Special Feature 01
MVCの正しい理解
Special Feature 02
ロードバランサのしくみ
2014 April
04
[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
Software Design
2014年4月18日発行
毎月1回18日発行
通巻348号
（発刊282号）
1985年7月1日
第三種郵便物認可
ISSN 0916-6297
定価
本体1,219円+税

Special Feature 01

本質を押さえて見通しのよいシステム作り

なぜMVCモデルは誤解されるのか？

（Java）
（JavaScript）
（PHP）
言語別で考える

Special Feature 02

ネットワークエンジニア養成
ロードバランサの教科書

新連載

シェルスクリプトではじめる
AWS入門

特別企画

今すぐ知りたいSIMのしくみ

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。

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64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。

WebRTC

ブラウザ間のリアルタイム通信を実現する「WebRTC」

「WebRTC（Web Real-Time Communications）」は、Webブラウザ間でビデオチャットやボイスチャット、ファイル共有などのリアルタイムコミュニケーションを実現するためのしくみです。これらのコミュニケーションを実現しようとした場合、従来であれば専用のアプリケーションやWebブラウザのプラグインを双方の環境にインストールする必要がありました。一方WebRTCを用いた場合、Webの標準技術としてWebブラウザ本体に実装されることを想定して作られているため、ユーザは通常のWebサービスを利用するのと同様にWebブラウザのみでリアルタイムコミュニケーションを楽しむことができます。

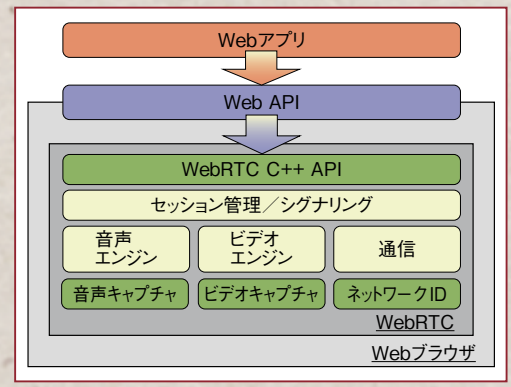
WebRTCの特徴は、Webブラウザ自身がほかのWebブラウザと直接通信するためのしくみを持ち、Webサーバを介することなくデータの送受信やストリーム通信を行えるという点です（ただし、最初に通信を確立するまでのシグナリングやNAT越えを行うためのサーバは別途必要となる）。1対1の通信だけでなく、1対多や、多対多の通信もサポートされており、幅広い用途に利用することができます。

WebRTCのアーキテクチャ

WebアプリケーションにWebRTCによるコミュニケーション機能を組み込むためには、JavaScript向けに用意されたWeb APIを利用します。Web APIではおもに次のような機能が提供されます。

- 端末のカメラやマイクからストリームデータを取得する
- UDP/IPを使用したマルチメディアセッションを確立する
- テキストデータやバイナリデータをP2Pで通信する

▼図　WebRTCのアーキテクチャ



WebRTCの実装には音声エンジンやビデオエンジンも含まれます。音声コーデックとしてはiSACやiLBC、Opusが、ビデオコーデックとしてはVP8が採用されています。またWebブラウザ同士でのP2P通信を実現するために、WebRTCではNAT／ファイアウォール越えのしくみとしてSTUNやICEといったメカニズムをサポートしています。したがって、Webアプリケーションの開発者がこれらの通信メカニズムの実装に頭を悩ます必要はありません。

WebRTCの現状

WebRTCは、もともとはGoogleによって提唱されたもので、現在はIETFによってプロトコルレベルの標準化が、W3CによってAPIレベルの標準化が行われています。標準化と並行してWebブラウザへの実装も進められ、ChromeやFirefox、OperaなどのWebブラウザがWebRTCをサポートしています。それに伴って、実際にWebRTCを組み込んだWebアプリケーションも登場しはじめました。

開発者向けの環境も整いつつあり、オープンソースのWebRTCライブラリが充実してきたほか、自前のアプリケーションにWebRTCを簡単に組み込めるようにするクラウドサービスなどもスタートしています。依然として標準仕様の策定が続けられている段階なので不確定な要素もありますが、Webの可能性を拡げる要素の1つとして注目しておくべき技術と言えるでしょう。SD

WebRTC
http://www.webrtc.org/

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。

Contents

Software Design Apr. 2014

1

第1特集 Special Feature 01

<Java/JavaScript/PHP>言語別で考える

なぜMVCモデルは誤解されるのか？

本質を押さえて見通しのよいシステム作り

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

Contents

Software Design Apr. 2014

2

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/

技術評論社の本が電子版で読める！

電子版の最新リストは

Gihyo Digital Publishingの

サイトにて確認できます。

http://gihyo.jp/dp

GIHYO Digital Publishing 未来の"普通"を、今。

Google play

amazon.co.jp

※販売書店は今後も増える予定です。

法人などまとめてのご購入については
別途お問い合わせください。

■お問い合わせ
〒162-0846
新宿区市谷左内町21-13
株式会社技術評論社　クロスメディア事業部
TEL：03-3513-6180
メール：gdp@gihyo.co.jp

ソフトウェア・デザイン［目次］

Contents

Software Design Apr. 2014






Column

Development

OS/Network

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) MASAFUMI KIMURA/a.collectionRF /amanaimages
Illustration イラスト ＞ フクモトミホ、高野 涼香、中川 悠京
Page Design 本文デザイン ＞ 岩井 栄子、 ごぼうデザイン事務所、 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。

Pocket Reference

小さくても、中身充実！

「あれ何だったっけ？」
「こんなことできないかな？」
というときに、すぐに調べられる
機能引きリファレンス。
軽くてハンディなボディに
密度の濃い内容がギュッと凝縮！
関連項目への参照ページもあって、
検索性もバツグン！

しげむらこうじ　著
四六判／512ページ
定価（本体3,200円+税）
ISBN978-4-7741-6335-2

若杉司　著
四六判／400ページ
定価（本体2,980円+税）
ISBN978-4-7741-6332-1

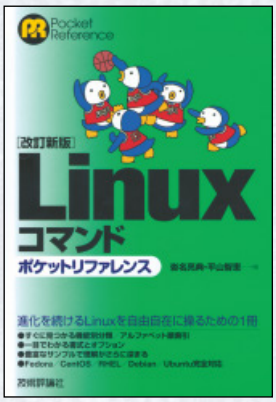
高江賢　著　山田祥寛　監修
四六判／536ページ
定価（本体2,580円+税）
ISBN978-4-7741-4592-1

山田祥寛　著
四六判／528ページ
定価（本体2,780円+税）
ISBN978-4-7741-4980-6

古籏一浩　著
四六判／592ページ
定価（本体2,380円+税）
ISBN978-4-7741-4819-9

田口貴久ほか　著　日本仮想化技術株式会社　監修
四六判／352ページ
定価（本体2,980円+税）
ISBN978-4-7741-5600-2

石田つばさ　著
四六判／448ページ
定価（本体2,180円+税）
ISBN978-4-7741-4836-6



沓名亮典、平山智恵　著
四六判／576ページ
定価（本体2,280円+税）
ISBN978-4-7741-3816-9

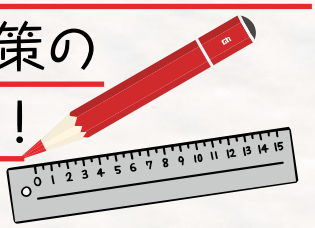
山森丈範　著
四六判／304ページ
定価（本体2,380円+税）
ISBN978-4-7741-4396-5

岡本隆史、武田健太郎、相良幸範　著
四六判／272ページ
定価（本体2,480円+税）
ISBN978-4-7741-5184-7

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

あなたを
合格へと導く
一冊があります！

効率よく学習できる
試験対策の
大定番！



山平耕作　著
A5判／720ページ
定価（本体3,300円+税）
ISBN978-4-7741-6089-4

エディフィストラーニング株式会社　著
B5判／496ページ
定価（本体3,200円+税）
ISBN978-4-7741-6006-1

岡嶋裕史　著
A5判／656ページ
定価（本体2,880円+税）
ISBN978-4-7741-6099-3

濱本常義ほか　著
A5判／592ページ
定価（本体3,200円+税）
ISBN978-4-7741-6178-5

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（本体2,880円+税）
ISBN978-4-7741-6100-6

内田保男ほか　著
A5判／512ページ
定価（本体3,200円+税）
ISBN978-4-7741-6101-3

大滝みや子、岡嶋裕史　著
A5判／736ページ
定価（本体2,980円+税）
ISBN978-4-7741-6111-2

大滝みや子　著
A5判／448ページ
定価（本体2,280円+税）
ISBN978-4-7741-6112-9

大滝みや子　著
A5判／608ページ
定価（本体2,980円+税）
ISBN978-4-7741-5940-9

加藤昭、芦屋広太ほか　著
B5判／464ページ
定価（本体1,680円+税）
ISBN978-4-7741-6110-5

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

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。

データベースエンジニア養成読本
データベースエンジニア養成読本編集部 編
定価 1,980円+税　ISBN 978-4-7741-5806-8

小銅弾のコードなエッセイ
小飼 弾 著
定価 2,080円+税　ISBN 978-4-7741-5664-4

JavaScriptライブラリ実践活用
WINGSプロジェクト 著
定価 2,580円+税　ISBN 978-4-7741-5611-8

〈改訂〉Trac入門
菅野 裕、今田 忠博、近藤 正裕、杉本 琢磨 著
定価 3,200円+税　ISBN 978-4-7741-5567-8

サウンドプログラミング入門
青木 直史 著
定価 2,980円+税　ISBN 978-4-7741-5522-7

OpenFlow実践入門
高宮 安仁、鈴木 一哉 著
定価 3,200円+税　ISBN 978-4-7741-5465-7

はじめてのOSコードリーディング
青柳 隆宏 著
定価 3,200円+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
定価 2,980円+税　ISBN 978-4-7741-5438-1

Webサービスのつくり方
和田 裕介 著
定価 2,180円+税　ISBN 978-4-7741-5407-7

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
定価 2,580円+税　ISBN 978-4-7741-5325-4

Androidアプリケーション開発教科書
三苫 健太 著
定価 3,200円+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
定価 3,400円+税　ISBN 978-4-7741-5143-4

サーバ/インフラエンジニア養成読本 管理・監視編
Software Design編集部 編
定価 1,980円+税　ISBN 978-4-7741-5037-6

サーバ/インフラエンジニア養成読本 仮想化活用編
Software Design編集部 編
定価 1,980円+税　ISBN 978-4-7741-5038-3

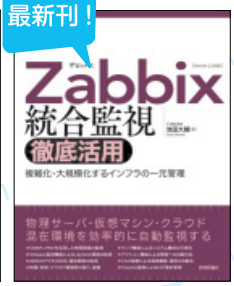
業務に役立つPerl
木本 裕紀 著
定価 2,780円+税　ISBN 978-4-7741-5025-3

Apache[実践]運用/管理
鶴長 鎮一 著
定価 2,980円+税　ISBN 978-4-7741-5036-9

最新刊！

乾 正知 著
B5変形判・352ページ
定価 3,200円(本体)+税
ISBN 978-4-7741-6304-8

最新刊！

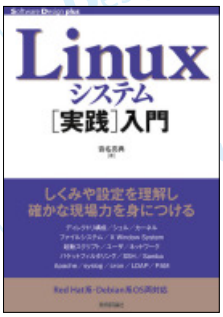


TIS㈱ 池田 大輔 著
B5変形判・384ページ
定価 3,500円(本体)+税
ISBN 978-4-7741-6288-1

㈱バイプドビッツ 著
A5判・224ページ
定価 2,480円(本体)+税
ISBN 978-4-7741-6205-8

久保田 光則、アシアル㈱ 著
A5判・384ページ
定価 2,880円(本体)+税
ISBN 978-4-7741-6211-9

ニコラ・モドリック、安部 重成 著
A5判・336ページ
定価 2,780円(本体)+税
ISBN 978-4-7741-5991-1



杳名 亮典 著
A5判・416ページ
定価 2,880円(本体)+税
ISBN 978-4-7741-5813-6

大谷 純、阿部 慎一朗、大須賀 稔、北野 太郎、鈴木 教嗣、平賀 一昭 著
㈱リクルートテクノロジーズ、㈱ロンウイット 監修
B5変形判・352ページ
定価 3,600円(本体)+税
ISBN 978-4-7741-6163-1

沼田 哲史 著
B5変形判・360ページ
定価 3,200円(本体)+税
ISBN 978-4-7741-6076-4

中井 悦司 著
B5変形判・384ページ
定価 2,980円(本体)+税
ISBN 978-4-7741-5937-9

Japanese Raspberry Pi Users Group 著
B5変形判・256ページ
定価 2,380円(本体)+税
ISBN 978-4-7741-5855-6

菊田 剛 著
B5判・288ページ
定価 2,480円(本体)+税
ISBN 978-4-7741-6128-0

水野 操、平本 知樹、神田 沙織、野村 毅 著
B5判・128ページ
定価 2,480円(本体)+税
ISBN 978-4-7741-5973-7

データサイエンティスト養成読本編集部 編
B5判・152ページ
定価 1,980円(本体)+税
ISBN 978-4-7741-5896-9

Software Design編集部 編
B5判・176ページ
定価 1,880円(本体)+税
ISBN 978-4-7741-5888-4

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

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

6% 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)

14,880円(税込み、送料無料)1冊あたり1,240円(6%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支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)

ネットワークエンジニア虎の穴

『自宅ラックのススメ』

文／tomocha（http://tomocha.net/diary/）

第11回 あなたの可能性を広げる自宅ラック――最終回

イラスト：高野涼香

自宅ラックのよろこび

今回で本連載は最終回。今回まで連載でお話させていただいた内容を振り返ってみましょう。

――自宅にラックを設置するよろこび、ラックを設置するにあたりサーバラックの基礎知識、耐荷重のお話、電気・電源・UPS選び、ネットワーク機器選び、監視の必要性、サーバ選定、ケーブリング、環境構築などとやってきました。このあたりの知識をベースに、皆さんは自分がやりたい環境を構築していると思います。ラックを構築するにあたり、両親との調整や、パートナーがいる場合はパートナーの決済が必要になるケースや、破綻したケースも少なからずあるでしょう（笑）。その中でなんとか調整しつつ、お金の面でもクリアできてやっと自宅ラックが導入できていると思います。それぐらいの意気込みがないと自宅へのラック導入はおそらく難しいと思います。独身はフリーでいいですよ（寒）。

自宅ラックはインフラエンジニアを育てる?

さて、自宅ラックを実現できたら、皆さんその環境で何をやっているでしょうか。筆者は、これまでの連載でも紹介しましたが、自分のサイトの運営（DNS、MAIL、Web）、VPN環境の構築、ファイルサーバの設置をして、データのバックアップも複数拠点で行うといったことをやっています。

まずは、一通り環境がそろったら、自宅サーバで自分のサイトやblogなどを立ち上げてみてもよいでしょう。単なるblogを持つだけなら、レンタルサーバやVPSを借りれば安上がりですが、自宅ラックならば、サーバ、ネットワーク（回線含）、OS、ミドルウェア……とすべて自前で用意しなければなりません。

また、常時運用し続けるということは、脆弱性の情報収集、セキュリティパッチの適用など必要に応じてメンテナンスが必要になりますし、ハードウェアの故障など発生した場合は、交換や修理なども必要になるでしょう。運用し続けるということはいろいろなノウハウもたまります。まあ、自宅ラック運用のための技術というよりは、インフラエンジニアとしての知識・ノウハウに近いですね。

写真1は、筆者の外向けサービスに使っているラック一式です。筆者の場合、このラックの細かな諸情報をwikiなどで管理しつつ、必要な情報はまとめるなどをいつも行っています。サーバはちゃんと動きだすと、なかなか壊れたりしないものですし、定常運用されることから、物理構成が原因になるような障害などは年に1回あるかないか程度しか出てきません。ラック内部の機器類の構成は、案外すぐに忘れてしまいますから、機材のシリアルナンバーや物理構成、ディスクのスロット番号とデバイス名の関連づけ、OSやインストールしたパッケージやミドルウェアのバージョンなどの必要なモノは控えておきましょう。いざというときには迅速に対応できるようにしておきましょう。

検証環境になったりします!

筆者のケースでは、仕事で必要になった際には、ネットワークやサーバなど環境の試験や動作テストなども行うこともありました。会社によっては、必要な機材をちゃんと買ってくれないものです。検証でベンダーから借りることができれば良いのですが、すべてがそういうわけではありません。ちょこっと調べたいときなどに自宅ラックは重宝します。一種のラボ環境ですね。現在はベンダー資格を取るのも視野に入れています。ベンダー資格を取ろうとした際、実機で学ぶことも増えてくるかと思います。

▼写真1　自宅ラック環境(ちなみに、一部の機材の整理・集約した作業中の写真で、地震発生時に転倒しないようにするための固定(ベルト)は写っていませんが対策はしています)

勉強会に行こう／勉強会を主催しよう!

自宅に環境を作っていると、いろいろな人と情報交換がしたくなると思います。そういうときは、ネットワーク系やインフラ系のメーリングリストや、勉強会もたくさんありますので、そういうところに参加してみましょう。技術者同士、意見交換や情報収集をやってみると、さらに理解が深まってよいでしょう。

どちらかというと自宅ラックというよりは、WANの話がメインになりますが、JANOGやMPLS JAPAN、IRS(Inter-Domain Routing Security)、といったところから、インフラ系として、qpstudy、hbstudyなどもありますね。そのほか、OSC(オープンソースカンファレンス)なども面白いかもしれません。

おわりに

ネットワークやサーバなど、興味があっても仕事や会社でなかなか触れない人が多いと思います。本稿をきっかけにして、ラックやサーバマシン、ルータ、スイッチングハブなどを自宅でも触ってみようという気持ちになってもらえたらいいですね。そうでない読者の皆さんも、スケールは違うものの案外似たり寄ったりした環境で、ネットワークは現実には動いていますので、「実際はこうやって動いているんだ」という概観を持っていただければ、幸いに思います。

最後に、本連載に1年間おつきあいいただきありがとうございました。SD

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

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

Volume 184 >>>>>

Interaction Award 2014に見る、新しいガジェットの潮流

インタラクションの世界的アワード

インタラクションアワード（Interaction Award）は、米国IxDA（Interaction Design Association）が主催するインタラクション技術を活用した作品や企画、製品に対する賞です。2012年から始まった新しいもので、今年は昨年以上に世界各国から作品の応募がありました。

審査はまず、応募作品の中から優秀なものが各部門ごとに20作品ほど、shortlist（受賞候補）として公開されます。そののち、shortlistに対する一般投票の結果を加味したうえで、世界各国から集められた実績のある6名の審査員による審査が行われます。

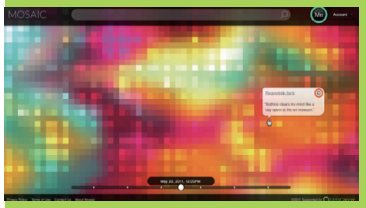
Interaction Award 2014 公式サイト
http://www.ixda.org/awards

shortlist一覧
http://awards.ixda.org/blog/2013/the-2014-interaction-awards-shortlist/

インタラクションアワードでは、その名のとおり人とモノの相互関係、動作や操作に視点が置かれています。人の行動や行為をどれだけ誘発することができ、新しい体験を提供できたかどうかが受賞の観点になっています。

数多くの作品の中から最終候補に残ったのは、世界17ヵ国から応募された75のプロジェクトです。その中から珍しいもの、興味深いものをいくつかピックアップして紹介します。各賞は次の6つのカテゴリに分けられ、複数部門で受賞した作品もあります。

- **Optimizing**＝最適化…日々の行動をより効率的に行う方法
- **Engaging**＝魅了…日々の出来事に喜びや注目を集め、その事柄に意味を与える
- **Empowering**＝助力…人々が限界を超え、今までできなかったような事柄をできるように仕向ける
- **Expressing**＝表現…人々の自己表現や創造性を手助けするしくみ
- **Connecting**＝つながり…人と人



「MetaCancer's Mosaic」。癌患者をサポートするためのシステム

「Gallery One」。美術館の所蔵品一覧表示し、検索するためのシステム


巨大掘削機の操作パネル「RCS 5 Boomer」


「Addicted Products」。ペットロボット風のネットワーク接続された家電機器プロジェクト


「Swegon IQ-Navigator」。複雑な換気システム操作のためのユーザインターフェース

「Dr. Wagon」。木製のブロックの組み合わせで、手続き形のプログラムのしくみを学ぶ子供向けオモチャ

※本記事で紹介しているものは国内未発表・未発売のものを含んでおります。

や、人とコミュニティ間のコミュニケーションを手助けするしくみ

- **Disrupting**＝再構築…当たり前となっている既存のサービスや事柄を壊して再構築し、新しい価値を生み出す

今年の傾向と作品の評価

過去のインタラクションアワードと比較して、今年はとくに参加国の増加が目立ちました。大企業の製品だけでなく、大学生の個人プロジェクトから、IndiegogoやKickstarterなどクラウドファンディングで資金を集めてプロジェクトが進行中のものまで、さまざまな形態と体制で物作りがなされていることが実感される内容ばかりです。

デジタルデバイスとしては、触れる形のあるものだけではなく、ネット上のサービスデザインとしてよく考えられた逸品的なものも高く評価されています。昨今の流行として、ウェアラブルデバイスのインタラクションを考えたものもいくつかありました。おもしろくて先進的なものばかりではなく、社会的課題を解決するためのアイデア、病気や障害を持った子供たちのことを考えたアイデアなど、課題を見つけ、その課題を的確に解決するというアプローチの作品も見受けられました。

ほかにも興味を引いたのは、もう長年使い続けた——決まりきった操作、インタラクションだと思われていたものに、革新をもたらした種類の受賞作です。そのようなものの1つに、トンネルを掘る巨大掘削機の操作パネル「RCS 5 Boomer」のデザインがあります（**前ページ上段右写真**）。ユーザ数、販売数が少ないため、インタラクションデザインにコストをかけたり、現地でのユーザテストに力を入れることが難しい分野であるので、さらに感銘を受けます。

この種のインタラクションデザインは、見た目が良い、親しみやすい、わかりやすい、などといったものとは少し目的が違います。ごく少数のエキスパートたちを対象に、慣れるにしたがって自身の手や指の延長のように感じられる操作感をもたらすこと、さらに間違いや事故を排除する考えも込められています。

作品リストの中には、Google Glassのようなすでに世の中に存在する先進的なデバイスを活用したサービスや、Adobe Kulerアプリのようにすでに広く使われているものがある一方で、まだコンセプトモデルしかないようなものもあり、1つの土俵で審査、評価するには難しいものも数多くありました。

受賞作が素晴らしいのはもちろんのこと、今回残念ながら受賞しなかったshortlistの中に、まだ広く知られていないながらも輝くアイデアが多数ちりばめられていますので、ぜひひとつひとつチェックしてみてください。

インタラクションの行方、マイクロインタラクションという観点

ユーザ体験を分析するときに、人の行為をとても細かく分割して考えることがあります。ユーザ体験の細部に注目するようになってきているということです。たとえば「カフェでコーヒーを飲む」という行為ひとつにしても、実際は、カフェに入ってから、メニューを見て一杯のコーヒーを注文するまでにさまざまなことを考えながら、いくつもの行動を起こしています。また、コーヒーが自分の席に運ばれてからも、砂糖を入れたり、スプーンでかき回したり、飲み頃の熱さになるまで待ったりと、複数の細かい行為を行っています。

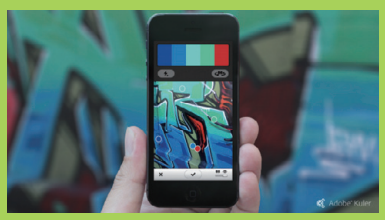
最近では、現実世界の物質とは違い、スマートフォンのタッチパネルで操作するような、現実世界を模倣した仮想空間での行為が増えるにつれて、「マイクロインタラクション」と呼ばれる細部における相互操作が話題にのぼるようになってきました。マイクロインタラクションで着目されるのは、次の4つの状態です。

