにできているんですね。長々と時間をかけるのは、従来の方法で作業を行っていると、短い時間間隔でのコミュニケーションのとり方がよくわからなくなるからです。昔はカーボン紙でタイプしたメモを使っていました。今はコンピュータネットワークがあるので、作業の速度も上がって当然です。ネットワークはジャズバンドの最初の音ほどではありませんが、高速になっているとは言えます。

— Jim Jagielski（Apache Software Foundation会長）

合意は私たちのDNAです。合意こそが、対立意見に耳を傾け、

恨みを持つことなく自説を放棄できるような開発者を惹きつけてきたのです。そういう人でなければ、ASFの開発プロセスからは脱落してしまうでしょう。ASFでは「悪態シンドローム」についてのプレゼンテーションも行っています。柔軟性のない人が1人いるだけで、いかにソフトウェアプロジェクトが損なわれるかという話を取り上げます。そういう人の考え方をどのように変えられるかを知っておくことが重要です。たとえ世界屈指の開発者でも、コミュニティやプロジェクトに害をなすことは許されないという認識も重要です。

— David Heinemeier Hansson

ときには全員の合意などいらないこともあります。独裁者がいるほうがうまくいく場合もあるのです。「こうやればいいじゃないか。こうしよう」と言う人が必要な場面もあります。全員を納得させたいなんて、思うのは簡単ですが、会議の参加人数が増えれば納得させるべき相手が増え、時間もかかります。

→ 起業について

— Sarah Allen
(Mightyverseチーフギーク)

ベンチャー企業では、現在のチームが将来も継続する可能性が高いことがよいですね。人が入れ替わることはあるものの、ベンチャー企業では、新しいことも社内の同じチームで手がけることになります。大企業では一般に、新製品に投資して旧製品を終了する場合、新製品に適したスキルを持った新しいチームを編成しようとします。旧製品に関わっていた人は解雇されるかもしれません。私が目標とするのは、何か違うことをやろうと決めても、それまでと同じ人たちと行える環境です。

— David Heinemeier Hansson

今できる限りのことをやっておけば、いつかすばらしいことが起きると多くの人が考えています。会社が買収され、巨万の富を手にし、すばらしい人生を送れると。でもそうはなりません。私も多くの起業家の話を聞いてきました。ビジネスで成功し、事業を売却して引退するのに十分な資金を得て、神話となっている「ビーチで1日中マルガリータを飲む生活」をしてきた人たちです。その後はどうなったでしょうか。半年もすると、

▼屋久島にて、奥様のSusanと(2010年)

彼らはまたゲームの渦中に戻ってきます。人はビーチのポテトになるようにはできていないからです。それでは知的な楽しみは得られません。そんなわけで彼らも半年後には復帰します。別のアイデアを持って。でもその新しいアイデアは前のものほどうまくいかなかったりするのです。

— Eric Ries
（「リーンスタートアップ」提唱者）

起業家が下す判断で一番難しいのは、「転換すべきか」「温存すべきか」です。数式のようにはいきません。もう少しこだわって、もう1回繰り返したおかげで、コンセプトがうまくいったという話は聞いたことがあるでしょう。一方で、何度も転換して出口が見えなくなった人もいます。

— Matthew Mullenweg

現実には、リリースするプロジェクトの多くは失敗するか、可能性はあっても成功までには至りません。しかし、世に出さなければ、完成させるために必要なフィードバックは得られません。「iPhone 4」がその良い例です。初期のiPhoneはそれほど優れた製品ではありませんでした。速度も遅く、動作も洗練されていませんでした。独自の美学はありましたが、転換期を迎えたのは第4世代になってからです。すばらしい成果を上げましたが、それは4回も繰り返したすえにやっとたどり着いたものです。

→ オープンソース

— David Heinemeier Hansson

37SignalsがRuby on Railsに膨大な時間をかけているのは当然でしょう。Ruby on Railsはビジネスを構築するためのツールだからです。ビジネスそのものがなかったら、努力はまったく無意味になります。問題が発生するのは、博愛の精神を持ったオープンソースの世界から、利益指向型の商業の世界に参入するときです。もちろん、毎日仕事が終わったら、誰でもご飯が食べられるようでなければなりません。単に才能を垂れ流ししているだけでは、魔法でもない限り食べてはいけません。

— Gianugo Rabellino
（Sourcesense創設者兼CEO）

オープンソースにコントリビュートするときには、シャイな自分と、笑われるのではないかという恐れを克服しなくてはなりません。言語を使いこなせないせいで、言いたいことを伝えられないと感じるのは嫌なものです。私自身にもそのような古傷があります。私はプログラミングを独学しましたが、英語を学校で学んだこともありません。でも、言葉の問題はいつでもついて回ります。また、ヨーロッパの人は多少とも自国に誇りを抱いています。英語なんか話したくない、自分の国の言語で話したいと思ってしまうのです。

→ プログラミング言語

— Bruce Tate

私はBasicが大好きですが、賢い言語なのにあまりよく思われていません。大学でCを学び始めたときに、半分死んだような気分になりました。

私たちは30年以上も前から、C/C++をベースにした言語でアプリケーション開発を行ってきました。Javaの構文のベースとなっているのは、もともとOSを構築するために作られた言語です。私は言語にも賞味期限があると考えています。Java仮想マシンは確かにすばらしいし、世間ではほかにも興味深いことが起きています。でもアプリケーション開発では、Javaがつねに使われてよかったわけではありません。

— Dave Thomas
（『達人プログラマー』ほか執筆者）

私はRubyがすべてに通用する言語だと思っているわけではありません。Rubyで書こうとは考えない種類のソフトウェアもありますから。もちろん、書いてみたいとまったく思わないソフトウェアもあります。

私が興味を持ち、楽しいと感じたのは、Rubyが既存の概念を実にうまく統合していたからでした。たとえば、Rubyのオブジェクトのモデルは、Smalltalkにとてもよく似ています。Matzは言語マニアで、他の言語からいろいろと寄せ集めたものを縫い合わせ

て、つぎはぎの怪物のような言語を作り上げました。Matzのすごいところは、まったく異なる材料から使える言語を編み上げたことです。PerlとSmalltalkなんて、言語的にはまったく似ていません。Matzはその2つをうまくカップリングさせ、優れた子どもを生み出しました。これを実現するには多大なスキルが必要です。