- **Triggers**＝トリガー…ユーザの行為によって起こる出来事
- **Rules**＝ルール…行為によって何ができるか、何が起こるか定義するもの
- **Feedback**＝フィードバック…操作される側からユーザに対して伝達されるメッセージ
- **Loops & Modes**＝ループとモード…継続的に起こること。操作体系の切り替わり

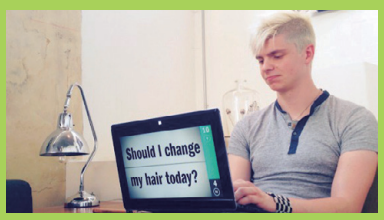
慣れたインタラクションも良いのですが、触って心地よいボタン操作といったお気に入りの操作や、目的によらず操作することそのものが楽しいものもあります。オリジナリティのあるインタラクションがユーザに愛着をわかせ、気に入って使ってもらえるサービスやオモチャになっていくのだと考えられます。

さらに、昨今ではジェスチャーによるデジタルデバイスとのインタラクション

「Walls Have Ears」。壁に耳ありを具現化した、音声認識による言葉の監視カメラ



写真から色パレットを抽出するセンスの良い便利アプリ「Adobe Kuler」



「Personal Billboard」はノートパソコンの背面を個人の広告にしてしまうプロジェクト

もKinectの登場などによって一般化してきました。ジェスチャー関連技術の企業Omekがジェスチャー利用における8つの法則を定義しています。

1. 精確さは良いことですが、でもそれはある程度まで
2. 既存のUIを参考にしないように
3. 肩より高く手を上げなくてはいけないような動作は避ける
4. 操作は素早く、少ない動作でできるよう、小さく短い動作に分割
5. すべてのユーザが右利きというわけではない
6. 国や地域によってジェスチャーには文化的な意味があるので注意
7. ユーザを「効果的なインタラクション領域」に留めておくようにインターフェースを設計する
8. あらゆる大きさの手を使ってテストをする

そして、技術先行型になりがちなハイテク技術の利用に際し、社会的ルールを考えていこうという流れも出てきています。Googleはウェアラブルデバイス Google Glassの使い方として、長時間使い続けない／接触の多いスポーツで使わない／礼儀正しく振る舞う／撮影許可をもらう、といった推奨ルールを提示しています。

街中で、デジタル装置の操作のために変な身振り手振りをしていたら、不思議がられたり、怪しまれたりします。これからはデジタルデバイスだけでなく、それらを使う生活環境も一緒に設計し、デザインしていかなければいけないのかもしれませんね。SD

「Iterazer」はフォルダなどではない、今までにない方法で画像を整理、編集する手法の提案

gadget 1

KonneKt

http://www.jansweijer.nl/projects/konnekt/

病気の子供たち向けのオモチャ

KonneKtは病気のため隔離病棟に入っている子供の患者と、隔離病棟の外にいる家族や友達とコミュニケーションするためのオモチャです。オランダの病院と協力して進められているプロジェクトで、オモチャの部品はレーザーカッターで少量生産され、吸盤や磁石と組み合わされています。ガラスを隔てた隔離病棟側と病棟の外とで、吸盤と磁石でつなぎ合わせながら、コミュニケーションを取ることができます。たとえば、ガラス越しに3目並べの対戦の様子が見られます。

gadget 2

Little People Apptivity Barnyard

http://www.fisher-price.com/en_US/brands/littlepeople/products/78345

タブレット内蔵オモチャ

Apptivity Barnyardはミニチュアの家にタッチパネル式のタブレット端末がはめ込まれたオモチャです。iPadアプリと連携して、植物を育てたり、農場で作物を育てたり、園芸を楽しみながら学ぶことができます。動物（家畜）や道具はタッチパネルに反応するようになっています。対象は1歳半から4歳。子供向けのデジタルオモチャとして受け入れやすいのは、オモチャの中に画面が入っているか、画面の中にオモチャが入っているものが良いと言われているので、王道をいくスタイルです。

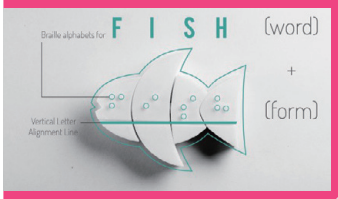
gadget 3

Fittle

http://www.indiegogo.com/projects/fittle-helping-blind-children-learn-through-touch

目の見えない子供向けオモチャ

Fittleは目の不自由な子供たち向けに、物の形と点字とで、物や形を学ぶことのできる学習オモチャです。現在クラウドファンディングのindiegogoで資金を集めているところです。Fittleは学生チームによる企画で、点字の読み書きだけでなく、単語のスペルを学び、点字と現実世界のモノとを密接に関連づけて学習するために作られています。部品同士をつなぎ合わせることで文字の順番を学び、手の触覚でモノの形と点字を学ぶことができます。造形には3Dプリンタが活用されています。



gadget 4

Juice Box

http://www.behance.net/bencollette

発電グッズ

Juice Boxは発電装置（ダイナモ）と蓄電装置が分かれた形のバッテリー機器で、自転車や風車、人力など、さまざまな方法で発電、蓄電できるデバイスです。蓄電装置側にはLEDライトも搭載され、懐中電灯のように使うこともできます。プラグをつないでカーバッテリーを充電したり、USBポートから携帯電話などを充電することが想定されています。まだプロトタイプの段階のようですが、発展途上国の生活で、家庭で1日に必要な電力をまかなうことを考えているそうです。発電のための方法が1種類でなく工夫することができること、電気の使い道としてさまざまに活用できることが特徴です。

結城 浩 YUKI Hiroshi
http://www.hyuki.com/
Twitter：@hyuki

11

結城浩の再発見の発想法

Immutable

Immutable ——イミュータブル

イミュータブルとは

イミュータブル(immutable)というのは、「不変な」という意味の形容詞ですが、技術者以外の方には馴染みの薄い単語だと思います。反対語はミュータブル(mutable)で「変化する」という意味になります。単純にいえば、イミュータブルは「定数のようなもの」ですが、その説明だけではニュアンスは伝わらないかもしれません。

たとえば、プログラムの中で「商品」を表現しているオブジェクトがあるとします。そして、そのオブジェクトは「価格」や「商品名」という情報を内部に持っているとします。

もしもそれが**イミュータブルなオブジェクト**ならば、価格や商品名は変化しません。いったん「100円のチョコ」という商品を表すイミュータブルなオブジェクトxを作ったら、そのオブジェクトxはずっと「100円のチョコ」を表現し続けます。もしも価格を100円から200円に変えたかったら別のオブジェクトを作ることになります。一方、**ミュータブルなオブジェクト**ならば、価格や商品名は変化する可能性があります。価格を200円にしたかったら、価格という内部情報を**変化させる**ことができるのです。

この例でわかるように、**イミュータブルとミュータブルの違いは「内部に持っている情報(状態)が変化するかどうか」にあります。**

メリット

「イミュータブルである」ことは「状態が変化しない」ことですから、**安心してコピーできる**メリットがあります。原本の状態が変化しないので、コピーしたものはいつでも原本と同じ状態であることが保証されるのです。

また、イミュータブルであることはマルチスレッドの環境で大きなメリットがあります。イミュータブルなオブジェクトは状態が変化しませんから、そのオブジェクトを守るための**クリティカルセクションは不要になります。**

さらに、**動作テストがしやすい**というメリットもあります。状態が変化しないので、過去の履歴によって振る舞いが変わらないからです。

デメリット

イミュータブルであることのデメリットは「少ししか違わないのに、**まるごと全部作りなおし**が必要になる」という点です。

ほんのちょっとの違いしかないオブジェクトがほしいのに、ゼロから作るのはたいへんです。とくに巨大なオブジェクトをイミュータブルにするのはコストがかかるものです。

イミュータブル・サーバ

ところが最近、**イミュータブル・サーバ**(Immutable Server)やイミュータブル・インフラストラクチャ(Immutable Infrastructure)と呼ばれる考え方が注目されています。これは、サーバ上にアプリケー

ション(以下、アプリ)をどのように設置するかという方法論です。

サーバに新しいアプリを設置する場合、現在動いているサーバ上に新しいアプリを設置し、そのサーバの設定を変更するのが普通です。これは「サーバという大きなオブジェクトの状態を変化させる」といえます。つまり、サーバをミュータブルなものとして扱っているわけです。いわばミュータブル・サーバです(**図1**)。

それに対してイミュータブル・サーバでは、発想がまったく変わります(**図2**)。イミュータブル・サーバに新たなアプリを設置したいときには、サーバの設定はまったく変えず、アプリが設置されている別のサーバを**まるごと全部作りなおし**、そちらに切り替えるのです。

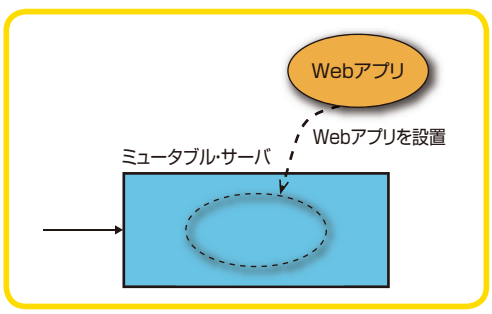
ハードウェアが高速になったことと、仮想マシンの技術が進歩したことによって、イミュータブル・サーバが注目されるようになりました。

イミュータブル・サーバは、状態(設定)が変わらないので、動作テストがしやすくなります。また、新しいサーバがうまく動作しなかったら、すぐに元のサーバに戻すこともできます。必要に応じて同じ設定のサーバをざくざく作り出すこともできます。

複数人の作業とイミュータブル

私たちの日常生活では「永遠に変化しない」ものは少ないですが、「長期間変化しない」ものはたくさんあります。社会のルールや法律などはそうですね。「変化しないこと」は、複数人が一貫した行動をとるために重要なのです。

▼図1　Mutable Server



会社で複数人が1つのプロジェクトにかかわるとき、「不変なことは何か」という知識を共有することは大切です。プロジェクトの納期や方針が毎日変化するようでは落ち着いて作業できません。メンバーが、与えられた情報を心に留めて作業に専念できるのは、「この情報はしばらくは変化しない」と思っているからです。それはリーダーから与えられた情報を、各人が心の中に**安心してコピーできる**という意味です。

変化を作るたびに新規作成

イミュータブル・サーバは「変化させる代わりに新規作成」していました。日常生活でもその発想が役立つことがあります。

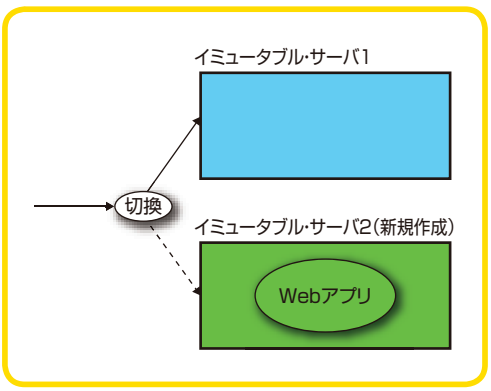
たとえば、1つの部屋で何種類もの作業を行うと混乱が起きることがあります。2つの作業が混ざってしまったり、資材の準備や片付けの手間がかかったりするからです。

イミュータブル・サーバのように「作業ごとに新しい部屋を割り当てる」という方法があります。そのようにすれば、作業を切り替えるときには場所を移動するだけで済むからです。

◆ ◆ ◆

あなたの周りを見回して、変化するものを探してみましょう。その中に不変であることを保証したら便利なものはありませんか。イミュータブルにして毎回作りなおしたほうがいいものはありませんか。ぜひ考えてみてください。SD

▼図2　Immutable Server



enchant ～創造力を刺激する魔法～

(株)ユビキタスエンターテインメント　清水 亮　SHIMIZU Ryo
URL http://www.uei.co.jp

第12回 深圳（シンセン）炎上

ギリギリの決断で個人保証と引き換えに開発資金を確保し、enchantMOONプロジェクトはなんとか開始されました。しかしその向こう側に待ち受けていたのは、製造業という未知への挑戦でした。

アラン・ケイ

資金と開発方針になんとか目処をつけ、プレスリリースを打った僕を待ち受けていたのは、予想を超える反響でした。しかもその大半は否定的なものでした。哲学者と映画監督を招いて新しいハードウェアを作る、としか告知しなかったせいでしょう。しかしこれは、充分予想できた反応でした。

僕は先鋭的なコンセプトであるNO UIを含め、この端末がどのように解釈されるべきか適切な語り部を探し始めました。なぜなら、これを単なるAndroid端末として見られてはコンセプトが生きなくなります。Kindle FireともGalaxy Noteとも違う、まったく新しい世界を作り出そうとしていることを理解してもらわなくてはなりません。

そこでまず最初に仕掛けたのは、映像でした。映画監督の樋口真嗣氏を招き、彼の指導のもと、哲学者の東浩紀のアイデアで短編映画「すばらしい新世界――Brave New World, enchant MOON――[注1]」が作られました。

この短編映画の元になった小説『すばらしい新世界[注2]』は、スティーブ・ジョブズが1984年にMacintoshを発売するときに引用したディストピア小説『1984[注3]』の対をなす作品で、アメリカの国語の教科書に載るほどポピュラーなものだそうです。『1984』が中央集権的なディストピアを描いた作品であるのに対し、『すばらしい新世界』は快楽によって支配された世界で、主人公が「自分で考え、自分で行動し、不幸になる権利」を要求するという作品です。

確かに、いま、App StoreやGoogle Playなどにアクセスすれば、誰でも完成度の高いアプリを信じられないほどの低価格で手に入れることができます。しかし、それによってアプリを作ろうと思うより先に、まず目的に合うアプリを探してみようという発想が出るようになってしまいました。プログラマとして生きてきた僕ですらそうなのですから、世の中のプログラミングと縁のない人々にとってはなおさらでしょう。

enchantMOONが示す新しい世界は、すべての人々が呼吸するようにプログラミングをする世界です。完成されたアプリを使うのではなく、

注1）http://enchantmoon.com/ja/movie/
注2）オルダス・ハクスリー著
注3）ジョージ・オーウェル著

自らアプリを作り出し、またアプリを組み合わせていきます。使いこなすには創意工夫が必要なのです。いわば、不幸になる権利を獲得する端末でもあるのです。

このコンセプトを正しく伝えるためには、まずその相手がアラン・ケイを知っている必要があります。なぜなら、エンドユーザ・プログラミングというパラダイムはアラン・ケイが1960年代に夢見て、そして未だ実現されていない考え方だからです。そこで、まずは哲学者と映画監督を招いて新しいハードウェアを作る、というだけのプレスリリースを打ちました。コンピュータの歴史に精通している人なら、これだけで意図が伝わると考えたのです。また、これだけのヒントで何を作ろうとしているかわからない人には、まだ本当のことを伝えるべきではないと思いました。

最初にそのアクティブソナーに反応してくださったのは、ジャーナリストの西田宗千佳氏でした。Twitter上で取材を申し込まれ、すぐに受けました。次に反応したのは、アスキー総研の遠藤諭氏で、増井俊之氏を招いた対談が緊急に設定されました。そしてその思いは、最終的に思わぬところへ到達します。

なんと、アラン・ケイ氏本人がenchantMOONに興味を示し、直々にコメントをくださるというのです。

つないでくださったのは、Scratchでプログラミングを子供たちに教える活動を続けていらっしゃる阿部和広先生で、先生からCESの会期中に思わぬオファーをいただいた僕は、帰国の予定を伸ばしてアラン・ケイ氏の働く、ビューポイントリサーチ社に向かうことにしました。

アラン・ケイ氏は大変精力的にお話をしてくださり、僕たちのやろうとしていることを応援してくださいました。そのときはまだ製品が完成していたわけではなかったので簡単なデモしかできませんでしたが、アラン・ケイ氏はこんな印象的な言葉をかけてくださいました。

「コンセプトは素晴らしい。けれども、最適化と設計は別の話だ。君たちはまだまだ試すべきことを試していない。もっとリッチなコンピュータともっと高性能な液晶ディスプレイを使って、やりたいと思うことすべてを試してみるべきだ。一度リッチな環境で試してから、実際のプロダクトに落とし込んでいくほうが、プロダクトを最適化しながらリッチにしていくよりずっと有意義な知見が得られるだろうね」

この指摘は非常に的を射ていて、確かに当時の我々にはそうしたことを行う余力がありませんでした。帰り道、僕と辻は、あまりに濃厚な会談を経てぼうっとしていました。僕は興奮が止まりませんでした。なにしろ、アラン・ケイに会ったのです。子供の頃からずっと憧れていたヒーロー、パーソナルコンピュータの父、人生で、彼と会えるチャンスが来るなんて、夢にも思ったことがありませんでした。それがつい先ほど、実現したのです。しかもただ会うだけでなく、我々のプロダクトについてディスカッションしたのです。これ以上興奮する出来事があるでしょうか。

しかし僕たちは、この時点で、産みの苦しみを半分も味わっていなかったのです。それから始まったことは、一言で言えば戦争でした。

予約受付開始

2013年4月23日正午。僕たちはenchantMOONの予約受付を開始しました。価格は39,800円。利益はほとんど出ない設定です。しかし、1人でも多くの仲間を、このプロダクトを買ってもらうことによって見つけたかったのです。もともと、僕たちは1,000台のMOQ(最小生産数)を考えていました。このくらいのロットが中国の工場で作るときにはギリギリです。それでも、小ロットに対応してもらえるからこそ作れるのです。

この瞬間まで、僕は毎日毎晩のように頭を悩ませました。一体、何台売ることができるのだろうか。在庫をさばけなければ、もう次のバッターボックスには立てません。どうすればいい

のか、そればかり考えていました。価格を安くした理由はそれもあります。とにかく利益よりも在庫をだぶつかせないこと。この一点でした。

予約当日の午後1時から記者会見を予定していた僕たちは、東氏と立ち上げたゲンロンカフェを会場としてスタンバイしていました。予約システムは社内のシステム部隊の協力が得られず、しかたなく僕が直前に作ったものでした。UEIの誰もが、「これは売れないだろう」と思っていたことがよくわかるエピソードです。僕はCTOの水野拓宏に、「もし落ちたら、頼むから注文サーバはそっちで作ってくれよ」と釘を刺しておきました。「もし落ちるほど人気があったら、そうしますよ」と水野は笑いました。正直、この時点では僕も水野も、これがどのくらい売れるものなのか想像ができませんでした。

正午、予約開始です。アスキーストアさんとUEI直販、2つのルートで開始しました。

このあと起こった顛末については、ご存じの方もいらっしゃるでしょう。かいつまんで言うと、予想を上回る注文アクセスにどちらの注文システムもダウンしてしまうという失態を演じてしまったのです。僕は大至急注文サーバを作り直すよう水野に依頼し、突貫工事でなんとか受注を再開させました。

最終的に、僕は生産台数を5,000台と決めました。24時間で4,000近い注文が来たので、保守用と店頭用在庫を確保するために、多めに生産することにしたのです。すると製造担当の上瀧英郎が青ざめた顔で僕に言いました。

「そんなに作れるかわからないですよ」

すでに1,000台という見込みで仮発注しているので、材料などはすべて1,000台を基準にそろえられている。それが5,000に増えると材料がそろうまでのタイムラグが大きくなるかもしれない、という話でした。とはいえこの時点で僕は、まだまだ楽観的な気分でいました。そうは言っても、工場なんだから一度に大量に作れるほうが嬉しいだろうと思っていたのです。しかしそれはとんでもなく甘い考えでした。

深圳（シンセン）へ

プログラマたちは疲弊していました。まったく未知のものを作っているのだから、それも当然です。なにしろ何が正解なのか、作ってみないとわからないのです。仕様変更を余儀なくされることもしばしば。開発フロアの雰囲気は最悪で、誰も彼もが見たことがないほど殺気立っていました。それでもなんとか機能が実装され、ようやく光明が見え始めてきたそんなとき、さらに厄介な問題が発生します。

「まだ作ってない?!」

英語で語気を荒らげる上瀧に事情を聞くと、まだ部品が確保できず生産がスタートできていないと。月末には出荷しなければならないのに、それが15日の話でした。僕は心底頭に来ました。先方はこちらの要求など無視して、勝手なスケジュールでものを作ろうとしていたのです。

「電話じゃ話にならない。今から深圳に行く」

僕はそのままオフィスを飛び出し、タクシーで京成上野駅から京急スカイライナーに飛び乗りました。秘書の柿澤からメールで香港行きのEチケットやホテルの手配などが矢継ぎ早に送られてきます。僕はイライラしながらも、中国でどう立ち回るべきか考えていました。中国に詳しい上瀧から注意事項がメールされてきました。

〈清水さん、中国ではたとえ正当な要求であったとしても、怒鳴り合いをしたり暴れたりしたら、共産党や人民軍が架空の罪で拘束し、身代金を要求してくるようなケースがあります。最悪の場合は殺されたり、監禁されたりするリスクもあります。くれぐれも無茶はしないでください〉

そういえば、と僕の脳裏をあるエピソードが過（よぎ）りました。ある米国の会社が、大量のSTB（セットトップボックス）を中国で生産していたところ、工場長が裏切り、クライアントの社員を監禁し、身代金と工賃の上乗せを要求してきたことがあったと。そのとき、その会社は大金を払って中国人民軍を雇い、工場を包囲して工

場長を拘束。そのSTBは無事出荷された、というものです。しかも、ほんの数年前に起きた出来事でした。この事件は秘密にされ、表沙汰にはなっていませんが、そういうことがあったと、まさにその会社で働く旧知の友人が笑い話のように語っていたのを思い出したのです。僕はiPhoneから、その友人に連絡をとりました。

〈これから深圳でひと暴れしてくる。もしかしたら拘束されるかもしれない〉

返事はすぐに来ました。

〈中国？　無茶はするな。拘束ですめばいいが、最悪、命を落とす〉

〈頼みがある。48時間連絡がなかったら、助けてほしい〉

〈わかった。だが期待はするな〉

社長である自分が向かうのはリスクが伴いましたが、社員を危険な目に遭わせるのはもっと嫌でした。家族の顔が頭に浮かびましたが、あえて連絡はしませんでした。それをしてしまえば、本当に僕は自分の身を投げ出してしまいそうだったからです。

契約違反

数時間後、僕は香港のホテルで朝を迎えました。深圳へは陸続きの香港ルートで行くのが安全とされていたので、そこからタクシーで向かいます。1年前にRockchipのブースで出会ったチェリーさんも、この日は通訳として同行してくれていました。彼女がいれば百人力です。

「清水さん、連中、とんでもないことをやらかしてます」

合流した上瀧が持ってきたプリントアウトには、最悪の内容が書かれていました。なんと中国の工場は、我々に無断でコピー商品を自社のWebサイトで販売しようとしていたのです。さらにひどいことに、彼らはすでにヨーロッパのバイヤーとコンタクトしており、コピー商品の出荷の約束まで取り付けていたのです。それでもっと売れるのではないかと考え、勝手に製造のタイミングを引き延ばしていたのでした。

「それ、なんですか？」

「これか？　酒だよ。手土産さ」

「連中に渡すんですか？　我々に対して犯罪行為をしてる相手ですよ」

工場に乗り込んだ僕たちは、担当者のセールスマネージャーの女性を呼び出します。彼女が流暢な英語でヘタクソな言い訳をするので、僕は怒りが頂点に達しました。

「あんたのボスを呼べ。ここに。今すぐ」

すぐに上司と名乗る男が現れました。僕は彼に穏やかな口調で言いました。

「君たちがやっていることは契約に違反しているだけでなく、犯罪だ。今すぐ言うことを聞かなければ、あんたとあんたのボスを、個人的に訴える。いいか。会社じゃないぞ。君たち個人を訴える。何億という額を個人的に賠償させるぞ。一生かかって払わせてやる。覚悟しろよ」

上司は英語がよく理解できていないようでした。セールスマネージャーの女性は僕のあまりにあからさまな恫喝に驚き、口をぱくぱくさせていました。チェリーさんが慎重に翻訳すると、上司は見る見る顔が青ざめていきました。

「おまえは破滅だ」

僕はセールスマネージャーに言いました。彼女はその場で解雇され、CEOが正式に謝罪に来て、ようやく製造ラインを確保してもらえることになったのです。CEOは英語が流暢で、しかも若く、精力的な男に見えました。

「CEO、これは『洗心』といって僕の故郷、東京のずっと北側にある、雪国で作られた酒です。久保田という高級酒の、さらに厳選された一流の日本酒です。我々の製品のこと、よろしく頼みます」

彼はいたく感動し、僕たちを郊外にある高級レストランで歓待してくれました。それから中国滞在中は何度も何度も2人で潰れるまで白酒（パイチュウ）を飲みました。

しかしこれもまだ、戦いの序章に過ぎなかったのです。SD

コレクターが独断で選ぶ!

偏愛キーボード図鑑

株式会社 創夢
濱野 聖人 HAMANO Kiyoto
khiker.mail@gmail.com
Twitter：@khiker

第12回

静電容量無接点方式の高級キーボード

REALFORCE

写真1　REALFORCE 108UG-HiPro

はじめに

今回は東プレ㈱のREALFORCE（**写真1**）を紹介します。REALFORCEは非常に有名な高級キーボードで、さまざまな職種の方に愛用されています。キースイッチに静電容量無接点方式を採用しており、抜群のキータッチを持つことで知られています。ラインナップも豊富にあり、スイッチの荷重の違いなど細かい部分が異なる機種が数多くあります。その中から筆者が所持している3機種を紹介します。

REALFORCE

REALFORCEは東プレ製のキーボードで、次の特徴があります。

- **静電容量無接点方式のキースイッチ**
- **豊富なラインナップ**

静電容量無接点方式とは高い耐久性と心地良いキータッチが特徴のキースイッチです。高い耐久性とは、無接点という名のとおりキー入力の際に電極に直接触れませんので、よくあるメンブレンスイッチと比べて壊れにくいということです。キータッチも良く、スイッチの荷重も30g、45g、55gと3種類があります。また、底までキーを押し下げずとも1mmほど押し下げるだけで押したと認識されるため、訓練しだいでほとんど力を入れずに打鍵できるようになります。

ラインナップも豊富です。大雑把ですが、**表1**にまとめてみました。

キー配列はフルキーボード、テンキーレスキーボードどちらも出ています。日本語配列と英字配列の両方があり、日本語配列ではWinキーのないバージョンもあります。新品の入手は困難なようですが、REALFORCEのテンキーもあります（**写真2**）。

スイッチはすべて静電容量無接点方式ですが、通常のタイプと静音のタイプがあります。静音タイプではより打鍵音が抑えられています。

スイッチの荷重を変えたバージョンも多数存在します。すべてのキーが30gのバージョン、45gのバージョン、55gのバージョンとあります。また、変荷重というバージョンも存在します。変荷重モデルは基本的には45gですが、力の弱い小指で打つキーは30g、Escは55gというようにキーごとに異なる荷重が設定されています。

筐体の色も黒色と白色のバージョンがあります。最近では青色や赤色

表1　REALFORCEのラインナップ

キー配列	日本語配列、英字配列、フルキーボード、テンキーレスキーボード
スイッチの種類	静音タイプ、通常タイプ
スイッチの荷重	変荷重、ALL 30g、ALL 45g、ALL 55g
筐体の色	黒色、白色
接続	PS/2、USB

写真2　REALFORCEテンキー

写真3　REALFORCE 87UB SE170S

写真4　REALFORCE 108UG-HiProのキー

などにカラーリングされた交換用のキートップも販売されています。

接続はPS/2とUSBの2種類がありますが、今はPS/2はほとんどなくなり、USBだけのようです。

REALFORCE 87UB SE170S

変荷重、英字配列、テンキーレスの静音モデルです(**写真3**)。日本語配列のバージョンも存在します。SE170SにはDIPスイッチがついており、これにより左のCtrlとCaps Lockを入れ替えられるようになっています。価格は約25,000円と通常のバージョンよりも多少高いですが、静音のためだけに数千円上乗せして買う価値はあります。打鍵音を気にしなければならないのであれば、この静音バージョンを選ぶと良いでしょう。

REALFORCE 108UG-HiPro

ALL 45g、日本語配列、フルキーボードでキートップに特徴のあるモデルです(**写真1**)。電子式タイプライターをイメージした形状とのことで、キートップのひとつひとつに窪(くぼ)みがあります(**写真4**)。そのため、タイピングをするとキートップが指にはりつくような印象があり、スムーズにタイプできます。普通のキーボードでは、ホームポジションを確認するためにFとJにポッチがありますが、このキーボードではそれが存在せず、代替としてほかのキーよりもさらに窪みが深くなっています。英字配列のバージョンも存在します。テンキーレスのモデルはまだないようです。

REALFORCE 89S-10th

REALFORCEの10周年記念モデルです(**写真5**)。静音、ALL 30g、PS/2、日本語配列でWinキーなしのテンキーレス、青と灰の特徴的な配色と、たいへん尖ったモデルです。もしかしたら、PS/2やWinキーレスというモデルはこの先見られなくなるかもしれません。1,000台限定での生産とのことでしたが、執筆現在(2014年2月)Amazon.co.jpでまだ入手が可能なようです。PS/2やWinキーレスのモデルがほしい方は確保しておくと良いでしょう。ただ、ALL 30gと荷重は相当軽く、軽過ぎるという印象を持つ方も少なくないと思います。その点だけには注意が必要です。

◆ ◆ ◆

筆者が初めて購入した高級キーボードもREALFORCEでした。店先で初めて触ったとき、パソコンに付属していたキーボードとのタッチの違いに感動したことを今でもよく覚えています。REALFORCEは少し大きめの量販店に行けば普通に店頭で販売しており、触って試せる場所も少なくありません。さまざまな種類がありますので、店頭で触ってみて、自身にあったものを選ぶと良いです[注1]。SD

写真5　REALFORCE 10周年記念モデル

注1)　ちなみに筆者のお勧めは、静音でテンキーレスの変荷重のモデル(REALFORCE 91UBK-S)です。

はんだづけカフェなう

ステッパーをはじめよう(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/

おさらい

前回は概要ですが、ステッパーのしくみと動かし方を扱いました。簡単におさらいをすると、ステッピングモーターの中には複数組の電磁石が入っていて、電磁石に電流を流して磁石にする(励磁(れいじ))ことで軸に取り付けられた永久磁石を引っ張るというものでした。ステッピングモーターには、電磁石の両端にだけ電極が付いているバイポーラと、両端に加えて真ん中にも電極のあるユニポーラという種類があることも説明しました(**図1**)。

本連載では、半導体が決まった方向への電流をON/OFFすることが得意であることから、一定方向に電流を流すユニポーラーを扱っています。

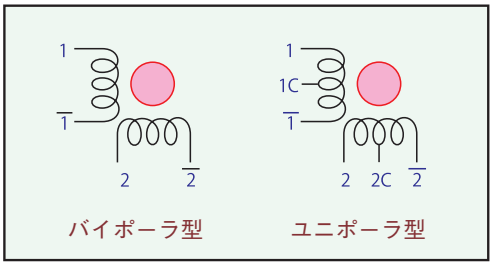
PCA9629A

前回の最後のほうで、ステッパーモーターコントローラなどと呼ばれるステッピングモーターの制御を委ねられるチップを簡単に紹介しました。今回取り上げるPCA9629Aというチップもその1つです(**写真1**)。PCA9629Aは、オランダに本社があるNXPセミコンダクターズという半導体メーカーの製品です。NXPは、もともとノンフライヤーやソニッケアー、そしてカセットテープなどのメーカーであるフィリップスの半導体部門から分社化した会社です。フィリップス社は、本連載でも何度か紹介してきたI²C(アイ・スクエア・シー)(Inter-Integrated Circuit)という2本の信号線を使う通信規格を提唱した会社で、今回のPCA9629AもマイコンとI²Cで通信するように作られています。

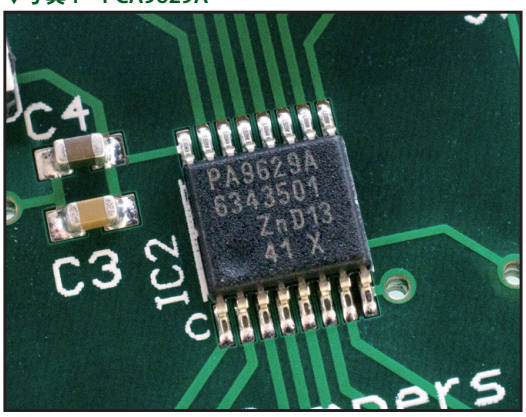
I²Cは、1つのバス(2本の信号線のペア)に複数のデバイスを接続できます。このため、マイコンの2つのピンを使うだけで、複数のデバイス、たとえば複数のPCA9629Aを接続してコントロールできます。前回の例ではステッピングモーターを1つコントロールするために4つのマイコンのピンを使いましたが、I²C接続のステッパーモーターコントローラを使用すれば2つのピンで複数のステッピングモーターをコントロールできます。

1つのバスに接続された複数のデバイスを見

▼図1　ステッピングモーターの回路記号



▼写真1　PCA9629A



分けるため、I²Cにはデバイスアドレスというものがあります。PCA9629Aはアドレスを4bitで設定できますので、16個のPCA9629Aを1本のバスに接続できることになります。

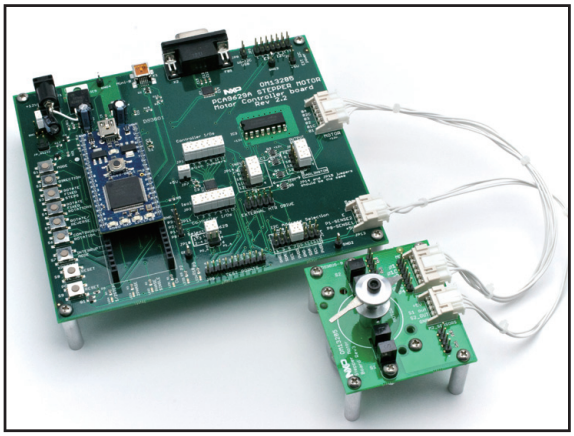
PCA9629Aをマイコンからコントロールするには、PCA9629Aのレジスタ（記憶領域）にコマンドをI²Cで書き込みます。コマンドの解説はデータシートに書かれていますが、サンプルコードも提供されています。NXPはLPCというARMマイコンも作っており、このLPCマイコンとmbed用のサンプルコードがあります。今回はステッパーモータードライバというのがどういうものなのかを手軽に試すため、NXPセミコンダクターズジャパンから評価ボード（**写真2**）を借りて、mbedで動かしてみました。

PCA9629Aの評価ボードであるOM13285には、LPCマイコンの評価ボードも、mbedも取り付け可能なソケットが付いています。ここにmbed LPC1768を取り付けてみます。次に、mbed.orgにあるサンプルコード（**図2**）を自分のmbedオンラインコンパイラにimportし、コンパイルします。コンパイルが完了するとバイナリがダウンロードできますので、これをmbedのドライブにドラッグ＆ドロップしてコピーします。mbedをリセットするとステッピングモーターが回転し始めました。

モーターを回転させる

ステッピングモーターをPCA9629Aを使って動かすためのコードを実際に見てみましょう。先ほどのサンプルコードを見ると**リスト1**のようなコードが書かれています。レジスタの初期化をしたあと、時計回り

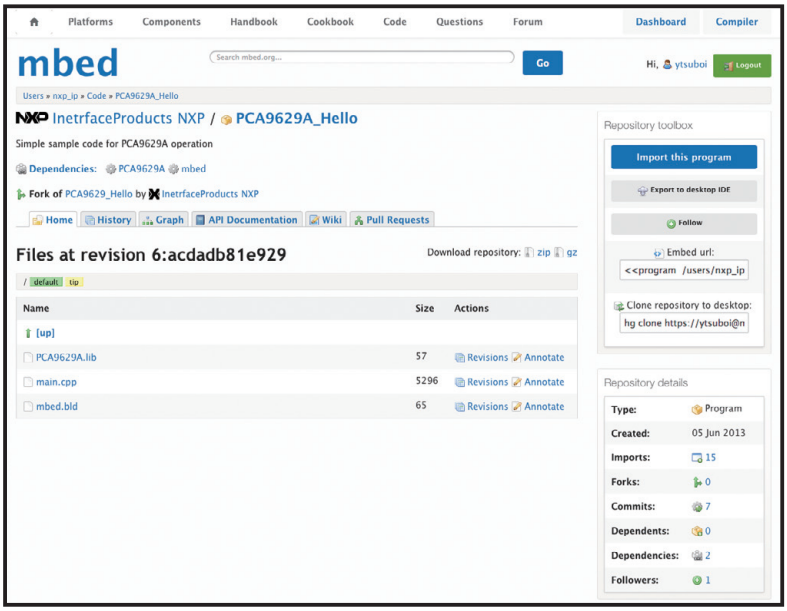
▼写真2　評価ボードOM13285とmbed



▼リスト1　ステッピングモーターを動かすコード

```
motor_controller.init_registers();

motor_controller.pps( PCA9629A::CW, 200 );
motor_controller.steps( PCA9629A::CW, 96 );

motor_controller.pps( PCA9629A::CCW, 100 );
motor_controller.steps( PCA9629A::CCW, 48 );

for ( int i = 0; i < 2; i++ ) {
    motor_controller.start( PCA9629A::CW );
    wait( 2.0 );

    motor_controller.start( PCA9629A::CCW );
    wait( 2.0 );
}
```

▼図2　サンプルコードのあるページ



https://mbed.org/users/nxp_ip/code/PCA9629A_Hello/

(ClockWise)に200pps(パルス/秒)の速度で96ステップ(2回転分)、反時計回り(Counter-ClockWise)に100ppsで48ステップ(1回転分)動かすという宣言をしています。その後は実際にモーターを動かすコマンドをウエイトを挟んで、実行していることがわかります。

前回のようにマイコンで直接ステッピングモーターを制御する方法では、ステッピングモーターを動かすために1ステップずつI/Oを操作して電磁石に電流を流したり止めたりする操作をする必要がありました。PCA9629はインテリジェントなステッパーモーターコントローラですので、こういった操作を簡単なコマンドを発行することで実行できます。

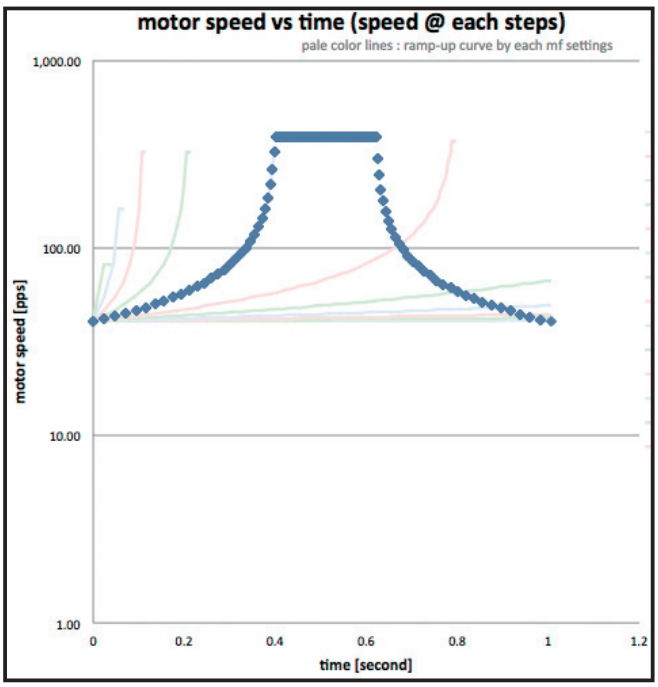
モーターの回転速度を変える

決まった速度でステッピングモーターを回転させるだけであれば、まだ直接制御を行ってもコードはさほど複雑にならないかもしれません。前回の最後に紹介した「台形制御」をしてみましょう。ステッピングモーターを制御するときには、脱調(だっちょう)[注1]が発生しないようにするために緩やかに加減速をさせる台形制御という方法がよく用いられます。

レジスタの設定値が直接コード(**リスト2**)に書かれているため、具体的な設定内容はさておき、これだけの行数で**図3**のようになめらかに加速をしたあとに等速でモーターを動かし、止める前にもなめらかに減速をさせることができます。直接ステッピングモーターを制御すると、コードはこの程度の長さではすまなくなってしまいます。

こういった台形制御を行うためのパラメータは、NXPが提供しているツールに条件を入力するだけでサンプルコードに組み込まれた形で手にできます。データシート(半導体の取扱説明書)を隅から隅まで読んで把握しなくとも、望む条件でモーターを動かすコードが簡単に手に入るようになっています。**図3**のグラフも、このツールでプレビューとして出力されるものです。

▼図3　台形制御



注1) モーターが意図したとおり動かないこと。

▼リスト2　台形制御のコード(レジスタに設定する値を直接設定している)

```
motor_controller.write( PCA9629A::RUCNTL, 0x28  );
motor_controller.write( PCA9629A::RDCNTL, 0x28  );
motor_controller.write( PCA9629A::PMA, 0x01  );
motor_controller.write( PCA9629A::CW__STEP_COUNT, 87  );
motor_controller.write( PCA9629A::CW__STEP_WIDTH, 854 );

motor_controller.write( PCA9629A::MCNTL, 0x80  );  //  start
```

励磁モード

前回は一相励磁という方法のみを説明しましたが、ステッピングモーターにはほかの回し方があります。1つの電磁石でステッピングモーターを回してきましたが、2つの電磁石に同時に通電したらどうなるでしょう。実際にやると、**図4**のようになります。これは二相励磁と呼ばれます。二相励磁は2つの電磁石に同時に電流を流しますので、倍の電流がモーターに流れることになります。引き替えに、2つの電磁石で引っ張りますので、倍近いトルクを得ることができます。

この二相励磁を行ったときには、一相励磁の1でも2でもない中間地点に永久磁石（軸）が位置していることに気づきましたか。一相励磁と二相励磁を併用すると、今までの倍の位置を永久磁石（軸）は取ることができます。この一相励磁と二相励磁を交互に切り替えることを、一-二相励磁、あるいはハーフステップドライブと呼びます。1ステップで1.8°ずつ回転するステッピングモーターを一-二相励磁でドライブすると0.9°ずつ回転させることができます。

PCA9629Aでは、この一相励磁、二相励磁、一-二相励磁の、どの励磁モードも取ることができるようになっています。

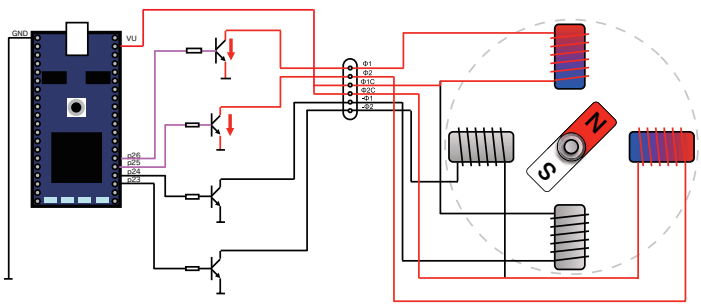
パワー段

ところで、この評価ボードには前回使用したものと同じようなトランジスタアレイが付いています。PCA9629Aもマイコンと同様に、大きな電流を流すことができないためです。こういったトランジスタアレイなどを用いて電流を流すことができる回路はパワー段などと呼ばれます。

今回紹介したPCA9629Aはインテリジェントなステッピングモーター制御を行うために開発されたチップですので、パワー段は外付けするように設計されています。

一方で、このパワー段を主軸に置いたステッピングモータードライバも存在します。たとえばAllegro社のA4988というチップは、トランジスタアレイなどを用意する必要がなく、チップを直接ステッピングモーターに接続して使用したりします。A4988は3Dプリンタのステッピングモーターを制御するために多用されているチップです。後編では、このA4988を使ってステッピングモーターを動かしてみたいと思います。SD

▼図4　二相励磁




```
motor_out0    = 1;
motor_out1    = 1;
motor_out2    = 0;
motor_out3    = 0;
```

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2014年4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 2名 EVOUNI Leather Arc Cover iPad mini Retina L37

職人がひとつひとつ吟味した上質なカウレザーを採用したiPad mini/iPad mini Retina専用ケース。高級感溢れる外見と、実用性を兼ね備えています。革の弾力を利用して、さまざまな角度にiPad miniを立てかけられます。色はラズベリー（ピンク）かハーネス（茶色）の各1名様ずつです。※製品にiPad miniは付属しません

提供元 リンクスインターナショナル **URL** http://www.links.co.jp/

02 1名 SD・microSDカードリーダ・ライタ SCR-SD03/BK

microUSBとUSBの両方のコネクタを搭載したSD・microSDカードリーダ・ライタ。コンパクトな直挿しタイプで持ち運びにも便利。ホスト機能（OTG）対応のAndroidスマホ／タブレット端末と、Windows 8.1/8/7/Vista/XP、Mac OSX 10.6以降のPCで利用可能。

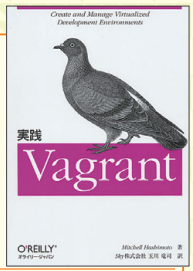
提供元 ミヨシ **URL** http://www.mco.co.jp/

03 5名 ボールペン付きタッチペン スタイラス C1

1本で油性ボールペンとタッチペン（静電容量方式用）が使えます。タッチ部には感度の高いソフト導電性シリコンゴムを採用。スマホなどをストレスなく操作できます。※上記のいずれか1本が当たります

提供元 ゼブラ **URL** http://www.zebra.co.jp/
TEL ゼブラお客様相談室　0120-555335（平日9時～17時）

04 2名 実践 Vagrant

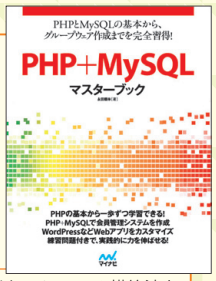


Mitchell Hashimoto 著、Sky㈱ 玉川 竜司 訳／
A5判、248ページ／
ISBN = 978-4-87311-665-5

Vagrantの使い方からプラグインの開発方法までを解説。日本語版独自の内容として、吉羽龍太郎氏による「Vagrantプラグイン」と「Packer」、伊藤直也氏による「Vagrantと Amazon EC2」を収録。

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

05 2名 PHP+MySQL マスターブック



永田 順伸 著／
B5変形判、384ページ／
ISBN = 978-4-8399-4759-0

1冊でPHPとMySQLの基本とWebアプリケーションの構築法を学習できる実践的なプログラミング入門書。実際に活用できる会員管理アプリケーションを作成しながら各種技術を詳しく解説します。

提供元 マイナビ **URL** https://book.mynavi.jp/

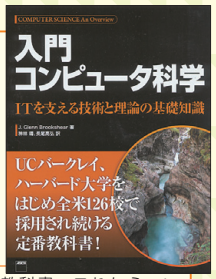
06 2名 絶対に挫折しない iPhoneアプリ開発「超」入門 増補改訂版

高橋 京介 著／
B5変形判、416ページ／
ISBN = 978-4-7973-7545-9

プログラミング経験ゼロから始めるiPhoneアプリ開発の入門書。アプリ開発の基本から、アプリをApp Storeに公開するまでの全手順を丁寧に解説します。増補改訂版で最新のiOS7に完全対応。

提供元 SBクリエイティブ **URL** http://www.sbcr.jp/

07 2名 入門コンピュータ科学



J. Glenn Brookshear 著、神林 靖、長尾 高弘 訳／
B5変形判、608ページ／
ISBN = 978-4-04-886957-7

米国大学の教養課程で使用されている定番教科書。これからコンピュータ科学を学ぶ学生や、大学でコンピュータ科学を学ばなかった社会人プログラマのための独習書として最適です。

提供元 KADOKAWA **URL** http://asciimw.jp/

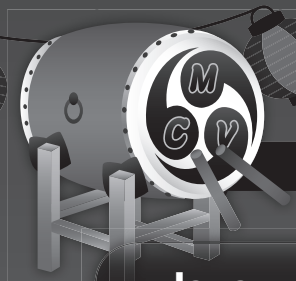
第1特集

Java/JavaScript/PHP言語別で考える

なぜMVCモデルは誤解されるのか？

本質を押さえて見通しのよいシステム作り

Webアプリケーション開発において、MVCモデルを適用して設計をし、実装をすることが多くあります。しかしながら、すべてのエンジニアがMVCモデルの意味を正しく理解しているかというと、あやしくなります。それは使用している言語によって考え方が違っていたり、解釈が違うことがあるからです。MVCの原型はオブジェクト指向言語Smalltalkにあります。そこからさかのぼるため、まずJavaにおけるMVCを取り上げます。そしてWeb開発でもっとも注目されているJavaScriptでのMVC適用例である、Backbone.js、AngularJSの使われ方をみていきます。そして同じくWebプログラミングでもっとも使われているPHPでのMVC利用例を解説します。本特集によってWeb開発で見通しのよいシステムを作る参考にしてください。



CONTENTS

第1章 Java編 ——MVCの原型を知り、JSP/ServletとSpring MVCで再確認!