— Vanessa Hurst

私が手がける仕事の多くでは、それほど効率性の高いコードを書く必要はありませんが、Cを半年ほど活用した経験は貴重だったと思っています。そのときの経験から、高水準言語でプログラムを書くときは、つねにデータ構造や最適化を意識するようになりました。

→ 日本の開発者、起業家

— Phil Libin
(Evernote Corporation CEO)

なぜ日本にはシリコンバレー型の起業指向の風土がないのか不思議に思います。新興企業はそれほどなく、米国に比べると数が非常に少ないです。起業に必要なことは3つ。まずはクリエイティブなエンジニアリングの才能です。日本には十分にあります。2つめは、何年かは1日16時間労働もやむを得ないという献身的な職業倫理観です。明らかにこれも十分すぎるほどです。おかしくなるほど毎日長時間働いている人がたくさんいますから。3つめに必要なのは健全な投資の風土です。あるアイデアをもとに起業し、1,000万ドルを得るような環境です。これは日本にはありません。日本では、ベンチャーキャピタルもはるかに消極的です。

— Cyan Banister
(エンジェル投資家)

日本の文化では、誰も目立ってはいけないそうですね。そのように調和を図るとか。でも私は個人的に、皆さんに目立ってほしいと思います。ベストな自分になってしかるべきだし、思い描いたことはすべてやってみてほしいです。どんなに大きな夢でもトライすべきでしょう。私も、成功の陰にはその10倍もの失敗があります。だからといってやめる気にはなりません。失敗を通じて私はもっと強くなれるからです。失敗を糧にすればよいのです。「他にどうしろというの？」と思いますね。

— Raju Vegesna
(Zohoエバンジェリスト)

日本の開発者はプロジェクトの細部まで綿密にこだわる傾向にありますね。事前のプランニングや文書・資料に注意を払います。一方、インドは、基本的な機能さえコーディングできたら、後は性急に先に進んでしまいがちです。理想的には両方を併せ持つのがよいでしょう。機能を実現したら、時間をかけてそれを完璧に仕上げる。日本の開発者がインドのオフィスを訪れるときの利点がそれです。開発者は1、2ヵ月滞在することが多いので、スキルを補完し合って強力なチームができあがります。もう1つの利点として、国際化に関してもっと考えるようになります。中国についてもそれは同じです。

— Danese Cooper
(Wikimedia創設者兼CTO)

日本のプログラマは、完全にベストな状態で成果を出そうとしているのでしょう。それはオープンソース的には価値がありません。コードを公開する場合、必ず「この部分がまずいことはわかっていますが、これはあくまで試作品です」とコメントを添えます。こういうアプローチのほうが会話を導きやすいと思います。コーディングにまだそれほど時間をかけていないからです。

— Eric Ries

日本の起業家と投資家を相手に講演したことがあります。その多くはまだ若く、誰もがこんな質問をしてきました。「そのアイデアを日本の流儀と両立させるにはどうしたらよいでしょうか」。私は、このアイデアはもともとトヨタの生産方式を参考にしたものだと答えました。彼らはクレイジーだと言わんばかりに私を見ました。平均的米国人よりクレイジーだという感じでした。彼らは、日本の流儀は官僚主義的なトップダウン方式で、革新をもたらさないものだと思っ

ていたからです。日本の若い起業家は、トヨタのような日本企業は創立当時の敏捷さを失っていると感じていたのですね。それは経営全般の典型的な行動パターンです。何かに秀でるようになると、それだけをもっと良くすることに集中しがちです。

過去を振り返って

— Marc Weber（コンピュータ歴史博物館創設キュレーター）

インターネットの前身であるArpanetに関連する展示品の重さを量ろうとした政府の査察官の逸話があります。その人はArpanetを率いていたBob Taylorに、「ソフトウェアはどのくらいの重さになるのか」と尋ねたのです。Taylorは、「重さはない」と答えました。形があるものではないですから。査察官は1週間後にまた訪れて、「パンチカードが入った箱が詰まった部屋を見つけた、カードを集めたら数トンになった」と報告しました。Taylorは「確かにそうだが、ソフトウェアはその穴の部分に存在しているのだ」と答えたのでした。

若い映画監督や作家は、過去の調査や古い作品の鑑賞に多くの時間を費やします。しかし、アカデミックなコンピュータサイエンスのプログラムを見たところで、どのようなことが行われていたかはほぼ皆目わからないのです。ネットワーク上の情報になると、誰しもほとんど記憶していません。

— Elizabeth “Jake” Feinler（インターネットのパイオニア）

インターネットは電気に似ています。たった1つの電球、ネットではUCLAとSRIのわずか2つのホストから始まったものが、今や世界全体を照らしているのですから。

未来を見据えて

— Steve Russel（ゲーム「Spacewar!」考案者）

将来がどうなるか考えたり、過去の経験から教訓を引き出すことはなかなかできません。むしろ、今どんなおもしろいプロジェクトを手がけられるだろうかということに関心があります。優れたプロジェクトは優れた物語に似ていると思います。ときには対立があったり、乗り越えるべき障壁や、予想もしていなかった展開があったりもします。そして、予想もしていなかった結末も。

— Jon Bentley（Avaya Labs Research功労メンバー）

直面する問題に対応できるようでありたいと望むなら、必要に応じた小さな荷物だけを持つようにすべきです。それが人生を生きる効果的な方法だと思います。SD

▼自室にて

Hack For Japan
エンジニアだからこそできる復興への一歩

Hack For Japan

第15回 気象データハッカソン

“東日本大震災に対し、自分たちの開発スキルを役立てたい”というエンジニアの声をもとに発足された「Hack For Japan」。本コミュニティによるアイデアソンやハッカソンといった活動で集められたIT業界の有志たちによる知恵の数々を紹介します。

●Hack For Japanスタッフ
関 治之 Hal Seki
Twitter @hal_sk

社会的課題をテクノロジで解決するコミュニティHack For Japanではハッカソンを中心にさまざまな活動を行っています。今回は、気象データを使ったハッカソンについて報告させていただきます。