Writer 長谷川 裕一、土岐 孝平、大野 渉(Starlight & Storm http://www.starlight-storm.com/)

MVCモデルの普及はJavaからでした。Servlet、JSPでのWebアプリケーション開発の案件が増えるに従い、MVCを利用して設計し開発を進める、そんな光景がいろいろな所で見られました。しかし、その原型となったSmalltalkの思想は追いやられながらでした。その結果、形骸的な使用方法が先行してしまい、MVCモデルを実際に使うときに混乱と誤解が生まれてしまったように思います。本稿では原点に戻り、ちゃんとMVCモデルを押さえてから、Javaでの利用方法を確認していきます。

Part1 MVCモデルを正しく理解するために

エンジニアであれば、新人研修やWebアプリケーションを開発するときなど多くの機会にMVCという言葉を耳にしたことが一度はあると思います。

しかし、多くのエンジニアは、自分にとって身近な特定のプログラミング言語(たとえば、Javaであったり、RubyやPHPなど)を通してのみMVCを理解していて(それはそれで重要ですが)、MVCがどのような背景から現れ、どのような経緯を経て現在のカタチに至ったのかは知らないことが多く、そのためにエンジニアとしての幅がいくぶん狭いように思えます。

そこで本稿では、まず最初に現在もっとも多くのエンジニアが理解しているWebアプリケーションのMVCを復習し、そのあと、歴史をさかのぼって初期のMVCを解説します。初期のMVCを理解することで、現在のMVCを今までと違ったとらえ方ができるようになり、エンジニアとしての幅が広がることと思います。

なお、Part1の内容は、これからエンジニアとして第一歩を踏み出す方にも読んでもらえるように比較的平易に書きましたが、もし書かれている内容にピンとこないことがあったら、まずはPart2以降やほかの章で特定のプログラミング言語のMVCを学び、その動作やしくみを理解してから読み直すとわかりやすいと思います。

MVC2の「2」の意味は何ですか?

MVCはModel-View-Controllerの略であり、現在では、Modelは「ビジネスロジックを処理するところ」、Viewは「表示するところ」、Controllerは「ユーザからのイベントを受け付け、処理と処理結果の表示を制御するところ」のように解説されてることが多いと思います。こうしたMVC(**図1**)をJavaで新人研修的に実現すると「Model = Javaアプリケーション」、「View=JSP」、「Controller = Servlet」となって、Webアプリケーションを説明するときによく利用されます。

そのMVCの流れは、①Controllerがイベントを受け取りModelにメッセージを送り、②Modelの状態が更新されます。次に③ControllerによってModelの状態変化がViewに通知され、通知されたViewは、④Modelの最新の値を取得し、⑤表示を行います。実際のWebアプリケーションでは、MVCを実現するためにさまざまなフレームワーク[注1]が利用されていますが、基本的な流れは変わらないでしょう。

このようなWebアプリケーションを開発するために利用されるMVCのことは、「MVC2

注1) 27ページ表1を参照してください。

(MVCモデル2)」や「Web MVC」ということもあります。本章でもこれらはMVC2と記述することにしましょう。なぜMVCではなく、区別するような呼び方をする必要があるのか、次節ではMVCの源流を探って、本来のMVCを明らかにしていきましょう。

MVCの原点とは何ですか?

XEROX PARC研究所で誕生

MVCの源流は1970〜1980年代、世界で最初のパーソナルコンピュータDynaBookを作ろうとしていたXEROX PARC研究所まで遡ります。そこでGUIの実現を議論した結果、「Model = 知識の表象[注2]」「View = Modelの(視覚的な)表象[注3]」「Controller = ユーザとシステムとを紐付けるもの」とするMVC(**図2**)が誕生しました[注4]。

ここで興味深いのは、このMVCはMVC2と異なり、ViewとControllerが密に結合しているという点です。MVC2では、ViewとControllerが疎結合になり、Controllerからのみviewに対して指示を行うという形になっていましたね。

MVC2では、Viewはロジックを持たずに、Modelを参照するだけというのが基本的な枠組みとすることが多いですが、現実問題としてViewには、テーブルを表示するためのループ処理など、HTMLを組み立てるロジックが入ってしまうことになり、基本的な枠組みから外れてしまうことが多いのもたしかです。こうした事態はMVCの視点から極端に評価すると、MVC2ではVとCは本来密結合であって良いものであるのに、それを無理に疎結合にしようとしたために面倒なことになっているということになります。

MVCでの処理の流れ

MVC(**図3**)の基本的な流れは、①Controllerがイベントを受け取り、Modelにメッセージを送り、②Modelの状態が更新される。③**Modelの状態変化はViewに通知され**、④通知されたViewは、Modelの最新の値を取得し、⑤再表示を行う、となります。

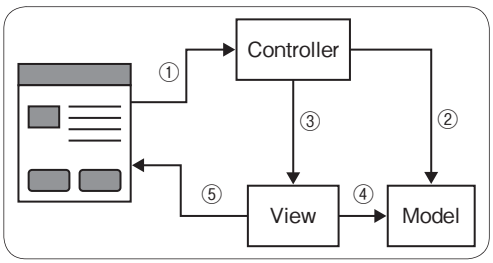
ここでのポイントは、Modelの状態変化はViewに通知されるということです(太字強調部分をMVC2の流れと比較してください)。つまり、本来のMVCではあるModelをユーザが見ていた場合、そのModelが何らかの理由で変更

注2) 昨今、DDD(ドメイン駆動設計)などでDomain Modelはメンタルmodelであるという議論がされていますが、これは目新しいことではなく、過去においてすでにModelは知識の表象=メンタルModelであるということが言われていたのです。

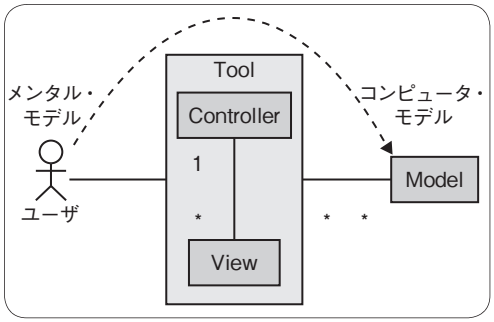
注3) よく勘違いされることですが原理的には、ViewはRDBに保存されたデータモデルをみるためのモノではなく、システムの中という仮想空間(現実的にはメモリ)中に展開されているModelを見るものです。実際にはModelもデータモデルに変換されRDBに保存されることが多いですが、オブジェクト指向原理主義者の視点からいえば、ModelをRDBに保存するのは便宜的なもので、たとえていえばRDBはModelを仮想空間から一時的に避難させるためのスワップ領域であり、本質的にはスワップ領域がRDBだろうが何だろうが、オブジェクト指向とは何の関わりもないし、スワップ領域などは本来ないほうが良いという考え方なのです。

注4) 『MODELS - VIEWS - CONTROLLERS』: 1979/12/10 Trygve Reenskaug http://heim.ifi.uio.no/~trygver/1979/mvc-2/1979-12-MVC.pdf

▼図1 現在のMVC(MVC2)



▼図2 MVC(参考:『MODELS - VIEWS - CONTROLLERS』)

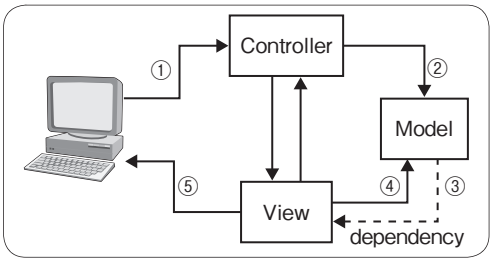


された場合、Viewを通してそのModelを見ているユーザにModelの変更が即時にわかるようになります(MVC2ではModelの変更はViewには通知されません。現状のMVC2で実装されたアプリケーションでは、Modelが変更されたことを知るためには、ブラウザの再読み込みボタンを何度もクリックしないといけないことが多いでしょう)。

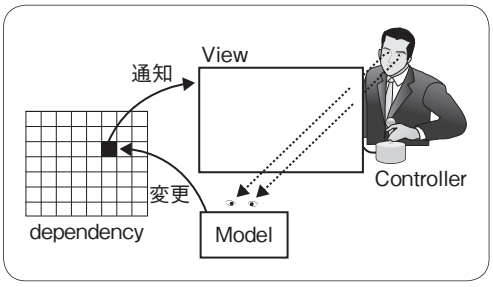
ここでMVC2の視点からMVCを評価してみると——ViewとControllerが密結合なのは良いとして——「ViewやControllerはModelを知っているが、Modelはそのほかの要素については何も知らない」という状態になっていないのではないかと不信感を持つでしょう。確かに、ModelからViewに対して状態変化を通知することができないと、Viewは再表示を行うことができないため、ModelとViewが密結合になっているように感じられます。

そこで、この通知をできる限り「間接的」に行わせ、ModelとViewを疎結合にするしくみとして、MVCを実現したSmalltalk[注5]では、dependencyというブロードキャスティング機構(不特定多数に同じ情報を同ときに送るしくみ)を利用することによって、ViewとModelの結合度を疎にし、Modelの変更をViewに知らせることにしています[注6](**図4**)。

▼図3　MVC



▼図4　依存性を利用するMVC(参考『使わないと損をするModel-View-Controller』http://aokilab.kyoto-su.ac.jp/documents/MVC/page1.html)



JSP Model 2、Seaside

MVCはそのあと、Smalltalk以外の言語でも利用されるようになってゆき、POSA本[注7]やGoFのデザインパターン本などでObserverパターンによるMVCの実現などが紹介されるようになっていきます。そして、1990年代後半からのインターネットとJavaの普及に伴い、現在のMVC2(Javaの場合ですが、[JSP：View]-[Servlet：Controller]-[Javaアプリケーション：Model]の組み合わせは、JSPModel2と称されました。詳細はPart2を参照してください。

なお、MVCの元祖であるSmalltalkにも、Webアプリケーション用のフレームワークSeaside[注8]などがあります。SmalltalkのMVCが誕生から約40年を経て、どのように発展したのか、興味のある方はぜひ調べてください。

注5) 世界で最初のオブジェクト指向言語であり、MVCは世界で初めてSmalltalk-80によって実現されました。また、最もオブジェクト指向の原理に合致しているとされています(オブジェクト指向の流派によって異なる解釈もあります)。

注6) SmalltalkのMVCはその洗練度順に「Controllerが頑張るMVC」「依存性を利用するMVC」「プラガブルを利用するMVC」の大きく3つが存在しますが(参考『使わないと損をするModel-View-Controller』1987/11/11 青木淳)、ここで解説しているMVCは最もポピュラーと考えられる「依存性を利用するMVC」を対象としました。

注7) Pattern-Oriented Software Architecture, A System of Patterns(Wiley Software Patterns Series)Frank Buschmann,他(著)1996年7月

注8) 「SeasideへGO!!」梅澤真史(http://Web.ogis-ri.co.jp/otc/hiroba/technical/seaside/seaside1/)「Seaside は、Webでゆがめられてしまったり MVC ではなく、本物のオブジェクト指向によるMVCを実現しています」(『SeasideへGO!!』から)とは、ここまで楽しく読み進んでいただけた方にはなんとも刺激的な文章ですね。

Part2 Webアプリケーションと MVC

Part2では、JavaのWebアプリケーション[注9]のMVC(正確にはMVC2を指します)について解説します。はじめに一般的なWebアプリケーションのMVCについて解説し、その後Javaの技術と紐付けて解説します。**図5**にWebアプリケーションの基本的な構成要素をまとめました。

クライアント[注10]上のブラウザからHTTPリクエスト[注11]が送信され、Webサーバ[注12]がリクエストを受信し、アプリケーションがDBと連携して処理を行い、HTMLデータを作成し、HTTPレスポンスとしてHTMLデータが送信されます。

Webアプリケーションに MVCを当てはめる

MVCを**図5**に当てはめると、Webアプリケーションの部分が、MVCの3つの役割に分けられることになります(**図6**)。

Controllerは、HTTPで送信されてきた情報を解釈して適切なModelに情報を渡し(**図6-①**)、Modelは、DBと連携して処理を行ってModelの状態を更新します(**図6-②**)。Contollerは Viewに対して処理を指示し(**図6-③**)、Viewは Modelの状態を取得してHTMLデータを作成します(**図6-④**)。

ブラウザの役目はViewという誤り

よく初心者が勘違いしてしまうのですが、ブラウザはViewではありません。そもそもブラウザはMVCの構成要素ではないのです。ブラウザは、Webアプリケーション(より詳細にいえばViewであるJSP)から送られたHTMLを受け取り、そのHTMLに従った処理(画面の表示やリクエストの送信)をするだけであり、MVCどころか、開発対象のWebアプリケーションですらないのです[注13]。

では、WebアプリケーションとMVCの関係が整理できたところで、次は、Javaの技術と紐付けて解説します。

MVCに関するJavaの技術

Javaでは、サーバ側のWebアプリケーションを作成するためのさまざまな仕様が存在し、Java EE(Java Platform,Enterprise Edition)としてまとめられています。これらの仕様の中で、Webアプリケーションを実現する最も基本的

注9) 複数のユーザがインターネットを通じてデータベースにアクセスし、安全に情報の読み書きするために作られたアプリケーションを指します。

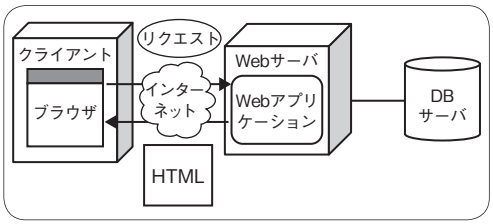
注10) 従来はクライアントといえばデスクトップPCを指すことが多かったですが、昨今では、業務システムでもタブレットなどがクライアントとして利用されています。

注11) HTTPはHTMLや画像などを通信するためのプロトコルであり、クライアントからWebサーバに対しての要求を「リクエスト」、その回答を「レスポンス」といいます。

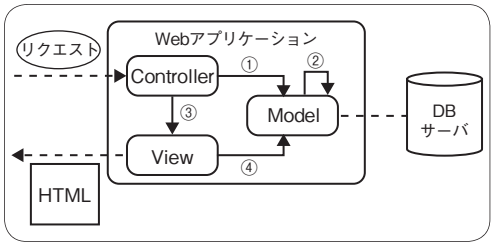
注12) クライアントとHTTP通信のやり取りを行い、静的なHTMLファイルや画像、もしくは(後述するWebコンテナ上で)Webアプリケーションを動作させ、その結果をクライアントへ返却するためのサーバ。

注13) Ajaxのように、ブラウザ上でJavaScriptプログラムを実行しサーバ通信やHTML要素の作成を行う場合は、ブラウザ上のJavaScriptプログラムがControllerやViewに含まれることになります。この場合も、ブラウザ自体はViewに含まれません。

▼図5 Webアプリケーションの主な構成要素



▼図6 WebアプリケーションのMVC



な仕様は、ServletとJSPです。

Servletのしくみ

Servletは、1997年にバージョン1.0がリリースされた古い歴史をもつ仕様です[注14]。Servletの仕様に従ったJavaのクラス(Servletクラスと呼ぶことにします)を作成し、配備記述子ファイル(web.xml)と呼ばれるファイルに登録した後、Webコンテナ[注15]とよばれる実行環境に配備すれば、Webアプリケーションとして動作します。

Servletクラスのサンプルを**リスト1**に示します。

注14) 最新はバージョン3.1(2014年2月時点)

注15) オープンソースの有名な製品としてApache Tomcatがあります。

▼リスト1　Servletクラスのサンプル

```
package sample;

import java.io.*;
import javax.servlet.*;
import javax.servlet.http.*;

public class HelloServlet extends HttpServlet {  ―――❶

    public void doGet(HttpServletRequest request, HttpServletResponse response)  ―――❷
            throws ServletException, IOException {
        String message = "Hello World!";
        PrintWriter out = response.getWriter();
        out.println("<html>");
        out.println("<head>");
        out.println("<title>Sample</title>");
        out.println("</head>");
        out.println("<body>");
        out.println("<h1>" + message + "</h1>");
        out.println("</body>");
        out.println("</html>");
    }
}
```

▼リスト2　配備記述子ファイルのサンプル

```
<?xml version="1.0" encoding="UTF-8"?>
<web-app xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xmlns="http://java.sun.com/xml/ns/javaee"
  xsi:schemaLocation="http://java.sun.com/xml/ns/javaee
    http://java.sun.com/xml/ns/javaee/web-app_2_5.xsd"
  id="WebApp_ID" version="2.5">

  <servlet>
    <servlet-name>HelloServlet</servlet-name>
    <servlet-class>sample.HelloServlet</servlet-class>―――❶
  </servlet>
  <servlet-mapping>
    <servlet-name>HelloServlet</servlet-name>
    <url-pattern>/hello</url-pattern>―――❷
  </servlet-mapping>

</web-app>
```

Servletクラスは、Servlet APIのクラスであるHttpServletを継承(**リスト1-①**)し、メソッドをオーバーライド(**リスト1-②**)して作成します。オーバーライドするメソッドは、HTTPのメソッド(GETメソッドやPOSTメソッドなど)に応じて、必要なものをオーバーライドします。**リスト1**のサンプルは、GETメソッドに対応したメソッドをオーバーライドしたものです。処理の内容の解説は省略しますが、HTMLのデータを出力する処理が行われていることが読みとれます。

次に、配備記述子ファイルのサンプルを**リスト2**に示します。

配備記述子ファイルには、Servletクラスの完全修飾クラス名(パッケージ名+クラス名:**リスト2-①**)と、URLパス(**リスト2-②**)を記述します。

Servletクラスと配備記述子ファイルが配備されたWebコンテナを起動し、ブラウザでアクセスすると、**図7**の画面が表示されます。

▼図7　画面のサンプル(Servletクラス)

JSP(JavaServer Pages)のしくみ

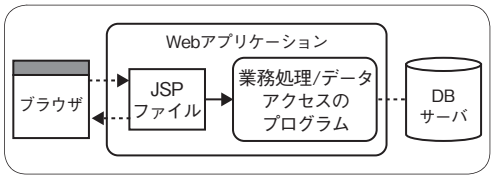
JSPは、1999年にバージョン1.0がリリースされた仕様です[注16]。JSPは、ServletのようにJavaのクラスを作成する必要もないですし、配備記述子ファイルに登録する必要もありません。代わりに、JSPファイルと呼ばれるHTMLに似たファイルを作成しWebコンテナに配備するだけです。Javaのプログラムを記述したい場合は、スクリプトレット(**リスト3-①**)と呼ばれる特別なタグを任意の場所に埋め込み、Javaのプログラムを記述することができます。JSPファイルのサンプルを**リスト3**に示します。

JSPファイルを配備したパスに対しブラウ

注16) 最新はバージョン2.2(2014年2月時点)。

▼図8　画面のサンプル(JSPファイル)

▼図9　JSP Model 1



▼リスト3　JSPファイルのサンプル

```
<html>
  <head>
    <title>Sample</title>
  </head>
  <body>
    <h1>
    <% out.println("Hello World!"); %>❶
    </h1>
  </body>
</html>
```

ザからアクセスすると、図8が表示されます。

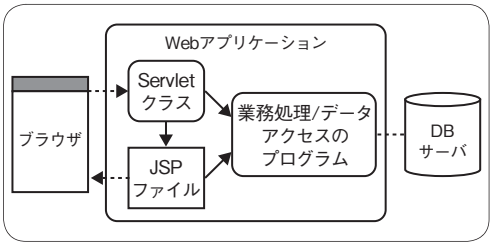
Java の MVC（JSP Model 1 と JSP Model 2 の違いとは）

JSPは、Servletよりも手軽にWebアプリケーションを作成できるので、登場した当初はServletを代替する技術として注目を集めました(そのように筆者は記憶しています)。Servletを使わずにJSPだけでWebアプリケーションを構築した場合の構造は、JSP Model 1[注17]と呼ばれます(図9)。

JSP Model 1は、手軽にWebアプリケーションを作成できますが、入力チェックや業務処理の呼出などのプログラムの記述が煩雑になってくる(つまりはControllerとViewの記述が1つのファイルに混在してしまう)ことで、可読性やメンテナンス性が悪くなってしまいます。この問題を解決するのが、JSP Model 2と呼ばれる構造です(図10)。

JSP Model 2では、リクエストを受け付ける部分と業務処理の呼出をServletクラスが行います。JSPは、業務処理の結果のデータを元

▼図10　JSP Model 2



注17) http://www.kirkdorffer.com/jspspecs/jsp092.html#model

▼リスト4　Servletクラスのサンプル（JSP Model 2）

```
package sample;

import java.io.*;
import javax.servlet.*;
import javax.servlet.http.*;

public class HelloServlet extends HttpServlet {

    public void doGet(HttpServletRequest request, HttpServletResponse response)
            throws ServletException, IOException {
        // 入力チェックや業務処理（Model）は省略します
        // 業務処理の結果を、message変数に格納するという想定です
        String message = "Hello World!";
        request.setAttribute("message", message);―――――❶
        getServletContext().getRequestDispatcher("/hello.jsp")―――――❷
        .forward(request, response);
    }
}
```

▼リスト5　JSPファイルのサンプル（JSP Model 2）

```
<html>
  <head>
    <title>Sample</title>
  </head>
  <body>
    <h1>
    <%= request.getAttribute("message") %>―――――❶
    </h1>
  </body>
</html>
```

にHTMLデータを作成します。これにより、Servletクラスと JSPファイルがそれぞれ得意な部分(Servletクラスは Controller、JSPファイルは View、そしてDBへのアクセスを含む業務処理は Javaアプリケーション)を分担することになり、可読性やメンテナンス性が向上します。

JSP Model 2を模した[注18]Servletクラスと JSPファイルのサンプルを**リスト4・リスト5**に示します。なお、配備記述子ファイルと表示される画面は、それぞれ**リスト2・図7**と同じです。

Servletクラス内で業務処理を呼出した後、requestスコープ[注19]と呼ばれるデータの入れ物に結果を格納(**リスト4-①**)し、JSPファイルに処理を指示しています(**リスト4-②**) 。JSPファイルでは、requestスコープの中からデータを取り出しHTMLに埋め込んでいます(**リスト5-①**)。

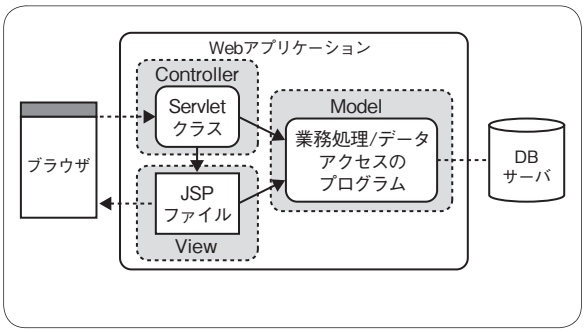
JSP Model 2は**図11**に示すように、MVCの構造で表すことができます。

JSP Model 2の構造が、Javaの基本的なMVCです(MVC2の「2」は、JSP Model 2の「2」から拝借したものとも言われています)。

ServletやJSPは過去の遺物になるか?

Part2では、ServletやJSPを用いたJavaの最も基本的と考えられているMVCについて解説しました。しかし、ServletやJSPを直接利用すると不便なことがあります。たとえば、リクエストの種類ごとにServletのクラスを作成し配備記述子ファイルに登録しなければならず、大規模なシステムになるとクラス数や配備記述子ファイルが肥大化してメンテナンス性が悪くなってしまいます。

▼図11　JSP Model 2とMVC



ServletやJSPは今やWebアプリケーションの基本的な技術要素として新人研修で利用されることはあっても、実際の開発現場では、より簡単で効率的にWebアプリケーションを構築できるフレームワークを利用し、直接利用されることは少なくなりました。Part3では、フレームワークを利用した開発現場のMVCについて解説します。

Part3 開発現場のMVC

どのようにMVCが使われているのか

ここまでで、JavaのWebアプリケーションの基本的なMVCについて解説してきました。Part3では、実際に開発現場でWebアプリケーションを開発するうえで必要な知識として、次の2点について解説します。

・MVCフレームワーク
・Modelの実装

MVCフレームワーク

Part2ではServletとJSPを直接用いてMVCを実現しましたが、ある程度の規模以上のアプリケーションでは、MVCフレームワークと呼ばれるフレームワークを導入することが多くあります(フレームワークについては28ページのコラム参照)。

注18) サンプルを簡単に動作できるようModelの部分に該当するJavaのアプリケーションを省略しているため、ここで提供するサンプルはModelの部分が貧弱になっていることに注意してください。

注19) スコープには、request・page・session・applicationの4種類があります。

MVCフレームワークとは、おもにController部分の共通的なしくみを提供するフレームワークのことです(**図12**)。

MVCフレームワークは、クライアントからリクエストを受信してからレスポンスを返すまでの一連の処理を実行してくれるフレームワークです。アプリケーション開発者は、「MVCフレームワークから実行される個別処理」とViewを用意してフレームワークに設定しておくだけで、あとはMVCフレームワークが一連のリクエストに対応する処理を実行してくれます。

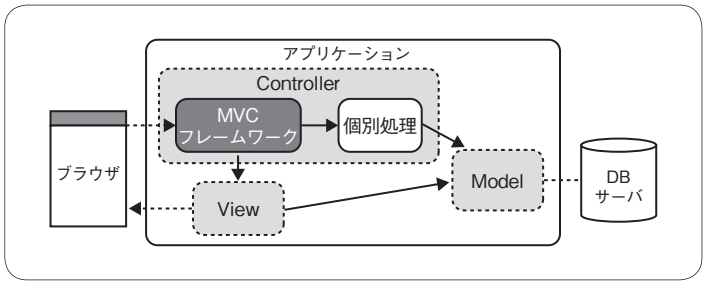
では、1つの例としてPart2の**リスト2**、**リスト4**で紹介したServletクラスの処理がどのように実現されるかを見てみましょう。今回は、MVCフレームワークの中でも最近とくに注目されているSpring MVCで実装した例を紹介します。

リスト6は**図12**の個別処理に該当する部分を、Spring MVCで実装した例です。Spring MVCでは@Controllerアノテーション(**リスト6-①**)を設定したControllerクラスに、個別処理をメソッドとして実装して(**リスト6-②**)、HTTPリクエストとの紐付けを@RequestMappingアノテーションで定義します(**リスト6-③**)。@RequestMappingアノテーションを設定したメソッドでは、Viewに受け渡す値を引数のModelオブジェクトに設定して(**リスト6-④**)、最後にView名をreturnします(**リスト6-⑤**)。あとはSpring MVCが、returnされた「hello」という文字列からhello.jspに遷移してくれます[注20]。

ポイントとしては、まずPart2の**リスト4**の②にあった「次の画面に遷移する処理」自体がなくなっていることが挙げられます。アプリケーション開発者はどの画面に遷移するかという情報を指定するだけでよく、あとはフレームワーク側が画面遷移を実行してくれます。もう1つのポイントとして挙げられるのが、Part2の**リスト4**ではHttpServletRequestやHttpServletResponseといったServlet APIが隠蔽されていることです。この2つのServlet APIを直接使うことでHTTPに関する多くの細かい処理を行うことができるのですが、逆に言えばアプリケーション開発者にこれらのAPIを公開し

注20) Spring MVCの共通設定として、returnされたView名を元にどのViewに遷移するかを設定しておく必要があります。たとえば「/WEB-INF/jsp/{View名}.jspに遷移させる」と言った設定が可能です。

▼図12　MVCフレームワーク



▼リスト6　Controllerクラスの例

```
@Controller ―――――――❶
public class HelloController {
    @RequestMapping(value = "/hello", method = RequestMethod.GET) ―❸
    public String hello(Model model) {
        model.addAttribute("message", "Hello World !!"); // ―❹
        return "hello"; ―❺
    }
}
```

❷

てしまうと、問題のある処理が実行されてしまう可能性が高くなります。Spring MVCでは独自にModelオブジェクトを用意していて、このModelオブジェクトではViewに渡すModelオブジェクトの制御しかできないようになっています。このようにして、アプリケーション開発者の実装を制限しているのです[注21]。

以上のように、MVCフレームワークを適用することで、アプリケーション開発者が複雑な処理を意識することなく単純に実装できるようになり、また不用意に問題のある処理が実行されるのを防ぐことができます。このほか、MVCフレームワークを適用するメリットとしては、すべてのリクエストに共通するような処理を拡張処理として適用できることが挙げられます。このようにして、アプリケーション開発者に意識させることなく共通処理を適用することもできるのです。

まとめとして、現場でもよく使用される主なMVCフレームワークを紹介しておきましょう(**表1**)。

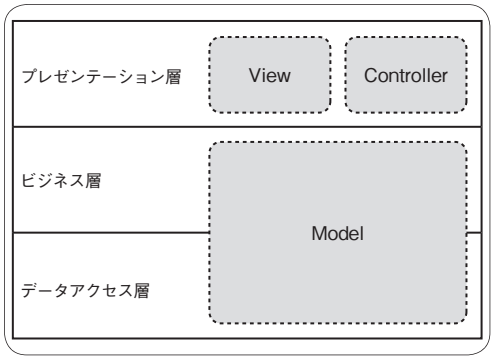
Modelの実装

ここまでMVCをターゲットに、Webアプリケーションの画面周りの話を中心に話を進めてきました。しかしWebアプリケーションは画面だけで構成されるわけではなく、むしろMVCでいうところのModelの部分が非常に重要です。本Partの最後に、Modelの実装についても簡単に見ておきましょう。

Webアプリケーションに限らず、アプリケーションを開発する際にはまず全体を大きく「レイヤ」と呼ばれる論理的な層に分割します。一般的な分割方法は3層構造で、**図13**のようになります。MVCとの対応付けも確認しておきましょう。

ControllerとViewは、クライアントにユーザインタフェースを提供する層で、プレゼンテーション層に配置されます。そしてModelはビジネス層とデータアクセス層にまたがって配置されます。言い換えると、Modelは大きくビジネス層とデータアクセス層に分割されています。ここからは、ビジネス層とデータアクセス層に

▼図13　一般的なレイヤ構造とMVC



▼表1　おもなMVCフレームワーク

フレームワーク	URL	概要
Struts 1.x	https://struts.apache.org/release/1.3.x/index.html	フレームワークの草分け的存在で、現在も最も利用者が多いと想定される。ただし開発はすで終了している
Struts 2.x	https://struts.apache.org/release/2.3.x/index.html	元はWebWorkというフレームワークがApache Foundationに寄贈されたもので、Struts 1.xの後継とされるフレームワーク
Spring MVC	http://docs.spring.io/spring/docs/current/spring-framework-reference/html/mvc.html	Spring Frameworkに含まれるMVCフレームワークで、Spring Frameworkの普及と共に最近特に注目されている
JSF	https://jcp.org/en/jsr/detail?id=344	Java EEで定められている、MVCフレームワークの仕様。JSFでアプリケーションを開発するためには、MyFacesやMojarraなどの実装が必要

注21) 必要に応じて、Controllerクラスのメソッド引数としてHttpServletRequestやHttpServletResponseを指定することで、直接使用することも可能です。

ついて解説します。

ビジネス層の実装

ビジネス層の役割は、アプリケーションのビジネスロジックを実装することです。特定の技術に依存することなく、システム化対象のビジネスルールや業務ロジックを実装することがポイントです。

ビジネス層の実装方式は、大きくトランザクションスクリプト方式とドメインモデル方式に分けられます。端的に言ってしまうと、トランザクションスクリプト方式は処理とデータを完全に分離して実装する方法(**図14**)、ドメインモデル方式は処理とデータの両方を持ったオブジェクト(ドメインモデル)を互いに通信させて実装する方法です(**図15**)。

トランザクションスクリプト方式は昔ながらの「処理に注目する方式」で設計を行うので、オブジェクト指向に習熟していない開発者であっても設計が可能な点が特徴です。ドメインモデル方式の場合はオブジェクト指向に習熟していないと設計がそもそもできないと言う難点はあるものの、オブジェクト指向の継承やポリモルフィズムなどのテクニックを使って、より拡張性や変更容易性の高いシステムに仕上げることができます。どちらが良いということは単純には言えませんが、開発規模や開発者の技術レベルによって、どちらの方式を用いるかを決定す

COLUMN

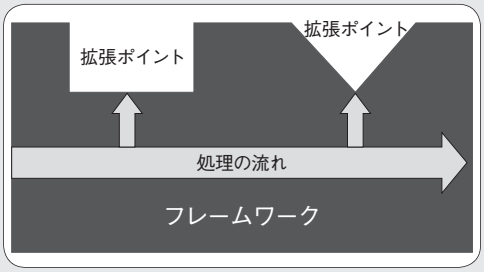
「フレームワークとは何か」

フレームワークとは、アプリケーションを構築する際の基盤となる共通プログラムのことです。フレームワークでは対象とする処理の一連の流れを共通的な処理として実装しています(**図A**)。そして、アプリケーションごとに異なる個別処理については拡張ポイントとして未実装となっています。つまりフレームワークは、「半完成品」と表現することができます。

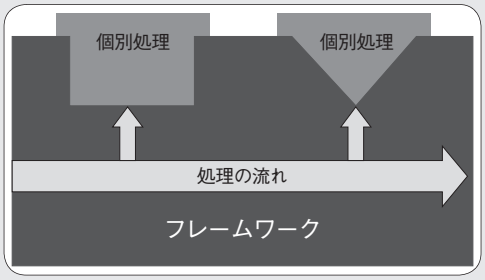
アプリケーションの開発者は、個別処理を作成してフレームワークの拡張ポイントに適用します。そうすることで全体として「完成品」となり、一連の処理を実現できるようになるのです(**図B**)。

Javaでアプリケーションを開発する場合は、オープンソースのフレームワークを組み合わせて開発することが一般的になってきています。また実際のアプリケーション開発では、オープンソースのフレームワークを組み合わせたものを、さらにラップする独自フレームワークを作成することが多くあります(**図C**)。

▼図A　フレームワーク



▼図B　個別処理の適用



▼図C　フレームワークの組合せ

る必要があります。

データアクセス層の実装

データアクセス層の役割は、データアクセス層より上位に対してデータベースの存在を隠蔽することです。つまりデータアクセス層では「データベースに格納されたデータ」と「オブジェクト」の間の変換を行う必要があります。このオブジェクトの世界とリレーショナルデータベースの世界の変換を行うことを、O/Rマッピング(Object/Relational Mapping)と呼びます。

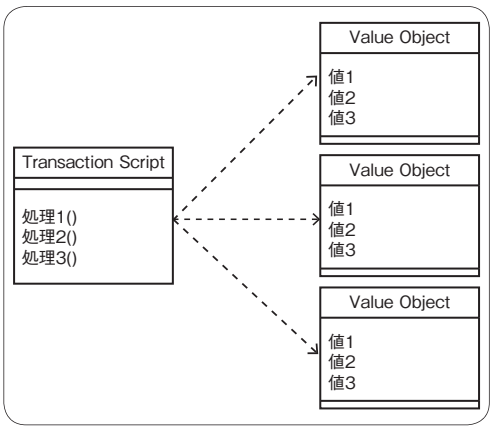
単純なO/Rマッピングでは、「Javaのクラス」と「テーブル」、そして「プロパティ」と「カラム」をマッピングさせるのが一般的です(**図16**)。

さらには「Javaクラス間の関連」と「テーブル間のリレーションシップ」のマッピングも行う必要があります(**図17**)。この関連とリレーションシップのマッピングが非常に難しく、設計者が頭を悩ませるところです。

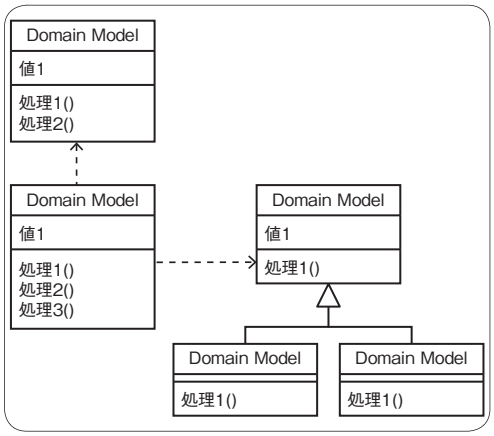
データアクセス層ではこのO/Rマッピングを実現するために、一般的にはO/Rマッピングフレームワークと呼ばれるフレームワークを採用します。O/Rマッピングフレームワークは O/Rマッピングを容易にするしくみを提供し、また複雑なデータベースアクセスのAPIを隠蔽してくれます。さらにはキャッシュのしくみを提供してくれるものもあります。現場でよく使用されるおもなO/Rマッピングフレームワークを**表2**に挙げます。

とくに比較されるのがHibernateとMyBatisです。過去には、設定を行うだけでSQLを自動生成してくれるHibernateが大きくもてはやされたことがありました。ただHibernateは、Hibernateを熟知していない開発者が不適切な使い方をしてしまうと無駄なSQLが大量に発行されてしまうリスクを抱えています。ある

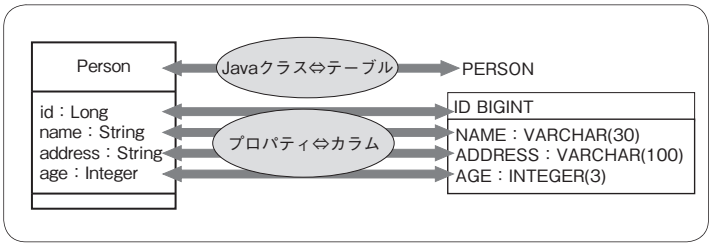
▼図14　トランザクションスクリプト方式



▼図15　ドメインモデル方式



▼図16　クラスとテーブルのマッピング



Hibernateを採用しているアプリケーションで不適切な使い方がなされてしまい、1つの画面を表示するのに10,000回を超えるSQLが発行されて大幅なパフォーマンスチューニングが必要となった、という話を聞いたことがあります。

この例のようなHibernateの問題点が世の中でささやかれはじめ、最近ではMyBatisの評価が上がってきているように感じます。MyBatisではSQLの自動生成は一切行われず、発行するすべてのSQLを設定ファイルに記述しておく必要があります。ただ逆に言えば、設定ファイルで記述したSQL以外が発行されることは絶対にありません。つまりSQLを管理するという観点では優れていると言えます。

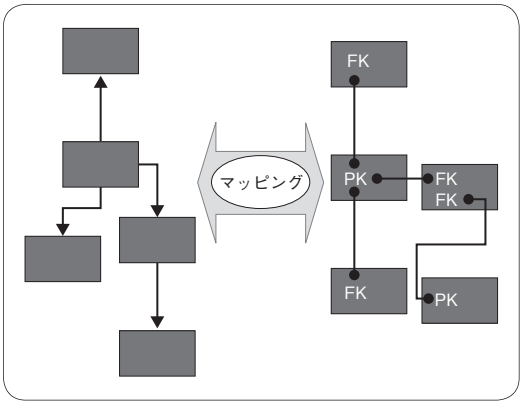
Hibernateも、Hibernateに熟知した開発者が適切に使用すれば非常に大きなメリットがあると言えるので、どのフレームワークを選択するかは状況によって変わってきます。ただ前評判だけで選択するのは非常に危険であることだけは覚えておいてください。

まとめ

Javaでアプリケーションを開発する際は、オープンソースのフレームワークを採用することが当たり前となってきています。その先駆けとなったのが本章でも軽く触れた、MVCフレームワークのStrutsです。Strutsの登場以降、さまざまなMVCフレームワークやO/Rマッピングフレームワークが登場しました。現在も多くのフレームワークが生まれ、また既存のフレームワークもめまぐるしく機能拡張を続けています。

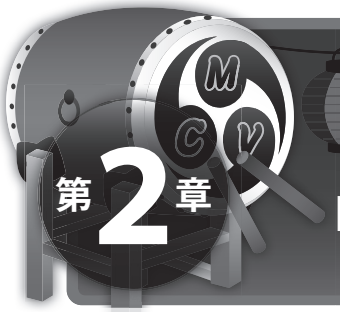
アプリケーション開発を行うためには、それらの豊富なフレームワーク群の中から適切なフレームワークを見つけて組み合わせていく必要があります。そのためにも、今回テーマとしているMVCモデルにおけるModel/View/Controllerの役割、さらには本稿で触れた3層構造における各層の役割を十分に理解しておくことが必要です。SD

▼図17　関連とテーブルのマッピング



▼表2　おもなO/Rマッピングフレームワーク

フレームワーク	URL	概要
Hibernate	http://hibernate.org/orm/	設定ファイルやアノテーションで、クラスとテーブルのマッピングを定義する、非常に高機能なO/Rマッピングフレームワーク。複雑なO/Rマッピングや関連のマッピングが可能で、SQLが自動生成されるのが特徴
MyBatis	http://www.mybatis.org/	設定ファイルにSQLを定義することでマッピングを行う、比較的シンプルなO/Rマッピングフレームワーク。SQLの自動生成は行われない
Spring JDBC	http://docs.spring.io/spring/docs/current/spring-framework-reference/html/jdbc.html	Spring Frameworkに含まれるO/Rマッピングフレームワーク。SQLを直接プログラム中に記述することでマッピングを行う



第2章 JavaScript編 ――クライアントサイドMVCの実装をBackbone.js、AngularJSを使って学ぼう

Writer 濱田 廣貴 ㈱日立ソリューションズ

ModelやViewといった用語を聞くと、サーバサイドのフレームワークを連想してしまう開発者も多いのではないでしょうか。もともとMVCはGUI開発などの複雑性を解消するものとして発展したもので、実はクライアントサイドでもよく導入されているのです。2章ではクライアントサイドMVCの一例として、JavaScriptのMVCを解りやすく解説します。サーバサイドとはちょっと違ったMとVのカンケイを一緒に見ていきましょう。

JavaScriptでMVC？

クライアントサイドMVCとは

まずはクライアントサイドMVCの特徴から解説します。サーバサイドのMVCと比較して、クライアントサイドのMVCでは**ModelとViewがObserverパターンになっている**という特徴があります。Observerパターンとは、あるオブジェクトが他のオブジェクトを監視(Observe)し、監視対象のオブジェクトからの通知を受けて何かしらの処理を行うためのデザインパターンです。

具体的にサーバサイドとクライアントサイドのMVCを比べてみましょう。サーバサイドのMVCのアーキテクチャはおおよそ**図1**のような形になっています。ModelとViewは直接結びつかず、Controllerを介してやりとりします。Viewの描画はリクエストに連動して行われます。

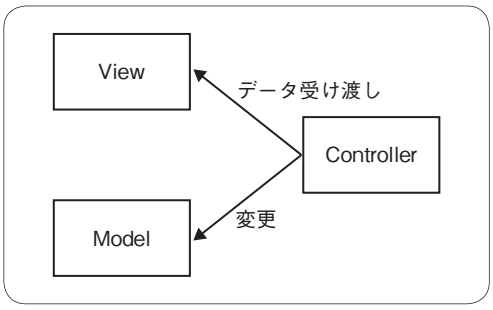
一方で、クライアントサイドのMVCはおおよそ**図2**のような形になっています。クライアントサイドの場合、ViewはModelからデータをもらって描画を行います。この際、Controllerは介さずに、Modelの変更を機にViewの描画が行われるのがポイントです。

Modelは変更があったタイミングでイベントが発火するようにします。Viewはこのイベントを監視し、描画の処理をこのイベントに紐付けます。これにより、Modelの変更によってViewを描画するということが実現できます。ちょうどこの部分がObserverパターンになっています。

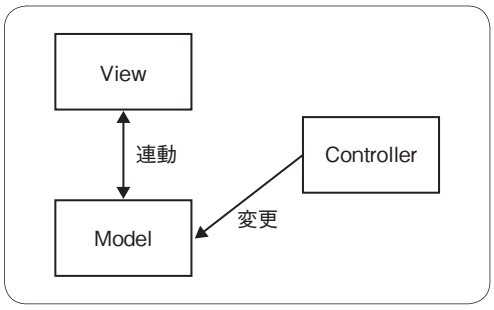
なぜMVCなのか

以上でクライアントサイドのMVCの特徴が理解できました。このような設計手法をJavaScriptに取り入れる利点とは何でしょうか。

▼図1　サーバサイドMVC



▼図2　クライアントサイドMVC



それを理解するために、よくある典型的なソースコードからMVCに分離しない場合の問題点を確認していきましょう。

ここでは一例として、**図3**に示すアプリケーションを実装します。このアプリケーションは赤・緑・青の各テキストボックスに数値を入力して色を作成します。「色の登録」ボタンを押すと、作成した色が16進カラーコードの形式に変換されてリストに追加されます。また、リストの項目をクリックすることで、作成した色をテキストボックスに読み込むことができます。

本アプリケーションのHTMLを**リスト1**に示します。本アプリケーションを完成させるために、どのような処理を追加するべきか考えてみてください。

JavaScript部分はおそらく**リスト2**のようになるのではないでしょうか。このコードは大きく2つの部分に分けられます。1つは追加ボタンを押したときの処理(①)、もう1つはリストの項目をクリックしたときの処理(②)です。

これで期待どおりの動作はしますが、表示部分が状態を保持しているという問題があります。具体的にはリストの項目が、入力された色の値を保持しています。表示部分が状態を保持していると、レイアウトの変更であっても内部処理を考慮した修正を行う必要があり、保守性が低下します。

この問題を解消するためには、表示部分(View)と状態の管理(Model)を分ければ良さそうです。

▼図3　色リスト管理アプリ

▼リスト1　index.html

```
<body>
  <div id="rgb">
    赤: <input type="number" id="red" value="0"><br>
    緑: <input type="number" id="green" value="0"><br>
    青: <input type="number" id="blue" value="0"><br>
    <button id="add-color">色の登録</button>
  </div>
  <hr>
  <h3>色リスト</h3>
  <ul id="color-list"/>
</body>
```

COLUMN

MVCの定義

単に「MVC」と言った場合にはオリジナルであるSmalltalk-80のMVCを指すことが多く、JavaScriptのMVCフレームワークはこのオリジナルMVCとは厳密には異なるアーキテクチャであることがほとんどです。よってJavaScriptのMVCフレームワークのことをよくMV＊などと表現します。

本章ではオリジナルMVCとは異なるのを承知のうえで、MVCという表現を用いることにします。

それではViewで保持している色の状態をModelに分離してみましょう。「色を登録」ボタンを押したときの処理は**リスト3**のようになります。

ここではHTMLのテキストに格納していた色の情報を、メモリ上の配列で管理するように変更しています。このようにすると、表示部分に変更があっても修正が少なくて済みます。ただし、このままだと配列に要素が追加されたときにViewを描画するということができません。そこで、Observerパターンを導入してこれを実現できるようにします。

次節ではBackbone.jsを使って実際にModelとViewをObserverパターンを使って紐付けてみます。

Backbone.jsとは?