気象データハッカソンとは？

気象データハッカソンでは、気象庁の協力のもと公開されている気象データおよび他のデータを組み合わせて活用することにより、新たなサービスに関するアイデアを得て試作品を開発します。そしてそのことにより、広くオープンデータの意義や可能性を世の中にPRすることを目的としています。オープンデータ流通環境の実現に向けた基盤整備の推進を目的に産官学が共同で運営する組織、「オープンデータ流通推進コンソーシアム」が主催し、Hack For Japanがハッカソンのオーガナイズを行い2012年12月1日に開催されました。

オープンデータによる社会変革については私個人的にもテーマにして活動している分野であり、過去にもInternational Space Apps Challenge（本誌2012年7月号 連載第7回に掲載）、オープンデータハッカソン（同2012年10月号 連載第10回に掲載）のオーガナイズをやらせていただいています。

オープンデータとは？

オープンデータとは、直訳すると文字どおり「オープン」なデータということになりますが、Open Knowledge Foundationが公開しているオープンデータハンドブック[注1]によると、“オープンデータとは、自由に使えて再利用もでき、かつ誰でも再配布できるようなデータのことだ。従うべき決まりは、せいぜい「作者のクレジットを残す」あるいは「同じ条件で配布する」程度である。”とあります。より具体的には、「利用でき、アクセスができる」「再利用と再配布ができる」「誰でも使える」のがオープンであることの定義とされています。

この定義のポイントは、「相互利用性」にあります。データをオープンに公開し相互利用性を高めることで、さまざまなデータセットを組み合わせることが可能となるからです。データセットを公開することで、その組織がもともと目的としていた利用用途以外の価値が生まれたり、新たなイノベーションが生まれることがオープンデータの狙いと言ってもいいでしょう。政府や自治体がデータプラットフォームになり、民間や大学がそのデータを使ってより良いアプリケーションを作るというわけです。

ワールド・ワイド・ウェブ（WWW）の生みの親であるティム・バーナーズ＝リーは、2009年のTEDで、「生のデータを今すぐに」という呼びかけを政府や科学者などに行い、翌年の2010年には、「オープンデータとマッシュアップで変わる世界[注2]」という有名な講演を行い、オープンデータによる社会的イノベーションの事例を数多く発表しています。

「オープンデータ」という言葉を使う場合、「オー

注1）Open Data Handbook URL http://opendatahandbook.org/
注2）ティム・バーナーズ＝リー「オープンデータとマッシュアップで変わる世界」URL http://www.ted.com/talks/lang/ja/tim_berners_lee_the_year_open_data_went_worldwide.html

プンガバメントデータ」という意味合いを持っていることが多いです。政府は税金を利用して大量のデータを日々作成していますが、そのデータは国民の資産であり、可能な限り公開されるべきであるという「Open By Default」という考え方が欧米を中心に広がってきているのです。

日本のオープンデータの状況

日本ではこの分野ではこれまで欧米に後れを取ってきましたが、震災時に政府や自治体などのデータがうまく活用されなかった反省などに後押しされ、最近にわかに機運が高まってきています。2012（平成24）年7月4日、政府の高度情報通信ネットワーク社会推進戦略本部（IT戦略本部）が「電子行政オープンデータ戦略[注3]」という文書を発表しました。序文に「オープンガバメントの推進に当たっては、公共データは国民共有の財産であるという認識の下、公共データの活用を促進するための取組に速やかに着手し、それを広く展開することにより、国民生活の向上、企業活動の活性化等を図り、我が国の社会経済全体の発展に寄与することが重要であるため、公共データの活用促進のための基本戦略として、『電子行政オープンデータ戦略』を以下のとおり策定する。」とあるとおり、政府としてオープンデータ化を進めていくという戦略です。

IT戦略本部の下部組織として、「電子行政オープンデータ実務者会議」という組織が置かれ、ルール整備や普及のための議論、技術的なフォーマットや公開の仕方などを検討中です。筆者はこのオープンデータ実務者会議の中の、ルール・普及ワーキンググループの委員も務めています。

また、経済産業省や総務省が積極的に実証実験を開始しており、たとえば経済産業省はCKANというオープンソースのデータカタログソフトウェアを利用して省内のデータをオープンに提供していく、オープンMETIという取り組みを始めようとしています。

注3）URL http://www.kantei.go.jp/jp/singi/it2/denshigyousei.html

オープンデータに関する最新ニュースはオープンデータの普及を目指す組織である「Open Knowledge Foundation Japan[注4]」のサイトが詳しいので、一度アクセスしてみていただければと思います。

気象データハッカソン

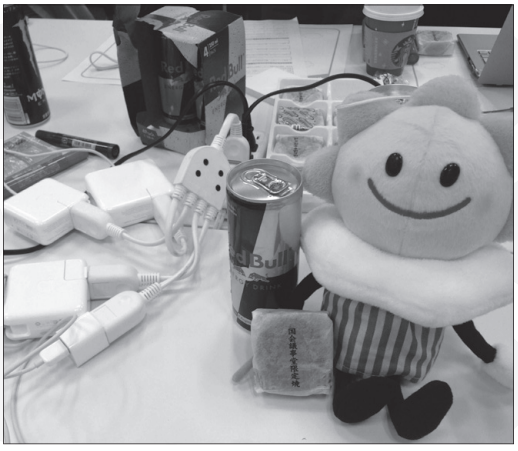
前置きが長くなりましたが、上記のような背景のもと、オープンデータの具体的な活用方法を探るために気象データハッカソンが開催されました。

今回のハッカソンでは気象庁にご協力をいただきました（**写真1**）。気象庁は全省庁の中でもかなり早くからデータの電子化を始めた省庁であり、今では大量のデータを持っていますし、ホームページでも予報データなどを公開しています。しかし、天気図などは閲覧できていても生データとなると、あまり使いやすい形式では取得ができません。一部のデータについては、申請を行えばDVDなどの媒体でデータを入手できたりしますが、API形式で提供されたりはしていません。

オープンデータ流通推進コンソーシアムのページやHack For JapanのFacebookページ上などで告知を行い、50名ほどが集まりました。事前にFacebookグループを作り、アイデアソンも実施し