特徴

Backbone.jsはJeremy Ashkenas氏が開発した、軽量でシンプルなMVCフレームワークです。オープンソースであり、MITライセンスのもとで配布されています。なお、執筆時点のバージョンは1.1.1です。Backbone.jsでははじめからイベントのしくみが用意されており、Observerパターンを簡単に構築できます。

例として、先ほどjQueryを使って実装した色リスト管理アプリをBackbone.jsで実装しなおしてみましょう。Backbone.jsを使った開発

▼リスト2　app.js

```
function rgbToHex(red, green, blue) {
  // RGBから16進カラーコードを計算する処理
}
function hexToRGB(hex) {
  // 16進カラーコードからRGBを計算する処理
}
//「色の登録」ボタンを押したときの処理
$('#add-color').click(function() {  ❶
  var red = $('#red').val();
  var green = $('#green').val();
  var blue = $('#blue').val();
  $('<li>').text(rgbToHex(red, green, blue))
           .appendTo('#color-list');
});
// リストの項目をクリックしたときの処理
$('#color-list').on('click', 'li', function() {  ❷
  var hex = $(this).text();
  var rgb = hexToRGB(hex);
  $('#red').val(rgb.red);
  $('#green').val(rgb.green);
  $('#blue').val(rgb.blue);
});
```

▼リスト3　Modelを導入した例

```
var colors = [];
$('#add-color').click(function() {
  colors.push({
    red  : $('#red').val(),
    green: $('#green').val(),
    blue : $('#blue').val()
  })
});
```

▼リスト4　index.html（リスト1）に追加

```
<script src="underscore.js"></script>
<script src="jquery-2.1.0.js"></script>
<script src="backbone.js"></script>
```

をはじめるには、公式サイト[注1]からダウンロードしたBackbone.js本体のスクリプトファイルをHTMLのscriptタグで読み込めばOKです(**リスト4**)。

また、Backbone.jsはjQueryおよびunderscore.jsというライブラリに依存していますので、それらのファイルも必要です。underscore.jsとは便利な関数を集めたユーティリティライブラリです。こちらもunderscore.jsの公式サイト[注2]からダウンロードできます。

Backbone.ModelとBackbone.Collectionによる Modelの定義

Backbone.Modelはその名のとおりMVCにおけるModelを表します。また、Backbone.CollectionはBackbone.Modelの集合を扱うためのもので、これもMVCにおけるModelにあたります。

Backbone.Modelを使った色の状態のModel定義を**リスト5**に示します。Backbone.jsではBackbone.Model.extendという関数の呼び出しによりModelを定義します(**リスト5-①**)。その関数の引数にはModel定義のために必要な情報をObjectの形式で渡します。この例では色が持つ状態ということで、赤・緑・青の3色の値を保持し、それぞれのデフォルト値が0であることを指定しています(**リスト5-②**)。

さて、本アプリでは色のリストを管理するのでした。こういった集合はBackbone.jsではBackbone.Collectionを使うと便利です(**リスト6**)。

Backbone.CollectionもBackbone.Modelと同様に、extendという関数を使って定義します。ここでは**リスト5**で定義したColor Modelの集合を扱うことを指定しています。

さて、このColor ModelやColors Collectionはただの JavaScriptのオブジェクトではありません。自身が変更されるときにイベントを発火するようになっているのです。たとえば、Colors Collectionは自身に新しく要素が追加されたときに"add"というイベントを発火するようになっています。addイベントとViewの紐付けについては後ほど説明します。

Backbone.Viewによるイベント処理

続いてViewを定義します。Backbone.Viewには役割が2つあります。ユーザからの入力を受け取ってModelを変更する処理と、Modelの変更を監視してHTMLを描画する処理です。

まずはユーザからの入力を受け取る部分を見てみましょう。**リスト7**はテキストボックスから色の登録を行う部分のJavaScriptのコード

注1) http://backbonejs.org
注2) http://underscorejs.org

▼リスト5　models/color.js

```
var Color = Backbone.Model.extend({  ―❶
  defaults: {  ―――――――❷
    red  : 0,
    green: 0,
    blue : 0
  }
});
```

▼リスト6　collection/colors.js

```
var Colors = Backbone.Collection.extend({
  model: Color
});
var colors = new Colors();
```

▼リスト7　views/newcolor.js

```
var NewColor = Backbone.View.extend({

  el: '#rgb',  ―――――――❶

  events: {
    'click #add-color': 'newColor'  ―❷
  },

  newColor: function() {  ―――――――❸
    colors.add({
      red  : $('#red').val(),
      green: $('#green').val(),
      blue : $('#blue').val()
    });
  }
});
```

です。

elというフィールドに、HTMLで作成したid属性を指定しています(**リスト7-①**)。この指定により、すでに存在するHTMLの上にBackbone.ViewのObjectを作成することができます。なお、HTMLは**リスト1**にあるjQueryと同じものです。

eventsというフィールド(**リスト7-②**)はclickイベントの処理についての記述です。これは、add-colorというid属性を持つ要素(「色の登録」ボタン)がクリックされたときにnewColorという関数を呼び出す、という意味です。

イベントのハンドラであるnewColor関数(**リスト7-③**)の処理の中身を見てみましょう。この部分の記述は**リスト3**で示したものとほとんど同じです。1つ違うのはcolorsというのがただの配列ではなく、Backbone.Collectionであるという点です。

Backbone.Viewによる描画

最後に色のCollectionからリストを描画する部分の実装を見てみます(**リスト8**)。

ここではthis.listenTo(colors, 'add', this.addColorView)という記述(**リスト8-①**)に注目しましょう。これがObserverパターンのキモ、Modelの変更を監視している部分です。つまり、Colors Collectionがaddイベントを発火した際に、自身のaddColorView関数(**リスト8-②**)が呼び出されるようになります。addColorView関数では、色のリストの項目の描画処理を行います。

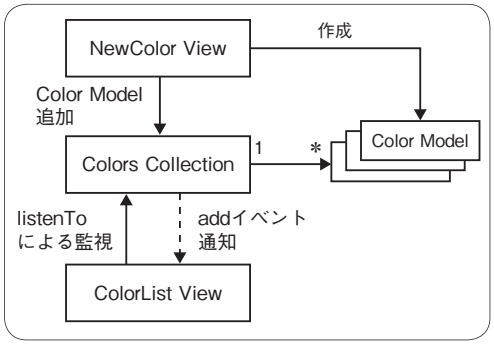
以上の4つのオブジェクトの関係を図示すると**図4**のようになります。ちょうどColors CollectionとColorList Viewの関係がObserverパターンになっていることがわかります。

AngularJSとは?

特徴

前節で、Backbone.jsを利用すればMVCの考え方に基づいたアプリケーションをシンプルに構築できることを示しました。しかし、テキストボックスからColor Modelを作る部分はred、green、blueごとの同様の記述を繰り返さなければならず、紋切り型のコードになってしまっています。実はこういったコードはAngularJSを使うと非常に簡潔に書くことがで

▼図4　Backbone.jsの構成



▼リスト8　views/colorlist.js

```
var ColorListView = Backbone.View.extend({

  el: '#color-list',

  initialize: function() {
    this.listenTo(colors, 'add', this.addColorView);  ❶
  },

  addColorView: function(color) {  ❷
    // リストの項目の描画処理
  }
});
```

きます。

AngularJSは、Google製のオープンソースのJavaScript MVCフレームワークです。AngularJSがもつ大きな特徴はデータバインディングを採用していることです。これはModelがもつデータとViewの表示を自動的に紐付けるしくみです。つまり、Modelの値が変更されれば自動的にViewが更新されるし、Viewの値(すなわちテキストボックスの値などです)が変更されれば自動的にModelの値も変化します。

本体のファイルは公式サイト[注3]からダウンロードできます。なお、執筆時点のバージョンは1.2.13です。AngularJSを使った開発をはじめるには、まずはBackbone.jsと同様、本体のスクリプトファイルをHTMLのscriptタグで読み込みます。また、bodyタグにng-appという属性を付けます(**リスト9**)。ng-appは、その属性がついているタグ以下についてAngularJSが処理を行うための記述です。サンプルではbodyタグに付けていますが、body以外の任意

▼リスト9　index.html(リスト1)への追加と変更

```
<script src="angular.js"></script>

<body ng-app>
```

▼リスト10　index.html

```
<script src="angular.js"></script>

<body ng-app>
<div ng-controller="ColorCtrl">   ❶
  </div>
  <div>
    赤: <input type="number" ng-model="red"><br>   ❷
    緑: <input type="number" ng-model="green"><br>
    青: <input type="number" ng-model="blue"><br>
    <button ng-click="newColor()">色の登録</button>   ❸
  </div>
  <hr>
  <ul>
    <li ng-repeat="color in colors">   ❹
    {{rgbToHex(color)}}   ❺
    </li>
  </ul>
</div>
</body>
```

▼リスト11　controllers/colorCtrl.js

```
function ColorCtrl($scope) {
  $scope.red   = 1;   ❶
  $scope.green = 1;
  $scope.blue  = 1;

  $scope.colors = [];   ❷

  $scope.newColor = function() {   ❸
    $scope.colors.push({   ❹
      red  : $scope.red,
      green: $scope.green,
      blue : $scope.blue
    };
  };
}
```

注3) http://angularjs.org

の場所に書くことができます。

ここからは、AngularJSを使って、再度色リスト管理アプリを実装してみます。

AngularJSによる色リスト管理アプリの実装

さて、それではサンプルアプリの実装の解説に入ります。AngularJSを使った場合、このサンプルアプリはHTMLファイルとJavaScriptの2つのみで実装できます。それぞれのソースコードを**リスト10および11**に示します。

Controller

AngularJSでは、ユーザからの入力の処理やModelのセットアップはControllerと呼ばれるオブジェクトで処理します。このControllerが担当する範囲を指定するために、divタグにng-controllerという属性を書いています(**リスト10-①**)。これでこのdiv要素以下のイベントの処理などにColorCtrlというControllerを使えるようになります。

リスト11はControllerの定義を書いたものです。

データバインディングの記述

まずは赤・緑・青の3色の値を入力する部分の実装から見ていきます(**リスト10-②**)。inputタグに付いているng-modelという属性がデータバインディングのための記述です。属性の値の部分にはModelの名前を指定します。この記述でred、green、blueという3つのModelを定義します。

Controllerを使って先ほど定義したModel(red、green、blue)のデフォルト値を設定してみましょう。Controllerの引数に$scopeというObjectが渡ってきます。実はこれがAngularJSのバインディングのキモです。ng-modelで指定したred、green、blueの各Modelは$scopeの同名のプロパティと自動的に紐付きます。ここではred、green、blueそれぞれに1というデフォルト値を設定しています(**リスト11-①**)。これだけの記述で赤・緑・青の各テキストボックスに自動的に1という値が入ります。

イベントの処理

次に、「色の登録」ボタンを押下したときの処理の実装を見ていきましょう。

まず、**リスト10-③**に示すように、ボタンの要素とクリックされたときのイベントハンドラの記述をHTMLに書きます。ボタンがクリックされたときのイベントハンドラはngClickという属性で指定します。この場合、Controllerにあるの newColorという関数が呼び出されます。

Controllerの実装を見てみましょう(**リスト11-③**)。Controller側では$scope.newColor = function(){}という形でイベントハンドラを定義します。イベントハンドラ内の処理(**リスト11-④**)は**リスト3**で示したコードとよく似た記述になっていますが、いくつか違いがあります。まず、$scopeにある各色の値はバインドされているので、わざわざテキストボックスから値を読み出す必要はありません。そしてcolorsは、$scope内に定義されている(**リスト11-②**)のでこれもバインドされます。つまり、colorsという配列に変更が加わるたびに、colorsを使って表示している部分が自動的に再描画されることになります。

リスト表示

最後に、colorsという配列を使って16進カラーコードのリストを出力する処理を見てみましょう(**リスト10-④**)。li要素に、ng-repeat="color in colors"という属性がついています。これはJavaの拡張for文のようなもので、colorsという配列の要素を1つずつ順に取り出してliタグを作ります。

liタグの中では{{rgbToHex(color)}}という記述でControllerにある同名の関数を呼び出し、色を16進カラーコード形式に変換して表示しています(**リスト10-⑤**)。colors配列に変更があると、自動的にこのリスト表示の部分も再描

画されることになります。

AngularJSの場合はHTMLの要素に追加されたng-modelなどの独自タグと、Controller内の$scope内の値がバインディングにより自動的に紐付きます(**図5**)。そのため非常に少ないコード量でModelの変更をViewに反映させることができます。

JSでMVCを実装するときの問題とは？

依存関係の解決

MVCフレームワークを使っての開発は役割ごとにファイルを分ける関係上、ファイル数が多くなりがちです。残念ながらJavaScriptには今のところ、言語仕様としてJavaのimport文のようなしくみがありません。一番手っ取り早いのはscriptタグを順番に並べる方法ですが、この方法は、とくに大規模開発でファイル数が増大すると管理が難しくなります。ここでは、こういった依存関係の問題を解決する方法の1つとしてRequreJSというライブラリを紹介します。

RequireJSでは各ファイルをdefine関数を使ってモジュールとして定義します。また、モジュール自体にも他のモジュールへの依存関係を記述することができます。モジュールを利用する側ではrequire関数を使って依存関係を指定します。

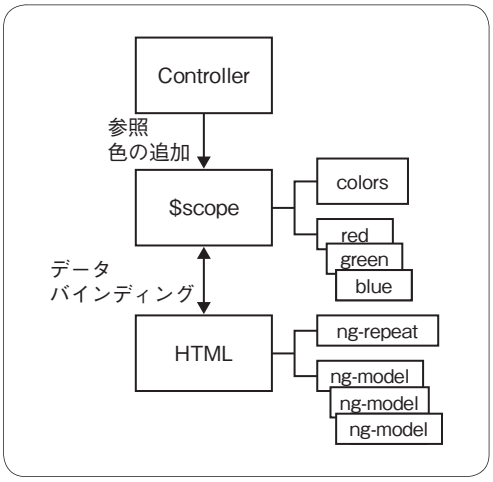
リスト7のコードを、RequireJSのモジュールとして定義する方法を**リスト12**に示します。モジュールの定義にはdefineという関数を使います。第1引数に依存するモジュールのリストを指定し、第2引数にモジュール本体の定義をコールバックの形で記述します。

さて、このモジュールをアプリケーションのエントリーポイントであるapp.jsから使うには、**リスト13**のように記述します。

このアプリケーションが利用する各ライブラリをRequireJSのモジュールとして利用するにはrequire.config関数を使います(**リスト13-①**)。ここではjQuery、underscore.js、Backbone.jsの各ライブラリをRequireJSのモジュールとして利用できるように設定しています。

そして、アプリケーションの起動に必要なモジュールをrequire関数で指定します(**リスト13-②**)。第1引数に依存するモジュールを書き、実際の処理は第2引数に関数として渡します(リ

▼図5　AngularJSの構成



▼リスト12　views/newColor.js

```
define([
  'underscore',
  'backbone',
  'jquery',
  'collections/colors',
], function(_, Backbone, $, Colors) {
  var NewColorView = Backbone.View.extend({
    …省略…
  });
  return NewColorView;
});
```

スト13-③)。

最後に本体のHTMLでRequireJSのスクリプトのみを読み込みます(**リスト14**)。data-mainという記述により、app.jsが自動的に読み込まれ、そこでさらに依存するColorモデルが読み込まれ、というように自動的に依存関係を解決してくれます。

RequireJSのほかには、Google Closure LibraryやRuby製のSprocketsなどのツールを活用する方法もあります。また、TypeScriptなど一部のAltJSは言語仕様として依存関係の記述ができるものもあります。こういった対処方法を知っておくことで、大規模なJavaScriptアプリケーションでも効率的にMVCによる開発を進めることができます。

おわりに

本章では、サーバサイドMVCとクライアントサイドMVCの違いを、ModelとViewのObserverパターンという点から解説しました。また、Backbone.jsやAngularJSといったMVCフレームワークでそれがどのように実現されているかを見てきました。

さて、MVCフレームワークはBackbone.jsやAngularJS以外にも数多くあります。ほかのMVCフレームワークではどのようなアーキテクチャになっているのでしょう？　それを簡単に調べられるのがTodoMVC[注4]というサイトです。このサイトではTODOアプリをさまざまなMVCフレームワークで実装したものが集められています。いろいろ試してみて、自分のお気に入りのフレームワークを見つけてはいかがでしょうか？ SD

注4) http://todomvc.com

▼リスト13　app.js

```
require.config({  ―――――――――❶
  shim: {
    underscore: {
      exports: '_'
    },
    backbone: {
      deps: ['underscore', 'jquery'],
      exports: 'Backbone'
    },
  },
  paths: {
    jquery: 'lib/jquery-2.1.0',
    underscore: 'lib/underscore',
    backbone: 'lib/backbone'
  }
});

require([  ―――――――――❷
  'models/color',
  'views/newColor',
  'views/colors'
], function(Color, NewColorView, ColorsView) {
  // アプリケーションの起動処理  ―――――――――❸
})
```

▼リスト14　index.html(リスト1)に追加

```
<script data-main="app" src="lib/require.js"></script>
```

第3章 PHP編 ——CakePHPを通してMVCを復習、FuelPHP/Symfonyで実践

Writer 星野 香保子(HOSHINO Kahoko) (有)テクノランド

本章では、PHPで開発を行う際に生じているMVCへの誤解を解き、MVCについての正しい理解を深めることを目的とします。また、日本で人気があるCakePHP、FuelPHP、Symfonyの各フレームワークとの関連について説明します。なお、本稿ではWebアプリケーション開発におけるMVCを対象にして述べていきます[注1]。

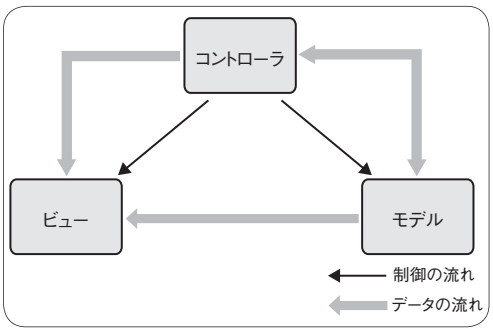
PHPでMVCを実装するのは問題ばかり?

PHPによるWebアプリケーション開発でMVCベースのフレームワークを使うのはもはや当然となっていますが、MVCに対して多少なりとも誤解している開発者がいると聞きます。

Webアプリケーション開発においてMVCを導入するメリットを簡単に述べると、生産性、拡張性、保守性の向上、テストの行いやすさなどが挙げられます。しかしMVCへの誤解が生じていると、せっかくフレームワークを使ってもMVCに適合した正しい設計がなされず、結果としてMVCを導入するメリットを存分に得られません。

注1) 第1章で解説されているように、MVCは本来Smalltalk言語でGUIプログラムを構築するために生まれた考え方です。PHPにおけるWebアプリケーションの設計・実装が、この本来のMVCに適合するかどうかについては本稿では論じません。

▼図1 MVCの概念図



よくある誤解

MVCの関連を図1に示します。まずはMVCベースのWebアプリケーションを開発するときに生じやすい誤解について見ていきます。

モデルに対する誤解

データベースにアクセスする方法の1つにO/Rマッピングがあります。O/RマッピングのO/RはObject/Relationalの略で、オブジェクト指向で扱うオブジェクトとリレーショナル・データベースで扱うテーブルを対応付ける(マッピングする)しくみです(図2)。O/Rマッピングを使うとデータベースへのアクセスがオブジェクト操作によって実現できるので、現在多くのPHPフレームワークに搭載されています。フレームワークの世界では、「O/Rマッピングを実現するオブジェクト」や「データベースのアクセスを行うオブジェクト」のことをModelと呼ぶことがあります。これが影響しているためか、Modelはまるでデータベース専門のように勘違いされることがあります(図3)。

Modelはアプリケーションの中心

しかし、実際にはMVCにおけるModelは「アプリケーション処理の中心」でビジネスロジックを実装するところです(図4)。つまり、アプリケーション固有の処理を実装し、アプリケーションが実現する業務の中心部分になります。

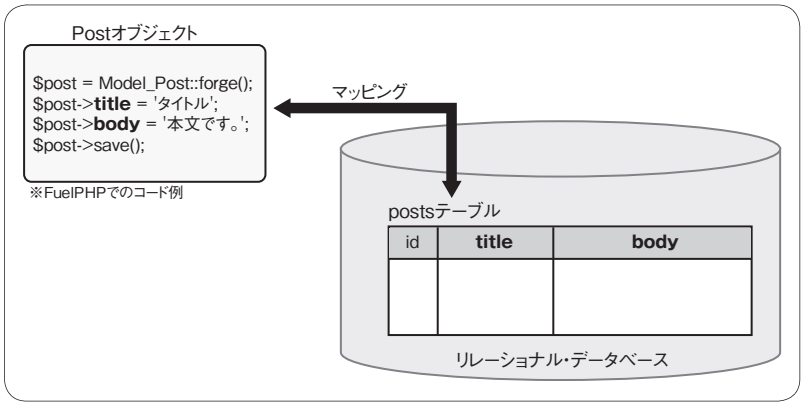
モデルが扱うデータの保存場所はさまざまで、それがたとえばデータベースであろうと、ファイルであろうと、Webサービスのようにリモートホストであろうと構わないのです(**図5**)。

CakePHPのモデルで発生する誤解とは

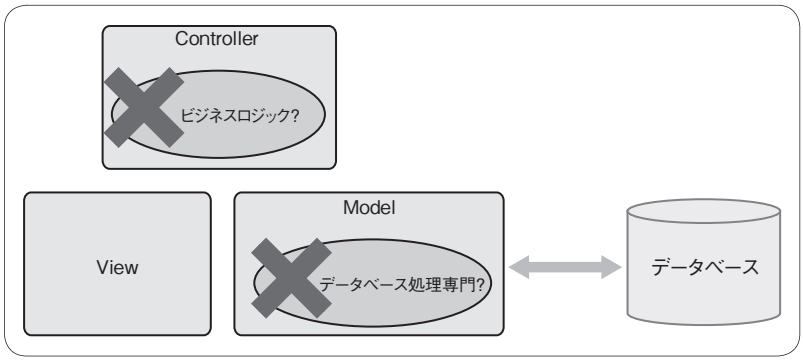
CakePHPのデフォルト設定では、1つのModelが1つのテーブルに紐付けられます[注2](**図6**)。そしてこのモデルオブジェクトを使ってデー

注2) CakePHPのアソシエーション機能やContainableビヘイビアを使用すると1つのモデルから複数のテーブルを扱うなど、Modelとテーブルの関連付けを柔軟に扱えます。またuseTableプロパティの設定により、テーブルを紐付けないModelを作成したり、Model名と異なるテーブルを扱ったりすることも可能です。

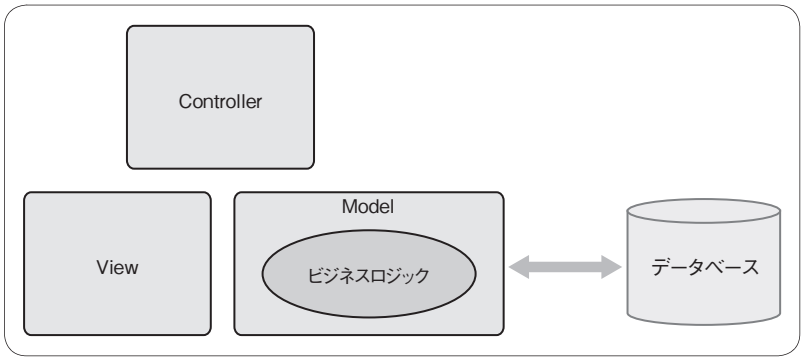
▼図2　O/Rマッピングのイメージ



▼図3　ModelとControllerに対する誤解



▼図4　Modelはアプリケーションの中心



タベースの操作を行うため、CakePHPにおけるModelはまるでデータベース処理だけを行うオブジェクトのように誤解されることがあります。

Webアプリケーションではビジネスロジックを実現するためにデータベースアクセスが必要になることが多いですが、Modelはデータベース処理だけを行うものではありません。