注4）URL http://okfn.jp/

◆写真1　気象庁のマスコットキャラクター、はれるんとケーブルとレッドブル



ていますが、そちらには170名ほどが参加し、40以上のアイデアが投稿されていました。

ハッカソンの模様

当日は、まずは気象庁の担当者から、気象庁のことや気象庁が扱っている気象データについて、法令などの解説を行っていただきました。参加者全員の自己紹介の後、アイデアを持っている人が内容を発表、それぞれに賛同者が集まることでチームを形成しました。

最終的に次の6チームに分かれハッカソンを実施しました。

①「おしゃれ予報」チーム
お出かけ先と気候、手持ちの洋服をもとにお勧めの服装をアドバイス

②「住みよいマップ」チーム
気候や生活利便性、災害リスクなどのデータを地図上に可視化

③「満ち干きマップ」チーム
浜辺の潮の満ち干きを可視化し、海水浴や潮干狩りなどに活用

④「体質ナビゲーション」チーム
本人の体質とその日の気候、予定などをもとにアドバイス

⑤「Crowdmapと地図のマッシュアップ」チーム
既存のサービス「Crowdmap」にさまざまな気象データをマッシュアップ

⑥「統計データ×気象データ」チーム
消費支出などの統計データと気象データの相関を分析・可視化

今回はエンジニアがあまり多くなく、ハッカソンへの参加自体もはじめてという方が結構いたにもかかわらず、皆さま活発に議論をしながらサービスイメージを作っていきました。気象庁の方もグループに入り、そのままの形式では使いにくいような複雑なフォーマットのデータから必要なものだけを抜き出す作業などをお手伝いいただきました。

3時間ほどの作業を行い、その後各チームからの発表を行いました（**写真2**）。さすがに時間が短くシステムの完成まではいきませんでしたが、住みよいマップチームやCrowdmapと地図のマッシュアップチームは、ある程度のプロトタイプを作ることができました。他のチームも、システムとしてどのよう

◆写真2　ハッカソンの模様

な機能をもったサービスを作るのかを示すことができておりました。

最後に参加者の中での投票を行い、最優秀プロジェクトに決まったのが体質ナビゲーションチームでした。熱中症や高血圧などになりやすい個人の体質情報と特定エリアの天候を掛け合わせることで、自分にとって症状が出るリスクが高い地域が地図上でわかるというものです。これにより、外出先のエリアが過ごしやすい場所か、住みやすい場所かなどを知ることができるわけです。実際に気象データを使って動くシステムやデモを作るまでには至りませんでしたが、有用性が評価され最優秀となったのだと思います。発表の後に気象庁の方や参加者から、「どこからデータを取ってこれそうか」などのアドバイスがあり参考になりました。

その後行われた懇親会にも多くのメンバーが参加し、有意義な交流を行うことができました。

オープンデータをテーマにしたハッカソンの意義

開催後も気象データハッカソンのFacebookページ[注5]では意見交換がされており、参加者も増えて200名を超えています。気象庁の担当者も参加しており、開催当時には公開されていなかった、気象庁データのXML配信システム[注6]の解説なども行われています。

オープンデータを進めるうえでの課題の1つに、「提供する側も使う側も、最適な手段や範囲をわかっていない」というものがあります。提供する側は、データを再フォーマットするにもサーバを立てるにもコストがかかってしまいます。一方、データを使う側も、そもそもどんなデータがあって自分たちのどのようなサービスに使えるのかの理解があまりありません。双方の対話なしにはとても最適解は見つからないのです。

ハッカソンを開催することで、両者に具体的なサービスを意識した対話が生まれます。データ提供側はより適切な提供方法を考えることができ、使う側はどのようなデータがあるかを知ることにより、新たなサービスや既存サービスを改善するための着想が生まれます。

対象ドメインに対するスペシャリストやクリエイターを緩くつなげられるコミュニティビルディングという意味でも、ハッカソンとFacebookグループの組み合わせは有効であると考えています。

一方、半日ではできることが限られてしまっているため、正直なところアウトプットとしてはあまりたいしたレベルにはなりません。チームとしてもハッカソンが終わったら解散してしまうのがほとんどであるため、ハッカソンで発見したコンセプトをどう育て、サービス化までつなげるかという部分は依然として課題になっています。何か良いアイデアをお持ちの方は、ぜひ筆者までご連絡いただければ幸いです。

あなたの協力が必要です

オープンデータを使って良いシステムを作るのはエンジニアであることから、政府のオープンデータの第一のユーザはエンジニアであるとも言えます。オープンデータによる社会変革にはエンジニアの協力が不可欠です。本書の読者にもエンジニアが多いと思います。ぜひ力をお貸しください。

一方、「私はエンジニアでない」という方でも活躍の場はたくさんあります。デザイナーや学者、PR企画、アントレプレナーなど、さまざまなタイプの方が参加することで参加者の多様性が生まれ、そこからイノベーティブなアイデアが生まれます。ぜひあなたのアイデアで世の中をより良い方向にするきっかけを生み出していただきたいと思います。

Hack For Japanでは、今後オープンデータをテーマとしたハッカソンをどんどん行っていきます。ぜひHack For Japanのホームページ[注7]やFacebookグループ[注8]などを通じてハッカソンに参加してください。SD

注5）URL https://www.facebook.com/groups/549985991685416/
注6）気象庁防災情報XMLフォーマット形式電文の公開（試行）について URL http://xml.kishou.go.jp/open_trial/index.html
注7）URL http://hack4.jp/
注8）URL https://www.facebook.com/groups/hack4jp/

温故知新
IT むかしばなし

第20回

コンピュータ科学

北山 貴広　KITAYAMA Takahiro　Twitter：@kitayama_t　kitayamat@gmail.com

はじめに

今回は、筆者が所有する古典的な書籍を中心に、コンピュータ科学に関するお話をしたいと思います。

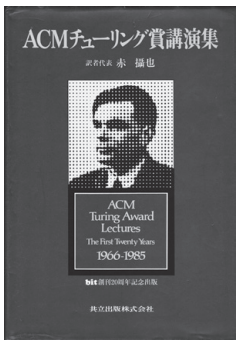
ACMとチューリング賞

ACM（Association for Computing Machinery）は、米国を中心とした世界的なコンピュータ科学（Computer Science）の国際学会です。コンピュータ界のノーベル賞と呼ばれる「チューリング賞」や、2011年に理化学研究所の「京」や東京工業大学の「TSUBAME2.0」が受賞したHPC（High Performance Computing）分野で優れた実性能と成果をたたえる「ゴードン・ベル賞」を授与する組織として知られています。