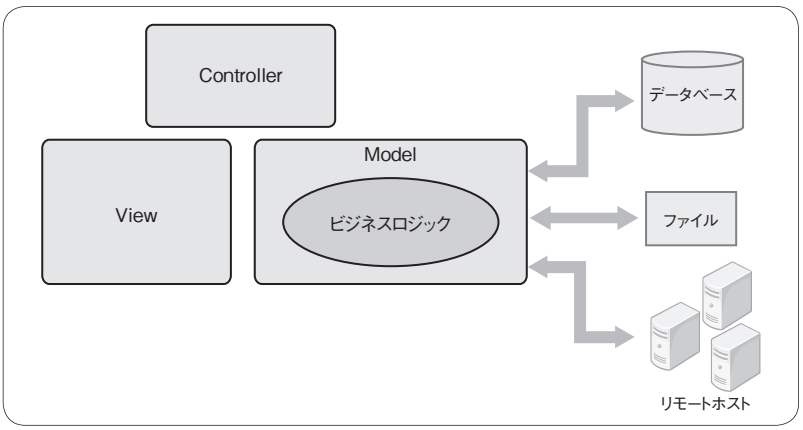
Controllerに対する誤解

Modelへの誤解により「Model＝データベース」のような認識が先行した結果、ビジネスロジックをControllerに実装してしまう事例を見ることがあります(**図3**)。しかし、これは誤りです。ControllerはModelとViewの仲介役なので、フレームワークの機能面で見れば中心にいますが、アプリケーションの中心ではありません。アプリケーション処理の中心はあくまでもModelです。

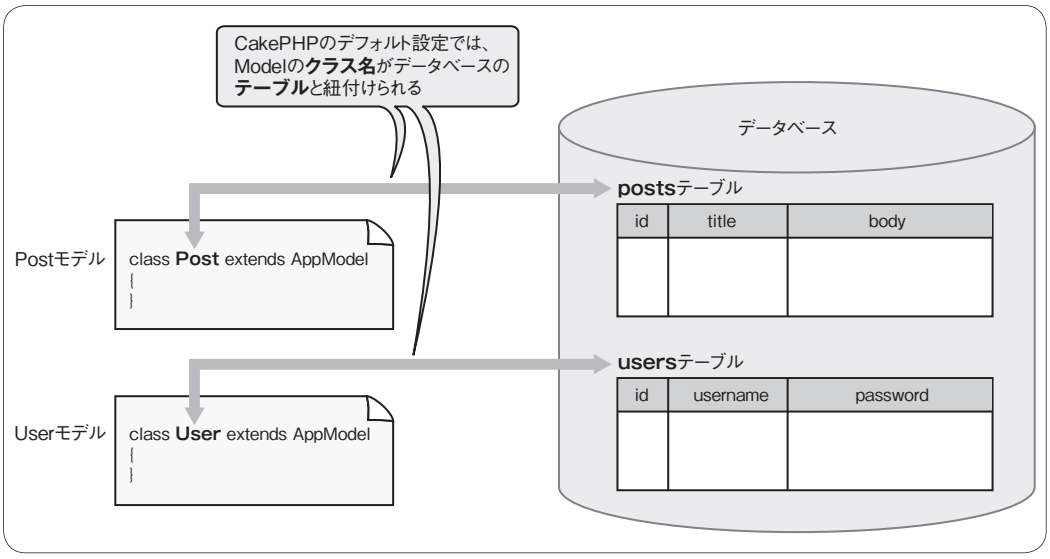
見た目にかかわるロジックを間違える？

MVCに基づく設計では表示部分とビジネスロジック部分は分離されるはずですが、表示にかかわるロジックを大量にModelに詰め込んでいる実装を見かけたことがあります。たとえば

▼図5　Modelが扱うデータの保存場所



▼図6　CakePHPのModelとテーブルの対応



「1234567」を「1,234,567」に変更するような処理は、見た目を整形するために行うものです。このような表示にかかわる処理は通常Viewが担当します。ただし、Viewにはあくまで表示にかかわるロジックだけを記述することに注意が必要です。また、ModelはViewに依存しないように設計するのが基本です。

表示に使うデータはどこが担う?

表示にかかわる処理はViewが担当しますが、Viewが表示するデータはどこから取得するのでしょうか?

CakePHPのMVC関連図(**図7**)を見ると、基本的にViewはControllerからデータを受け取る構造になっています(**図7**の矢印5)。この影響のせいか、「MVCにおけるViewはModelを参照できない」という認識を持っている人もいるかもしれません。しかしMVCにおけるViewは、Modelのデータを参照可能です。ただし、Viewがモデルのデータを変更することはできません(**図8**)。

FuelPHPでは、ビューモデル(ViewModel)というものが存在し、Viewの生成に必要なロジックを記述できます。ビューモデルは、Modelからデータを取得し、表示用の整形処理などを行ったのちViewにデータを引き渡すことができます(**図9**)。

各種フレームワークとMVCの関係

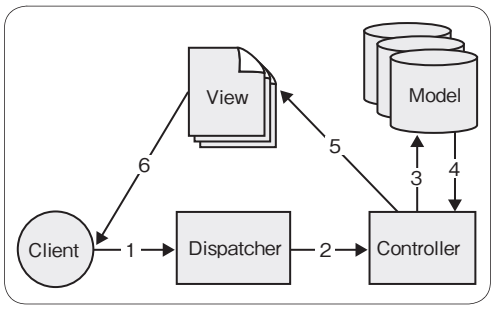
PHPのフレームワークの多くはMVCベースで構成されていますが、それぞれのフレームワークごとにMVCの扱いや構造には特徴や違いが見られます。以降では、CakePHP、FuelPHP、Symfonyの各フレームワークとの関係について見ていきましょう。

CakePHPにおけるMVC

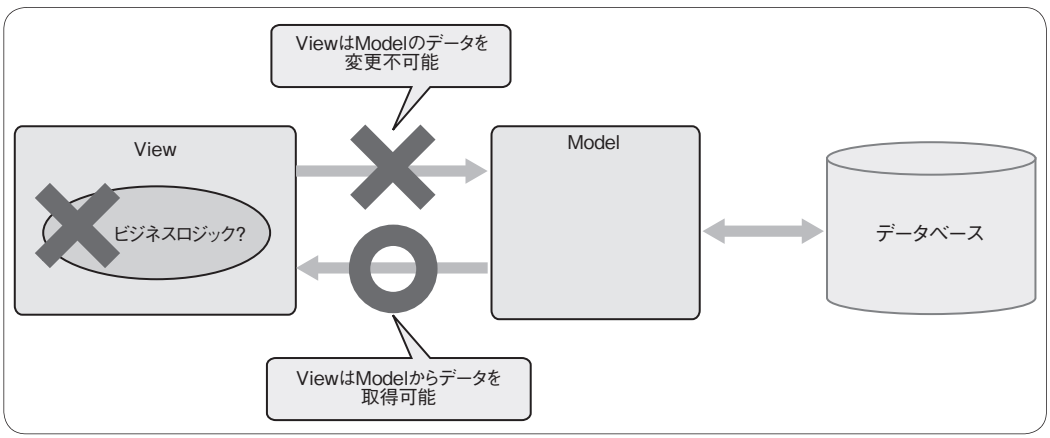
CakePHP[注3]はMVCアーキテクチャを採用しているフレームワークで、CakePHPのドキュメント[注4]にはMVCについて次のような説明があり、MVCの各要素の関連を示す**図7**が掲載

注3) http://cakephp.jp/
注4) http://book.cakephp.org/2.0/ja/

▼図7 CakePHPのMVC(CakePHP Cookbookから抜粋 http://book.cakephp.org/2.0/ja/)



▼図8 ViewはModelのデータを変更できない



されています。

・Model……アプリケーションのビジネスレイヤーを担当する
・View……リクエストに対する出力を生成する役割を担う
・Controller……ModelとViewの仲介者

CakePHPのMVCでは、基本的には1つのModel（CakePHPにおけるModel）に1つのテーブル[注5]が紐付けられます。しかし、Model同士を関連付けることで複数のテーブルを扱ったり、またはテーブルに対応付けないModelを作成したりすることも可能です[注6]。

CakePHPの規約

CakePHPには「設定より規約」（convention over configuration）というコンセプトがあり、プログラムで扱うController/View/Modelのクラス名やファイル名について命名規約が定められています。また、データベースのテーブル名やフィールド名についても命名規約に従うことでCakePHPの機能を最大限に利用できます。

たとえばブログ投稿アプリケーションを作るとして、基本となる名称をPostと決めたとします。この場合Controller/View/Modelの名前は**表1**のようになり、Postモデルには、postsという名前のテーブルが紐付けられます[注7]。

このような規約に従って命名することでさまざまな設定の手間を軽減でき、高速に開発を進められるというメリットがあります。

MVCの記述例

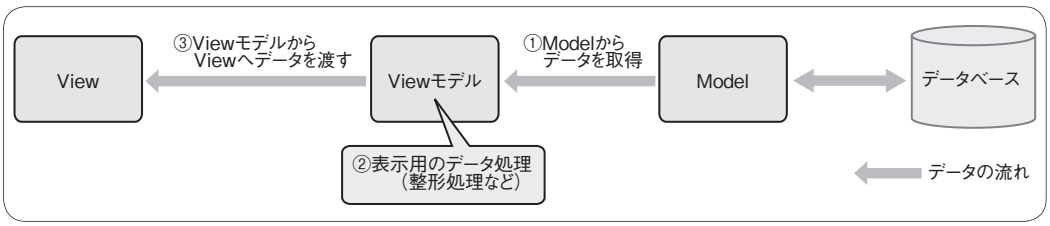
CakePHPにおけるMVCのサンプルコードを示します。**表2**のようなpostsテーブルのデータを一覧表示する簡単なアプリケーションで、実行結果のイメージは**図10**のようになります。

リスト1がController、**リスト2**がModel、**リスト3**がViewです。CakePHPでは、画面全体の構成を「レイアウト」というファイルで定義し、表示する中身を「ビュー」として定義します。**図10**では、画面の上下部分（ヘッダとフッタ）はCakePHPのレイアウト機能によって表示され、Viewで作成した内容は画面の中央部分に表示されています。

たとえばブラウザから**http://設置URL/posts/**のようにURLを指定すると、Postコントローラのindexアクション[注8]（indexメソッ

注5）データベースのテーブル
注6）詳しくはCakePHPマニュアルの「モデルの属性」のページを参照してください（http://book.cakephp.org/2.0/ja/models/model-attributes.html）。
注7）表1に示した以外の命名規約もあります。詳しくは次のWebページを参照してください（http://book.cakephp.org/2.0/ja/getting-started/cakephp-conventions.html）。
注8）アクションとは、ユーザからのリクエストに応じて受け付けを行う機能で、リクエストに対応するメソッドをコントローラ内に定義します。

▼図9　FuelPHPのビューモデルの処理



▼表1　CakePHPの命名規約の一例

	Controller	View	Model
名称	Posts	Post	Post
クラス名	PostsController	—	Post
ファイル名	posts_controller.php	xxxxx.ctp（xxxxxはアクション名）	post.php

※英字の大文字と小文字は区別されます。

ド)がフレームワークから呼び出されます(**リスト1の①**)。indexアクションでは、Postモデルを利用してデータを取得し、setメソッドを使ってViewへデータを引き渡しています。命名規約に従うことにより、対応するView(index.ctp)を使って表示が行われます。

ビューの実装

MVCでは見た目に関する部分をViewとして分離します。Viewの分離にはフレームワークに組み込まれているテンプレートエンジンを使うのが定番となっています。テンプレートエンジンを使うと、HTMLの部分をテンプレートファイルやViewファイルと呼ばれる外部のファイルに記述してプログラム本体とは別に管理できます。CakePHPでは、拡張子.ctpのファイルに、Viewで出力するテンプレートの内容を記述します。**リスト3**では、Postsコントローラからsetメソッドで引き渡されたデータを変数$postsで受け取り、実際に出力しています。

モデルの処理

このアプリケーションは非常に簡単なので、モデル(**リスト2**)には何も処理を記述していません。指定したテーブルからシンプルなクエリで検索結果を取得するような処理は、すでにフレームワークに組み込まれているので、コードを記述しなくても実現できてしまいます。

実際の開発の際には、モデルにはビジネスロジック、バリデーション関連処理(バリデーションルール、バリデーションチェック処理)、ビューに依存しないデータ処理などを実装します。

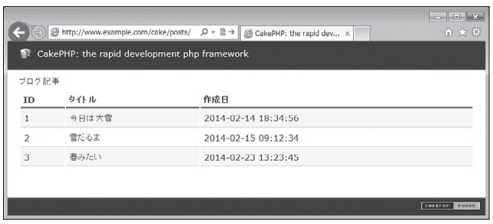
プラグインでMVCアプリケーションを再利用

CakePHPでは、モデル・ビュー・コントローラのセットをまとめて「プラグイン」を作ることができます。一般的に「プラグイン」というと、便利な処理が集合したライブラリのようなイメージを持つかもしれませんが、CakePHPのプラグインはMVCを1つにまとめて再利用できる

▼リスト1　CakePHPにおけるControllerのサンプル

```
<?php
// /app/Controller/PostsController.php
class PostsController extends AppController
{
    public $helpers = array('Html', 'Form', 'Session');

    public function index() {  ←①
        $this->set('posts', $this->Post->find('all'));
    }
}
```

▼図10　CakePHPアプリケーションの実行結果



▼表2　postsテーブルのデータ

id(int)	title(varchar)	created(datetime)
1	今日は大雪	2014-02-14 18:34:56
2	雪だるま	2014-02-15 09:12:34
3	春みたい	2014-02-23 13:23:45

▼リスト2　CakePHPにおけるModelのサンプル

```
<?php
// /app/Model/Post.php
class Post extends AppModel
{
}
```

▼リスト3　CakePHPにおけるViewのサンプル

```
<!-- /app/View/Posts/index.ctp -->
<h1>ブログ記事</h1>
<table>
  <tr><th>ID</th><th>タイトル</th><th>作成日</th></tr>
  <?php foreach ($posts as $post){ ?>
  <tr>
    <td><?php echo h($post['Post']['id']); ?></td>
    <td><?php echo h($post['Post']['title']); ?></td>
    <td><?php echo h($post['Post']['created']); ?></td>
  </tr>
  <?php } ?>
</table>
```

ようにしたものを指します。プラグインを使うと、アプリケーションをまるごと部品にしてほかのアプリケーションから利用できます(**図11**)。

FuelPHPにおけるMVC

FuelPHP[注9]はPHP5.3以降に対応したフレームワークです。MVCアーキテクチャを採用していて、FuelPHP日本語Documentのサイト[注10]ではMVCについて次のような説明があります。

- Model……ある種のデータ表現であり、データを変化させるメソッドを持つ
- View……ブラウザにデータを提供するファイル。ロジックからレイアウトを分離する
- Controller……Modelを使用してデータを取得するメソッドを実行し、ビューを呼び出す

FuelPHPではORMパッケージ[注11]を利用したモデルを作成すると、オブジェクトとデータベースのレコードを対応付けてO/Rマッパとして使用できます。また、CakePHPほど多くはありませんがFuelPHPにもいくつかの命名規則が存在します[注12]。

注9) http://fuelphp.jp/

注10) http://fuelphp.jp/docs/1.7/

注11) FuelPHPにおけるパッケージとは、便利なコードや設定等をまとめて再利用できるようにしたものです。

注12) FuelPHPの命名規則については、次のURLを参照してください(http://fuelphp.jp/docs/1.7/general/coding_standards.html)。

MVCの記述例

FuelPHPにおけるMVCのサンプルコードを示します。先に述べたCakePHPのコード例とほぼ同じですが、**表3**のようなpostsテーブルのデータを一覧表示する簡単なアプリケーションです。実行結果のイメージは**図12**のようになります。

リスト4がコントローラ、**リスト5**がモデル、**リスト6**がビューです。モデルを実装する方法はいくつかあるのですが、ここではFuelPHPに付属のoilコマンド[注13]を使い、ORMパッケージを利用するモデルの例としています。

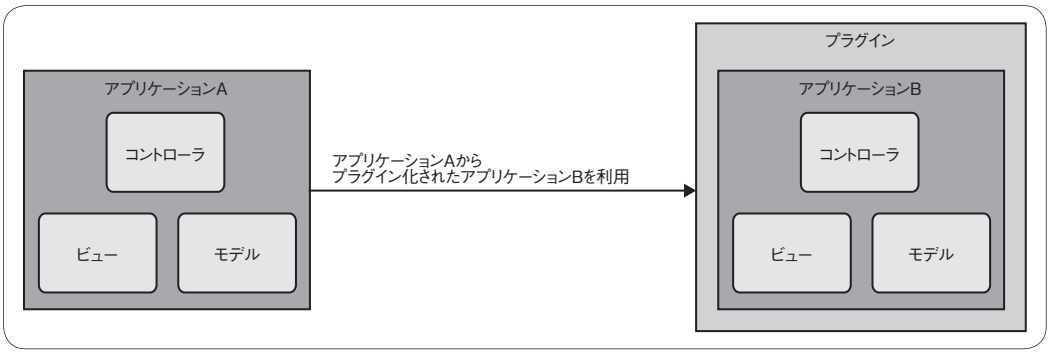
たとえばブラウザから**http://設置URL/post/**のようにURLを指定すると、Postコントローラ(Controller)のaction_indexメソッドがフレームワークから呼び出されます(**リスト4の①**)。action_indexメソッドでは、まずView(post.php)を割り当て、Postモデル(Model_Postクラス)を利用してデータを取得した後、Viewへデータを引き渡しています。

Viewの実装

FuelPHPにおいても、見た目に関する部分はViewファイルに切り出します。拡張子.phpのファイルにViewで出力するテンプレートの内容を記述します。Viewを使うには、コントローラ側で使用するViewの割り当てを行います。リ

注13) oilはFuelPHPが提供するコマンドラインツールで、開発を支援するための多くの機能が含まれています。

▼図11　CakePHPのプラグイン



ストト4ではforgeメソッドを呼び出し、Viewファイルのpost.phpを割り当てています。リスト6では、Postコントローラ(Controller)からsetメソッドで引き渡されたデータを変数$postsで受け取り、実際に出力しています。

ビューモデル

FuelPHPでは、Viewの生成に必要なロジックを記述できるビューモデル(ViewModel)というものが存在し、必要に応じて利用できます(図13)。ビューモデルの処理のイメージは、先に示した図9のようになります。ビューモデルは、モデル(FuelPHPにおけるモデル)の機能を呼び出してデータを取得したり、データを表示用に整形したりする目的で使われます。ビューモデルにはあくまでもView生成に必要なロジックを実装し、ビジネスロジックを含めないようにします。

ORMパッケージを利用したモデル

リスト5のモデル(Model_Post)は、FuelPHPに付属のoilコマンドを使い、ORMパッケージを利用する形式としました。これにより、Modelは O/Rマッピングの機能を持ちます。たとえばテーブルにレコードを追加する場合、リスト7のような記述を行うことで可能です。

▼リスト4　FuelPHPにおけるControllerのサンプル

```
<?php
// /app/classes/controller/post.php
class Controller_Post extends Controller
{
    public function action_index()  ❶
    {
        $view = View::forge('post');
        $view->set('posts', Model_Post::find('all'));
        return $view;
    }
}
```

▼表3　postsテーブルのデータ

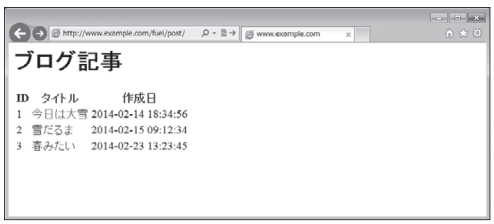
id(int)	title(varchar)	created_at(int)	updated_at(int)
1	今日は大雪	2014-02-14 18:34:56	1392370496
2	雪だるま	2014-02-15 09:12:34	1392423154
3	春みたい	2014-02-23 13:23:45	1393129425

モデルのtitleプロパティがpostsテーブルのtitleカラムに対応しています。

モジュールとHMVCリクエスト

FuelPHPでは、モデル・ビュー・コントローラのセットをまとめて「モジュール」を作ることができます。CakePHPの「プラグイン」と同じような機能で、アプリケーションをまるごと部品にして他のアプリケーションから利用できます(図14)。またFuelPHPでは、HMVCリクエスト(Hierachical MVC request)という機能を使うと、あるControllerから別のモジュールのControllerのメソッドを呼び出せます(図15)。

▼図12　FuelPHPアプリケーションの実行結果



▼リスト5　FuelPHPにおけるModelのサンプル

```
<?php
// /app/classes/model/post.php
class Model_Post extends \Orm\Model
{
    protected static $_properties = array(
        'id',
        'title',
        'created_at',
        'updated_at',
    );

    protected static $_observers = array(
        'Orm\Observer_CreatedAt' => array(
            'events' => array('before_insert'),
            'mysql_timestamp' => false,
        ),
        'Orm\Observer_UpdatedAt' => array(
            'events' => array('before_update'),
            'mysql_timestamp' => false,
        ),
    );
    protected static $_table_name = 'posts';

}
```

SymfonyとMVC

Symfony[注14]はとても高機能なフレームワークです。あらゆる種類のWebアプリケーションを開発可能な高い拡張性を持っています。執筆時点での最新バージョンは、2.4.2です。

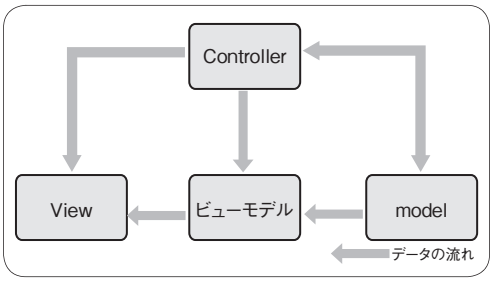
かつてのSymfonyはMVCフレームワーク

以前のバージョンであるSymfony 1.Xのマニュアルのページ[注15]を見ると、MVCベースのフレームワークであると明記されていて、MVCについては**図16**を用いて詳しく説明されていました。

注14) http://symfony.com/

注15) http://symfony.com/legacy/doc/book/1_2/en/02-exploring-symfony-s-code

▼図13　FuelPHPのビューモデルのイメージ



Symfony2はMVCにこだわらない方針

しかし、現在のSymfonyはMVCベースであることを以前ほど強調していません。『Symfony2 is an HTTP framework; it is a Request/Response framework.』とは、Symfonyの開発者の1人であるFabien Potencierさんの言葉です[注16]。SymfonyはMVCにこだわらず、基本原則としてはHTTP（リクエスト／レスポンス）の中心的な責務を果たすことに重きを置いたフレームワークと言えるでしょう[注17]。

開発者に自由度を持たせる構造

Symfonyフレームワークから明示的に提供されるのは、MVCでいうところのコントローラとビューに該当する部分です。ビジネスロジックの部分は、Symfonyの豊富なコンポーネント（Web開発に共通的な機能ライブラリの集合体）を利用したり、サードパーティ製のライブラリを利用したり、もちろんModel部分を自作したりしながら、開発者次第でどのような形にもで