またACMには、幅広いコンピュータの特定分野における会議や出版などの活動を行うSIG（Special Interest Group）があり、その中でもSIGGRAPHと呼ばれる毎年夏にアメリカで開催されるCGをテーマにした国際会議が有名です。そして、コンピュータ分野での最新の技術や研究の動向を知るのに有用な月刊誌の『Communications of ACM』を発行しています。

ちなみに「チューリング賞」は、イギリスの数学者でコンピュータ科学者でもあるアラン・チューリング（Alan Mathison Turing）氏の功績をたたえて名付けられました。彼は「計算」と「アルゴリズム」の概念を定式化したチューリングマシンを考案して今日のコンピュータの基礎を作り、第二次世界大戦中にドイツ軍の暗号解読を行った人です。

▼写真1　ACMチューリング賞講演集



ACMチューリング賞講演集

『ACMチューリング賞講演集』（**写真1**）は、共立出版がかつて発行していた『bit』という雑誌（コンピュータ・サイエンス誌とタイトルがついていた）の創刊20周年を記念して、1989年に共立出版より出版されました。

チューリング賞が制定された1966年から1985年までの20年間の受賞記念講演22編をすべて翻訳して収録しています。ハードカバーでとても立派な装丁だったためか、価格は9,450円と少々値が張りました。現在のようにインターネットが普及していないころに、コンピュータ科学の分野で権威あるチューリング賞の内容を日本語で読むことができるという貴重な情報だったこともあり、かなり迷った末に思い切って購入して、今日まで大事に持っている1冊です。

最新のチューリング賞に関する情報は、ACMのサイト[注1]に受賞者の経歴などが詳しく掲載されており、翻訳前のオリジナル原稿の受賞講演も“ACM TURING AWARD LECTURE”としてPDFで掲載されています。

アルゴリズム+データ構造=プログラム

『アルゴリズム＋データ構造

注1）http://amturing.acm.org

＝プログラム』[注2]は、Pascalを作ったことで有名なニクラウス・ヴィルト氏(※1984年にチューリング賞を受賞)の著作で、1979年9月に初版が発行されました。約400ページの古典的な書籍で、教科書に使われるようなハードカバーは、約30年経った今でも良い質感を保っています。

残念ながらカバーは捨ててしまったのですが、とても目を引く明るい派手なオレンジ色で印象的でした。この書籍は、「日本コンピュータ協会」という立派な発行元と、「本郷」という所在地と、そこそこ高価な値段だった印象があります。本書は「コンピュータ・サイエンス研究書シリーズ40」となっているので、このころに発行されたコンピュータ関連の書籍としては、かなり立派なシリーズものでした。

章構成は、基本的データ構造、ソーティング、再帰的アルゴリズム、動的データ構造、言語構造とコンパイラで、現在のプログラミング理論のベースとなっている古典だと思います。中のプログラムはPascalで記述されています。1980年当時は教育用プログラミング言語としてPascalが主流であったため、このころのアルゴリズムの解説はPascalを使って説明されたものが多いです。

プログラミングの心理学

『プログラミングの心理学』(**写真2**)はジェラルド・M・ワインバーグ氏の著作です。彼はソフトウェア開発における人間的な振る舞いという独特の視点をたくさん書籍を出されていて、この書籍以外では『ライト、ついていますか－問題発見の人間学』が、そもそも何が誰にとっての問題なのかを客観的に考える習慣がつくのでお勧めです。

『プログラミングの心理学』は、訳書が何回も別の出版社から発行されています。時代を超えて普遍的な事項を扱っているのが理由でしょう。

今でも強烈に記憶に残っているエピソードが2つあります。1つは、“性格の変化”というもので、日頃温厚な人が怒りっぽくなった原因が“歯が悪かったため”でした。

もう1つは学校において優秀で派遣された人が、現場で成績不良になってしまったという話です。原因は1,500マイル離れた妻が癌で集中できなかったのでした。個人的な理由をどう扱うかは難しいですが、いろいろなことにおいて、本当の原因は別のところにあってそれをキチンと追い求めて対応するのが望ましいという、筆者の今の行動原理にもつながっています。

Code Complete

ほかに比べると全然古典的書籍ではないのですが、筆者がプログラミングに関してある意味すごく開眼した書籍として紹介したい書籍が『Code Complete』です。過去に『プログラム書法』などで取得したエッセンスがすべて折り込まれており、なぜそうなるかのWHYの説明が随所にあるためとても納得できました。プログラミングに関して今までに読んだ本での知識が、この本で整理されて身についたと考えています。

終わりに

現在筆者の手元に残っている書籍は、保管スペースなどの制約で何度も捨てられる機会を逃れて生き残ってきました。コンピュータ関連の古典的名著として紹介されると、時間が経っても変わらないベースとなる原典というものがあると実感しています。現在では絶版になっているものも少なくありません。

世界的に普及している電子書籍専用デバイスのKindleが昨年日本国内で発売されるなど、電子書籍が一般的に普及して使われるようになってきていますが、こういった古典的な名著こそ、電子書籍として生き返って現在のエンジニアが読むことができるようになってほしいと思います。SD

▼写真2　プログラミングの心理学

注2）片山卓也訳。日本コンピュータ協会発行。

この個所は、雑誌発行時には記事が掲載されていました。編集の都合上、
総集編では収録致しません。

この個所は、雑誌発行時には記事が掲載されていました。編集の都合上、
総集編では収録致しません。

Service

IDCフロンティア、クラウドサービスのSLAを99.999%に向上／拠点分散型クラウドストレージサービスを先行提供開始

クラウドサービス全ラインアップのSLAを99.999%へ

㈱IDCフロンティアは、クラウドコンピューティングサービス「IDCフロンティア クラウドサービス」および「IDCフロンティア プライベートクラウドサービス」のSLA（Service Level Agreement：品質保証制度）を99.995%から99.999%へサービスレベルを向上し、4月1日より提供を開始する。

同社は2012年4月にも、SLAを99.99%から99.995%にレベルを向上させており、今回はそれをさらに上回るレベルの引き上げとなる。

過去半年間の稼働実績平均値が99.999%を超えていることから、仮想マシンとインターネット接続可用性の稼働率をそれぞれ月間99.995%から99.999%へ品質保証のレベルを向上させることとなった。

「IDCフロンティア 分散ストレージサービス powered by Yahoo! JAPAN」を先行提供開始

㈱IDCフロンティアは、国内の複数データセンター間でデータを分散し世界最高水準の堅牢性と可用性を持つ「IDCフロンティア 分散ストレージサービス powered by Yahoo! JAPAN」（以下、IDCF分散ストレージ）をヤフー㈱と共同開発した。今春の正式サービス開始に先立ち、1月29日より先行提供を開始した。

IDCF分散ストレージは19ナイン※1の堅牢性と、6ナイン※2の可用性、また、他社サービスと比較して最大約6倍のパフォーマンスを誇るオブジェクトストレージサービス。

同サービスを構築するにあたって、同社が2012年7月に戦略的資本提携を行った、米国Basho Technologies,Inc.（以下、Basho社）が持つ、分散型データベース技術を用いたAmazon S3 API互換のストレージ基盤「Riak CS」を採用している。

同サービスは、拠点内でデータを多重化保存していることに加え、西日本拠点および東日本拠点の2拠点でデータを自動分散しており、高い堅牢性と可用性を併せ持っている。

利用者の高いニーズに応えるため、まずは1月29日より無料でAPIとSDKの提供を開始する。Web上で操作を行うコントロールパネルとサービスへの課金は今春の正式サービスから開始を予定している。さらに、将来的には国内外の他拠点追加や分散データベースサービスの展開も検討しているという。

■IDCF分散ストレージの特長

①拠点内の多重化保存と複数データセンター間のデータ複製による高い堅牢性と可用性
②主要ストレージサービスと比較して最大約6倍のパフォーマンス
③Amazon S3と高い互換のあるREST形式Web APIを提供
④IDCフロンティアのデータセンターおよびクラウドサービスとの拠点内ダイレクト接続（提供予定）

※1　19ナインは99.9999999999999999%。
※2　6ナインは99.9999%。

CONTACT ㈱IDCフロンティア
URL http://www.idcf.jp

Topic

The Linux Foundation、「LinuxCon Japan 2013」と「Automotive Linux Summit Spring 2013」の発表者を募集

The Linux Foundationは、5月27～28日に開催される「Automotive Linux Summit 2013（以下、ALS）」と、5月29～31日に開催される「LinuxCon Japan 2013」（会場はいずれも椿山荘（東京都））の発表者を募集している。

ALSは自動車におけるLinux活用をテーマとしたイベントで、自動車関連の専門家やオープンソースエキスパートが集まる。LinuxCon Japanはアジア地域における最大のLinuxカンファレンス。コア開発者、管理者、ユーザ、コミュニティマネージャ、業界専門家が一堂に終結する。いずれもさまざまなプログラムが用意されており、Linuxの関係者同士が交流するよい機会となっている。

両イベントとも募集しているプログラムは、50分間のプレゼンテーション、パネルディスカッション、BoF（分科会）、および2時間のチュートリアル。応募の締め切りは2月28日深夜11時55分。

応募内容の詳細と発表案の提出は両イベントのサイトを参照のこと。

CONTACT The Linux Foundation
URL http://www.linuxfoundation.jp

Hardware

サンワサプライ、
2色のLEDを選択できるバックライト付きキーボードを発売

サンワサプライ㈱は、USB接続に対応したUSBキーボードの新モデル「SKB-WAR2」を1月上旬に発売した。

2色（ブルー／ホワイト）のLED色を選択できるバックライト付きのキーボード。LEDバックライトにより暗い部屋でも確実に文字を認識できる。キーストロークが深く確実なキータッチが得られるメンブレンタイプを採用し、激しいキータッチを抑える滑り止めを2カ所設けている。

キー数は109キー（日本語）、キーピッチは19mm、キーストロークが2.0±0.1mm、動作力が50±5g。本体サイズは467（幅）×25.7（高さ）×173.8（奥行）mm、重量は約550g、ケーブル長は約1.5m。Windows 8/7/Vista/XPを搭載したPCのほか、PlayStation 3やWiiでも利用できる。価格は6,090（税込）円。


▲SKB-WAR2

CONTACT サンワサプライ㈱
URL http://www.sanwa.co.jp

Hardware

エバーグリーン、
手首に負担のかからないエルゴノミクスデザインの光学マウスを発売

インターネットショップ「上海問屋」運営する㈱エバーグリーンは、Windows 8とMacに対応したLEDストレッチングレーザーマウス「DN-46382」を発売した。

腱鞘炎になりにくいエルゴノミクスデザインのマウスで、手首をひねる動作が少なくなるよう設計されている。自然な角度でにぎることができるため、長時間使用しても手首に負担がかからない。

2ボタン、2サイドボタン、ホイール（スクロール）ボタン、カウント切替えボタンを備えている。ストレッチライト切替スイッチの長押しでライトのON/OFFを切り替えたり、ライトの点灯時にDPI切替／カラー変更スイッチを押すことで、LEDライトのカラー（グリーン／ブルー／パープル／レッド）を変えられる。

対応OSはWindows 8/7（64bit/32bit）、Vista、XP、Mac OS X（10.2以降）。価格は2,999円（税込）。


▲DN-46382

CONTACT 上海問屋
URL http://www.donya.jp

Software

エクセルソフト、
Windows 8に対応のデバイスドライバ開発ツール「WinDriver v11.20」を発売

エクセルソフト㈱は、Windows 8に対応したJungo社のUSB/PCI/PCI-Expressデバイスドライバの開発ツール「WinDriver v11.20」を1月23日より販売開始した。

WinDriverは、USB/PCI/PCI-Expressのデバイスドライバをユーザーモードで開発できるツールキット。OSの内部構造や、カーネルレベルのプログラミング知識がなくても開発できるのが特徴。Windows 8/7/Server 2008/Vista/Server 2003/XP（x86 32bitまたはx64 64bit）、Windows CE.NET/MobileおよびLinuxに対応し、WinDriverで開発したコードは対応するOS間で互換性がある。

同製品には短期間でドライバを開発できるウィザードによるグラフィカルな開発環境、API、ハードウェア診断ユーティリティ、サンプルコードが含まれる。

同製品のカーネルドライバが、ハードウェアへのアクセスを提供するので、インストール後、すぐに対象のデバイスと通信が可能。常に最新のOSをサポートするようにアップデートも行っていく。

価格は51万5,550円（税込）～。

CONTACT エクセルソフト㈱
URL http://www.xlsoft.com/jp

Letters from Readers

情報安全護符シール、使っていますか？

2013年1月号では付録として法輪寺電電宮情報安全護符シールを付けました。シールの付録は初めての試みでしたが、いかがでしたでしょうか？　エンジニアたるもの神頼みではなく、己の技術力で不慮の事故に備えなければいけませんが、事故はないに越したことはありません。万全を期したあとの、最後の仕上げとして、ぜひ使ってみてください。

2013年1月号について、たくさんのお便りありがとうございました！

第1特集　いざというときに備えるシステムバックアップ

データのバックアップの基礎から、クラウド環境でのバックアップやFlashストレージ導入のポイントなど、今どきのデータ保護ノウハウを取り上げました。

職場はまだテープを使っていますが、クラウドでのバックアップが勉強になりました！　ただネットワーク帯域をどうにかしないとまだ実用はできないかな……。

神奈川県／尾崎さん

大切だとわかっていてもつい疎かになってしまうものですので、取り組み方を見直す良い機会になりました。

千葉県／ひーちゃんパパさん

クリスマスに客先のサーバが吹っ飛んで一部データを戻せずに憂鬱な年越しのなか、特集記事をシミジミと読みました。知識はやはり体験しないと身につかないものです（おかげでバックアップの予算がつきそうな気配）。

東京都／豊島さん

「データ復旧の実際」が普段直接耳にすることがない話で興味深く拝見しました。保存メディアの信頼性の話題でMOが推薦されていたことに驚きました。光で非接触なら、BDはまだ新しい媒体ということで不安があるのはわかるのですが、大丈夫だろうと思っていたDVDもいろいろ不安があることがわかり、参考になりました。

大阪府／出玉のタマさん

復旧業者に取材した「データ復旧の実際」は担当も初めて知ることが多い記事でした。外付けHDDは平置き型のほうが安全、など個人にも活かせる話が多かったのではないでしょうか。

第2特集　2013年に来そうな技術・ビジョンはこれだ!

技術、開発手法、行動指針など、ITエンジニアが取り組むべき課題はいろいろとあります。業界の第一線で活躍している人々は、今、何に注目しているのでしょう。2013年の学習対象や目標の参考にすべく、ズバリ語ってもらいました。

フリーになって10年目が暮れようとしている今、今後の10年を考えるうえでいろいろ参考になりました。

大阪府／うたさん

2013年も頑張ろう、とモチベーションが上がった。

神奈川県／nagaさん

今後、勉強していくべき方向の参考になった。

神奈川県／yamamotoさん

第5章の「SIerは生き残るか?」は、いつも仕事をしていて感じているSIerの将来の不安について、ほかの人も同じようなことを感じているとわかり、少し安心しました。ただ、不安の解決までは至らなかったので、まずは自分にできること、そう、SDを読み続けることを続けることにします。

千葉県／今井さん

少しでも多くの読者の参考になるよう幅広い分野の方々に執筆してもらいました。今年の目標が見つかったなら幸いです。

一般記事　クロス開発環境「Qt Quick」入門

Qtから派生した「Qt Quick」というアプリケーション開発フレームワークについて紹介しました。

Qt Quickは初めて知りましたが、実際のコードの例が載っていたのがよかったです。昔、似たようなクロスプラットフォームツールとして「wxWindows」というライブラリがあったことを懐かしく思い出しました。

東京都／n0tsさん

内容やおもしろさが伝わってくる非常に良い内容だったと思います。個人的にもQtに興味を持ちました。

東京都／ひろぽんさん

Qtについて、まとまっている記事はめずらしい。

兵庫県／hiroさん

Qt Quickを初めて知った読者も多かったようです。実際のコードや画面を多用して説明しましたが、ツールや開発方法のイメージがわかったでしょうか?

一般記事　インターネットを陰で見守るAkamai

Akamaiは知る人ぞ知るCDNサービスを提供する企業です。今回はインターネットセキュリティの観点から、Akamaiがどんな役割を果たしているのかを取り上げました。

Akamaiは名前は知っていましたが、何のサービスをしているのかよくわからない企業でした。同じような企業の記事を今後も楽しみにしています。

東京都／blackbirdさん

非常に多くのサーバ群をどうやって安全に、かつ可用性を高く維持しながら運用していたのだろうと疑問に思っていました。具体的に「どういった製品群」「どういったスクリプト」などはないものの、具体的なイメージがつかめただけでもためになりました。

東京都／ぱちんさん

Akamaiでは実際にどんな運用をしているのか、もっと突っ込んだ記事を望む声も多くありました。今後の企画の参考にさせていただきます。

特別企画　法輪寺電電宮 情報安全護符シール

企画としておもしろいと思う。情報機器に貼るシールを自作したい気分になった。

神奈川県／野崎さん

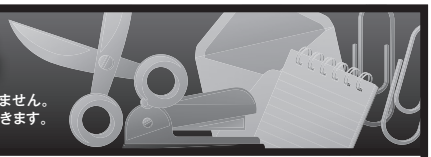
雑誌らしいおもしろい試みと感じました。自社のサーバに貼りました。

東京都／kossyさん

ITエンジニアなら、京都に行った際はぜひ本場の電電宮にも足を運んでみてください。

エンジニアの能率を高める一品

仕事の能率を高める道具は、ソフトウェアやスマートフォンのようなデジタルなものだけではありません。
このコーナーではエンジニアの能率アップ心をくすぐるアナログな文具、雑貨、小物を紹介していきます。



FILCO KeyPuller

480円（税込）／ダイヤテック　http://www.diatec.co.jp

今回紹介するのはキーボードのキーを引き抜く専用工具です。キーの隙間に溜まっているホコリをきれいに取り除くにはキーを外して掃除するしかありません。針金などで取り外せないこともないですが、全部のキーをやるのはたいへんです。このKeyPullerなら写真のようにキーの角に引っかけて気持ちよく引き抜けます。Enterキーやスペースキーは1ヵ所の角にしか引っかけられないので、力を込めないといけませんが、そのさじ加減が難しいです。力任せに引き抜くとキーが跳んでいってしまうので気をつけてください。それにしても、キーを外したキーボードはすごく汚いです。ホコリ以外にも粉のようなゴミがたくさん詰まっていました。キーボードを逆さまにしてゴミを落とし、抜いたキーを1つずつ拭いて、再びキーボードに付け直すと、ピカピカで気持ちがいいです。まるで耳掃除のあとのようにすがすがしいです。（読者プレゼントあります。16ページ参照）

祝　1月号のプレゼント当選者は、次の皆さまです

①ScanSnap iX500 Deluxe ..... 東京都　太田タケシ様
②弥生会計 13 スタンダード ..... 愛知県　鳥居良行様

★その他のプレゼントの当選者の発表は、発送をもって代えさせていただきます。ご了承ください。

次号予告

Software Design
April 2013

2013年4月号
定価1,280円　176ページ

3月18日発売

［第1特集］僕（私）の言語の学び方

裏口からのプログラミング入門

当社好評書籍『Webサービスのつくり方』の和田裕介さんをはじめとして、美容師からプログラマ、レコード店員からスマートフォン開発、書店店員からiPhoneアプリ開発起業、文系新卒からR＆D、などなど異業種からソフトウェアの世界へ進んだエンジニアの言語の学び方・考え方を大公開!

［第2特集］アジャイル再入門・エンゲルバーグ先生再降臨

ソフトウェア開発に効く Small Objectをご存じですか?

［一般記事］ITエンジニアが果たす役目とは?

スクウェア・エニックス「ゲーム開発の舞台裏」

お詫びと訂正

以下の記事に誤りがございました。読者のみなさま、および関係者の方々にご迷惑をおかけしたことをお詫び申し上げます。

■2013年2月号

●目次#02　第2特集 タイトル
［誤］：超効率勉強法
［正］：超効率的勉強法

●目次#03　Software Designer［45］タイトル
［誤］：エンジニアはいかに学び続けるべきか
［正］：生涯学び続けることが仕事の一部

SD Staff Room

●子供が生まれて思うのは、こんなにも無力で手がかかる生き物はいないということ。世の中に良い人間も悪い人間もたくさんいるけど、みな等しく（範馬勇次郎以外w）そうしたかよわい存在だったわけで、心の寛容性のタンク容量が少なくなってきたときは、それを思い出すようにしている昨今。合掌（本）

●インドメタシンの湿布を自分の席で両肩に上手に貼るのにも最近慣れてきた。ゼノールエクサムSXというゲルも準備しているが効き目は湿布の方が上。集中力を増すために額に貼る熱さまシートも完備。仕事用メガネも3タイプ用意。ノートも含めて机上に8画面。地震用警報作動中w　さて、もうひと頑張り。（ま）

●ヨーヨーに次ぐ我が家のブーム。それは「百人一首」！　しかも競技かるた（もどきだけどね）。『ちはやふる』オモシレー。息子などは自分で札を読んでICレコーダーに録音するほど。当然子どもと対戦するわけですが、いまはいい勝負。でも覚えの早い息子にはたちまち太刀打ちできなくなる予感……（キ）

●ただいま、「Software Design総集編」の制作を進めています。3月末に2001～2012年の記事をDVD-ROMに収録して出します。その後には、1990～2000年分を出すことも計画中です。楽しみに待っていてください。ああ、告知してしまった。もう、後には引けない……。（よし）

●今月号で最終回を迎える連載を担当させていただいてきました。毎月できたての原稿を読みながら、このワクワク感を付加価値としてどう読者にお届けするかと考えながら過ごした年月。ほやほやの原稿はやはり湯気が立ち込めていてエキサイティングです。連載の書籍化もありますのでどうぞお楽しみに!　（ほ）

●3月に健康診断がある。年末に体重計に乗って愕然。去年の5キロ増。慌てて納豆ダイエットにコアリズム。そう効果は出ない。もがき苦しんでいたら、年末に肺炎になり2日食事できず2キロ落ちた!　神様～ありがとう!　あと3キロだ!　次はインフルでもノロでもいい。時間がない。（肉川）

●今年の冬はスノボにはまっていて、12月中旬からほぼ毎週末いろいろな雪山に滞在している私。都内でも大雪が降ったが、そのおかげで雪が積もった道でも滑らずに普通に歩けたよ、という話をしたら、それは関係ないでしょうと話す人話す人みんなに否定され中。でも慣れってあると思うんだけどな～。（雪女）

ご案内
編集部へのニュースリリース、各種案内などがございましたら、下記宛までお送りくださいますようお願いいたします。

Software Design 編集部
ニュース担当係

［FAX］
03-3513-6173
※FAX番号は変更される可能性もありますので、ご確認のうえご利用ください。

［E-mail］
sd@gihyo.co.jp

Software Design
2013年3月号

発行日
2013年3月18日

●発行人
片岡　巌

●編集人
池本公平

●編集
金田富士男
菊池　猛
吉岡高弘
＊
細谷謙吾（書籍編集長）
取口敏憲

●編集アシスタント
松本涼子

●編集協力
坂井直美
金子卓也(トップスタジオ)

●広告
中島亮太
北川香織

●発行所
㈱技術評論社
編集部
TEL：03-3513-6170
販売促進部
TEL：03-3513-6150
広告企画部
TEL：03-3513-6165

●印刷
図書印刷㈱

この個所は、雑誌発行時には広告が掲載されていました。編集の都合上、
総集編では収録致しません。

この個所は、雑誌発行時には広告が掲載されていました。編集の都合上、
総集編では収録致しません。