注16) URL http://fabien.potencier.org/article/49/what-is-symfony2

注17) Fabienさんの発言の中には「私はMVCが好きではない」というフレーズが何回か出てきて、MVCにまったくこだわっていないことがうかがえます。

▼リスト6　FuelPHPにおけるViewのサンプル

```
<!-- /app/views/post.php -->
<html>
<head>
  <meta charset="UTF-8">
</html>
<body>
<h1>ブログ記事</h1>
<table>
  <tr><th>ID</th><th>タイトル</th><th>作成日</th></tr>
  <?php foreach ($posts as $post){ ?>
  <tr>
    <td><?php echo $post->id; ?></td>
    <td><?php echo $post->title; ?></td>
    <td><?php echo date('Y-m-d H:i:s', $post->created_at); ?></td>
  </tr>
  <?php } ?>
</table>
</body>
</html>
```

きる、というのがSymfonyフレームワークの方針のようです。

設計・実装のポイント

フレームワークにはそれ自身の明確な設計思想があります。言うまでもないことですが、まずは使用するフレームワークのMVC構造と使い方を正しく理解することが大切です。以降では、MVCベースのフレームワークで設計する上でのポイントをいくつか述べます。具体例についてはCakePHPを用いて説明します。

モデルの処理をまとめる・分割する

一般にバリデーション(データの検証)はモデルが担当します。画面の入力項目が多いとバリデーション関連の記述だけでも大量になります。またアプリケーションの規模が大きい場合は、モデルに含むコードも必然的に膨大になります。1つのモデルが大きくなり過ぎたり、1つのモデルにいろいろな機能を詰め込み過ぎるのは良くありません。そのような場合には、共通的な処理を切り出したり、機能ごとにモデルを分割したりする必要があります。

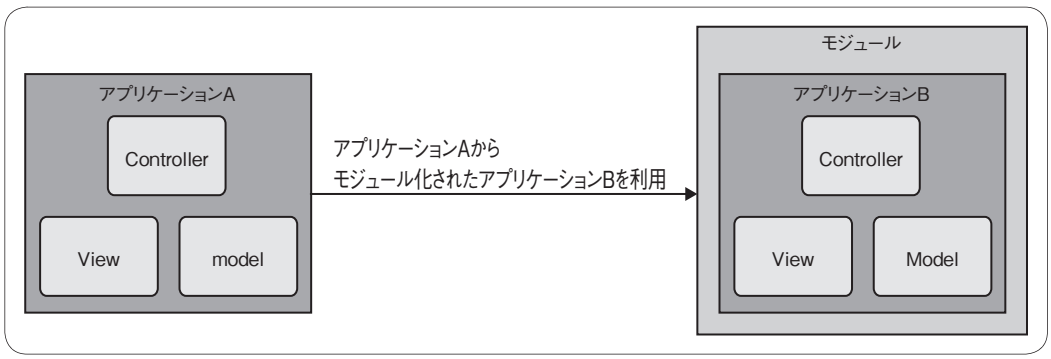
CakePHPでモデル処理を共通化するには

CakePHPで、共通化したモデルを作る方法としては次のようなものがあります。

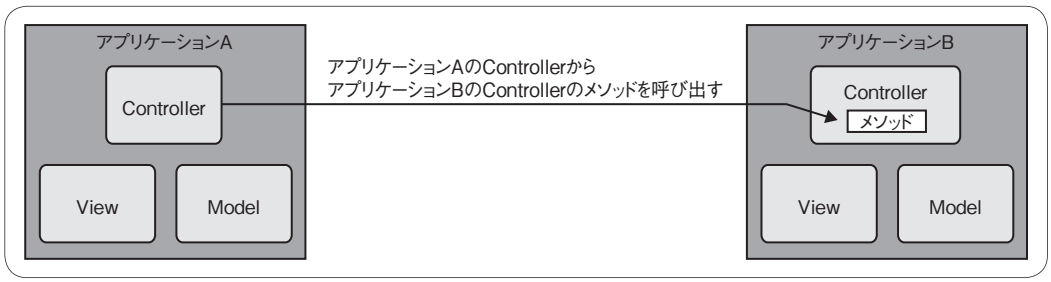
▼リスト7　FuelPHPにおけるORMを利用したサンプル

```
<?php
// /app/classes/controller/post.php
class Controller_Post extends Controller
{

    // 中略

    public function action_add()
    {
        $post = Model_Post::forge();
        $post->title = '今日は寒い';
        $post->save();

        Response::redirect('post');
    }
}
```

▼図14　FuelPHPのモジュール



▼図15　FuelPHPのHMVCリクエストのイメージ



・共通的なモデル処理をまとめて親クラス（AppModel）に実装する（継承による再利用）
・ビヘイビアを作成して利用する（委譲による再利用）
・アクションごとにモデルを作る[注18]

CakePHPの「ビヘイビア」とは、Modelを機能拡張するためのものです。Modelに共通的な処理をまとめてビヘイビアとして作成し、Modelから呼び出して使えます。

Viewに関連するロジック

「Modelはアプリケーションの中心」の節でも説明しましたが、MVCに基づく設計では、見た目にかかわるロジックをModelには含めません。だからと言ってビューテンプレート内に見た目にかかわるロジックをたくさん記述してはメンテナンスが大変です。

このような場合の対策には、CakePHPの場合、ビューを拡張する「ヘルパー」というものを活用できます。表示用のタグ生成や文字列整形などの処理をヘルパーにして、ビューからは部品のように使えます。FuelPHPの場合は、47ページで説明したビューモデルを活用できます。

コントローラに搭載する機能

モデルにはビジネスロジックを搭載し、Viewは表示関連を担当します。それでは、Controllerに処理を搭載するとしたらどのような機能でしょうか？ 「ユーザからのリクエストを判定する処理」や「URLに直接関わる処理」を行う必要がある場合はControllerの出番です。

ログイン認証はMVCが協力して実現

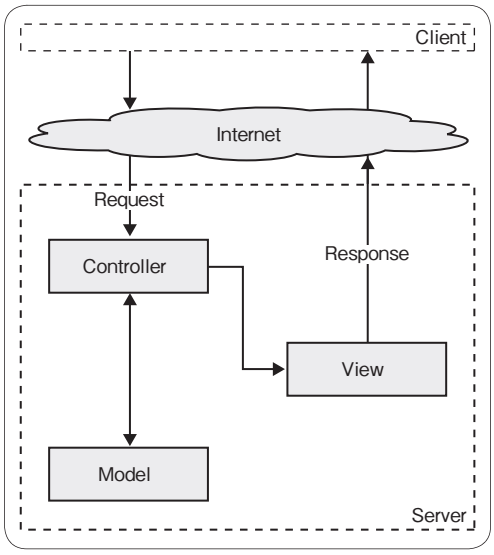
たとえばログイン認証機能は、ユーザがリクエストしてきたURLについて判断する必要があるのでController側に一部の機能を実装します。CakePHPの場合、ControllerにAuthコンポーネント[注19]を組み込んでログイン認証の機能を実現できます。ただしログイン認証機能は、ログイン画面の表示やデータベースアクセスも必要なのでControllerだけで実現するものではありません。またHTTPリクエストのヘッダ情報を判定して機能を振り分けたいという場合は、Controllerで判定とページ遷移の処理を行うことにより実現できます。

おわりに

PHP用のフレームワークのMVCは、各フレームワークによって大きな概念は変わらないものの、それぞれの設計思想に合わせて応用された構造になっています。設計・実装の際は、MVCの役割分担を明確化することが大切で、肥大化する各MVCについてはフレームワーク

注18) CakePHPでは「1モデル＝1テーブル」の原則がありますが、アクションの種類が多いときはそのようなモデルを扱いづらい場合があります。この対策に、アクションごとにモデルを作るという方法があります。詳しくは、章末の参考文献[2]を参考にしてください。

▼図16 symfony 1.XのMVC（symfony 1.Xのドキュメントから抜粋 http://symfony.com/legacy/doc/book/1_2/en/02-exploring-symfony-s-code）



注19) CakePHPにおける「コンポーネント」とはコントーラを機能拡張するしくみのことです。

の機能を使ってまとめたり分割したりできます。

最後に、PHPでMVCベースの正しい設計・実装を行うために役立つ参考書籍を次ページに紹介します。参考文献[1]では、Symfonyフレームワークの豊富な機能を活用しながらMVCに沿ったアプリケーションを開発する方法について学べます。参考文献[2]では、CakePHPフレームワークの基本的な使い方や各MVCを機能拡張させるための実践的な方法を知ることができます。SD

●参考文献

[1]『効率的なWebアプリケーションの作り方～PHPによるモダン開発入門』小川 雄大(著)、技術評論社

[2]『CakePHP2実践入門』安藤祐介、岸田健一郎、新原雅司、市川快、渡辺一宏、鈴木則夫(著)、技術評論社

COLUMN

「CakePHP 3.0の動向」

CakePHPの現在のバージョンは2.4ですが、次のメジャーアップデート版となるCakePHP 3.0の開発も着々と進んでいるようです。2014年1月には、CakePHP 3.0.0開発プレビュー版が公開されました。開発プレビュー版とは、機能やAPIに不完全な部分はあるものの開発者向けに試してもらおうと一式を公開するものです。正式版リリースに向けて、世界中の開発者からの多くの有益なフィードバックを得ることを目的としています。

CakePHP 3.0.0開発プレビュー版でアナウンスされている内容[注20]とCakePHP 3.0への移行ガイド[注21]をもとに、CakePHP 3.0の特徴についていくつか紹介します。CakePHP 3.0を動作させるためのシステム要件は次のようになります。

- PHP 5.4.3以降に対応
- mbstring拡張モジュール
- mcrypt拡張モジュール

PHP 5.4.3以降に対応し、CakePHPのすべてのコアクラスは名前空間に属するようになります。

CakePHP 3.0では多くの改善点がありますが、中でもとくに注目すべきはモデル周りの大幅な変更です。CakePHP 3.0のモデルには刷新されたORM(Object Rerational Mapping)の機構が導入され、データベースアクセスに関連する多くのクラスが追加されます。個人的な意見ですが、これにより本来のモデルの仕事(ビジネスロジックの実装)とデータベースアクセスの仕事の役割分担を設計に反映しやすいように感じられます。

現在のCakePHPでは、モデルを使ってデータを取得するとデータは単なる配列で返るのですが、プログラム処理の状況によっては適合しづらい面がありました。CakePHP 3.0では、モデルが返すデータはオブジェクトになります。レコード単位のEntityオブジェクトとして保持でき、データの扱いやすさが格段に向上します。またクエリの作成についても配列で行っていたものがQueryオブジェクトを使う方式になり、SQLでいうUNIONやサブクエリ(副問い合わせ)などを使う複雑な検索も以前より簡単にできるようになるとのことです。モデル／ORMについては抜本的な構造改革がなされたと言ってもいいほどで、以前と比べて使い勝手がとても良くなることを期待できます。

CakePHP 3.0では、ほかにも改良点がいろいろと挙げられています。

- Composer(パッケージ管理ツール)によるインストールを推奨
- PSR[注22]への対応の促進
- 設定のさらなる簡便化
- ルーティング機能の向上
- パフォーマンスの向上

CakePHP 3.0の正式版リリースにはまだ数ヶ月はかかると予想されますが、どのような美味しいケーキができあがってくるのか楽しみに待ちたいと思います。

注20) http://bakery.cakephp.org/articles/markstory/2014/01/05/cakephp_3_0_0_dev_preview_1_released
注21) http://book.cakephp.org/3.0/en/appendices/3-0-migration-guide.html
注22) PSRとは、PHP-FIG(PHP Framework Interop Group)が策定したPHPの標準コーディング規約です。

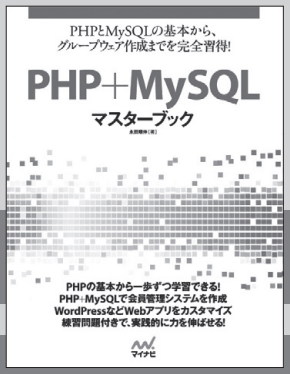
BOOK no.1 実践 Vagrant

Mitchell Hashimoto 【著】／Sky㈱ 玉川 竜司 【訳】
A5判、 248ページ／価格＝2,600円＋税／発行＝オライリー・ジャパン
ISBN＝978-4-87311-665-5

Vagrantは、仮想マシン環境を自動的に構築するためのツールである。同じ仮想マシン環境を作れるために、複数人で開発する場合などに役立つ。本書は、Vagrantの作者のMitchell Hashimoto氏が、Vagrantの概要からWindows、Mac OS X、Linuxの各環境から仮想マシンを構築する方法、プロビジョニング、ネットワーク、複数マシンのクラスタのモデリング、ボックス、プラグインによる拡張などを解説している。現在はクラウド（Amazon EC2など）にも対応しているようだが、まずはVirtualBoxで実際に試してみるのがいいだろう。

BOOK no.2 PHP+MySQLマスターブック

永田 順伸 【著】
B5変形判、 384ページ／価格＝2,700円＋税／発行＝マイナビ
ISBN＝978-4-8399-4759-0



1冊でPHPとMySQLを使ったWebアプリ開発が体験できる本。本書の3分の2はPHPの文法やMySQLの基本操作、PHPでMySQLを扱う方法を解説し、残り3分の1で簡易的な会員管理システム開発を実習する構成となっている。1冊とはいえ、かなり細かい情報が網羅的に盛り込まれている。会員管理システムの実習では、アクセス制限、セッション管理、認証、タイムアウト処理、APIによる他Webサービスとの連携など、基本的な機能をざっと実践でき、Webアプリ開発のエッセンスを学ぶのに良さそうだ。サンプルコードや実行結果の図が多く、XAMPPのおかげで開発環境も簡単に用意できるので、ぜひ本書のとおりに手を動かしてみてほしい。

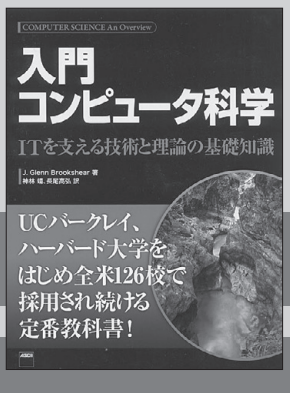
BOOK no.3 絶対に挫折しない iPhoneアプリ開発「超」入門 【iOS7対応】増補改訂版

高橋 京介 【著】
B5変形判、 416ページ／価格＝2,800円＋税／発行＝SBクリエイティブ
ISBN＝978-4-7973-7545-9

iOS 7に対応し、加筆修正された人気の入門書。フルカラーで掲載されたXcodeの画面を見ながらの手順説明はわかりやすく、たどっていくだけでサンプルアプリが完成する。ここでの挫折はないだろう。実装内容の説明はアプリを完成させたあと。まずは完成する喜びを味わうのが本書のポイントだ。アプリ開発者になるためには実装内容の理解が必要だが、本書だけでそれが可能なほどプログラミングはあまくない。それでも、初めてプログラミングをする人にとっては用語や構造、オブジェクト指向まで平易にまとめられているので、この後に手にすることになるであろうObjective-Cの本を読むための手助けに充分なる。

BOOK no.4 入門 コンピュータ科学 ITを支える技術と理論の基礎知識

J.Glenn Brookshear 【著】／神林 靖、 長尾 高弘 【訳】
B5変形判、 608ページ／価格＝3,800円＋税／発行＝KADOKAWA
ISBN＝978-4-04-886957-7



コンピュータ科学の基礎が身についていないと、高い品質のシステムを作ることはできない。たとえば、浮動小数点数の丸め誤差について知らなければ、数値の正確性を求められるシステムは作れないだろうし、データ構造の知識がなければ、バッファオーバーフローの脆弱性を持ったプログラムを作ってしまう恐れもある。本書はそんな基礎知識を自習できる学習書。データの格納と操作のしくみ、OS、ネットワークとインターネット、アルゴリズム、プログラミング言語、DB、ソフトウェア工学など伝統的な内容から人工知能やCGなどの最新技術まで内容は幅広い。分量も608ページと多いが、その分、通読できたなら確かな自信につながるだろう。

第2特集

ネットワークエンジニア養成

ロードバランサの教科書

本特集ではネットワークの負荷分散を行うロードバランサの基礎を、ハードウェア製品、ソフトウェアでの実現、クラウド上での実装の3つに分けて解説します。

Chapter1ではロードバランサの基本動作から、サーバの負荷分散の方法やシステムの冗長化について解説し、実際の製品レベルでの動向を紹介します。

Chapter2では、オープンソースを用いたソフトウェアとしてのロードバランサの実例と、Windows ServerでのNLBについて紹介します。

Chapter3では、クラウド上で提供されるロードバランササービスを紹介し、実際の使い勝手や、どのように実装されているかについても紹介しています。

CHAPTER 1

基本機能と最新機能を整理 ロードバランサからADCへ

文：鶴長 鎮一（つるなが しんいち）
shinichi.tsurunaga@gmail.com

協力：F5ネットワークスジャパン㈱　ギド フォスメア

Webサーバの普及でロードバランサの需要が高まっています。ロードバランサを導入することで、サーバの負荷分散やシステムの冗長化が可能です。本稿では、F5ネットワークスジャパン社の協力のもとに、前半でロードバランサの基本動作について解説し、後半で最近の動向を解説します。

ロードバランサの機能

負荷分散のためのロードバランサ

サーバにかかる負荷を減らし、大量のリクエストを捌くには、サーバのCPUやメモリを増強するか、サーバの台数を増やす必要があります。サーバを増強する方法は「スケールアップ（垂直スケール）」と呼ばれ、よりハイスペックなサーバに置き換えることで処理能力を向上しますが、ハイスペックなサーバは費用が高く、値段と性能が比例しません。そのため、2倍の費用をかければ、性能も2倍になるといった予測をたてることができません。またサーバが停止すればシステム全体がダウンする危険性があり、アベイラビリティの面で不安が残ります。サーバの台数を増やす方法は「スケールアウト（水平スケール）」と呼ばれ、アクセス数の増加に応じてサーバの台数を増やすことで大量のリクエストに対応します。サーバ台数を2倍に増やせば、性能も比例して向上するため費用対効果が高く、スケーラビリティに優れています。また複数のサーバで処理を行うため、そのうちの1台が停止してもほかのサーバで処理を継続できるなど、アベイラビリティも優れています。スケーラビリティやアベイラビリティで有利なスケールアウトですが、クライアントからのリクエストを複数のサーバに分散させるのに、特別なしくみが必要になります。それを可能にしているのが「ロードバランサ（負荷分散装置）」です。

ロードバランサにはサーバにインストールして利用するものから、米F5 Networks社の「BIG-IPシリーズ（**写真1**）」に代表されるネットワークアプライアンスまでさまざまなものがあります。専用のネットワークアプライアンスは、サーバとインターネットの間に設置でき、サーバの構成を大きく変えたり、特殊なチューニングを施したりする必要がないため導入が簡単です（**図1**）。

ロードバランサ以外の負荷分散方法と課題

ロードバランサを使用せずに、負荷を分散させる方法もあります。たとえばクライアント側で、www1、www2のようにホスト名を変えれば、リクエストを複数のサーバに分散できます。ただし、どのサーバにアクセスするかは、クライアントしだいとなるため、アクセス数やトラフィック量に応じてサーバを割り振ることはで

▼写真1　F5 Networks社のBIG-IPシリーズ

きません。また指定したサーバに障害が発生しても、ほかのサーバに処理を引き継ぐこともできません。何よりクライアント側でホスト名を変えるのは利便性が良くありません。

「DNSラウンドロビン」と呼ばれる方法なら、クライアントでホスト名を変えなくても、アクセスするサーバをその都度変えられます。通常DNSサーバのゾーンレコードには、1つのホスト名に対し1つのIPアドレスを記述しますが、1つのホスト名に対し複数のIPアドレスを記述することで、DNS問い合わせのたびに違うアドレスを返すようにでき、結果的にアクセスするサーバを分散させられます。ただしDNSラウンドロビンでは、処理能力が高いサーバへのアクセスは高頻度に、処理能力が低いサーバは低頻度にといった、サーバの処理能力に合わせてアクセス頻度を変えることはできません[注1]。また障害が発生したサーバへのアクセスを防止することもできません。さらにDNSの変更を行っても、キャッシュによりタイムラグが生じてしまうため、即座に構成変更を反映できません。

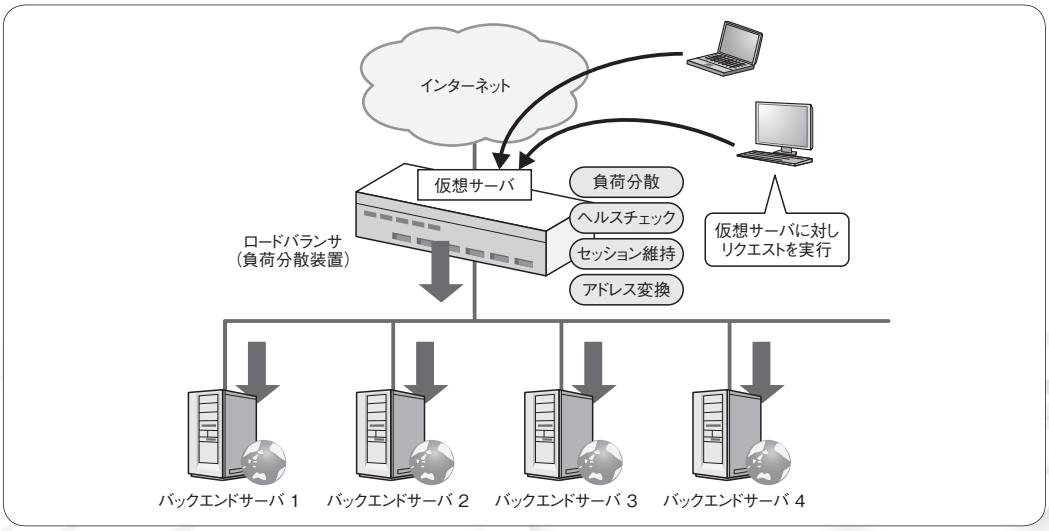
ラウンドロビン／コンテンツスイッチング／動的振り分け

DNSラウンドロビンを使った負荷分散は、費用をかけずに利用できるものの、耐障害性や分散方式に問題があり、24時間365日連続稼働を要求されるシステムには不向きです。よりミッションクリティカルなシステムには、さまざまな分散方式を備えたロードバランサを使用します。分散方式には主なものだけでも**表1**のようなものがあります。順番にサーバを割り振る「ラウンドロビン方式」には、単純に設定された順に割り振り同じ割合でサーバを使用する方式のほかに、優先順位に従って割り振る「優先順位方式」や、あらかじめ決められた割合で割り振るこ

▼表1　負荷分散の方式

方式名	負荷分散の方法
ラウンドロビン方式	バックエンドサーバを順番に使用
優先順位方式	設定した優先順位に従う
重み付け方式	設定した割合に従う
コンテンツスイッチング	HTTPヘッダやURL、ペイロードによってバックエンドサーバを決定
最速応答時間方式	応答が最も早いバックエンドサーバを優先
最少コネクション方式	接続しているコネクション数が最少のサーバを優先
最少トラフィック方式	トラフィック量が最も少ないサーバを優先

▼図1　ロードバランサの役割



注1）SRVレコードを用いればDNSだけで冗長化と負荷分散を実現できますが、SRVレコードに対応したクライアントは多くありません。

とができる「重み付け方式」があります。

優先順位方式では、各サーバに「優先順位(Priority)」を設定し、優先順位の高いサーバからリクエストを割り振り、それが一定以上超えた場合に、次に優先順位が高いサーバに割り振ります。優先順位が低いサーバを普段はほかの用途に使用しピーク時にだけ利用したり、優先度が最も低いサーバに「アクセス集中のため表示できません」といったコンテンツを用意しておき、アクセスが混雑していることをユーザに告知したりできます。

重み付け方式では、割り当てるサーバの頻度を「重み(Ratio)」で変えることができます。分散させるサーバの性能に差がある場合、低スペックなサーバへの割り振りを減らし、高スペックなサーバへの割り振りを増やすといったことが可能になります。たとえば高スペックなサーバに対する重みを「3」に、低スペックなサーバに対する重みを「1」に設定することで、3:1の割合で高スペックなサーバへの割り振りを増やせます。

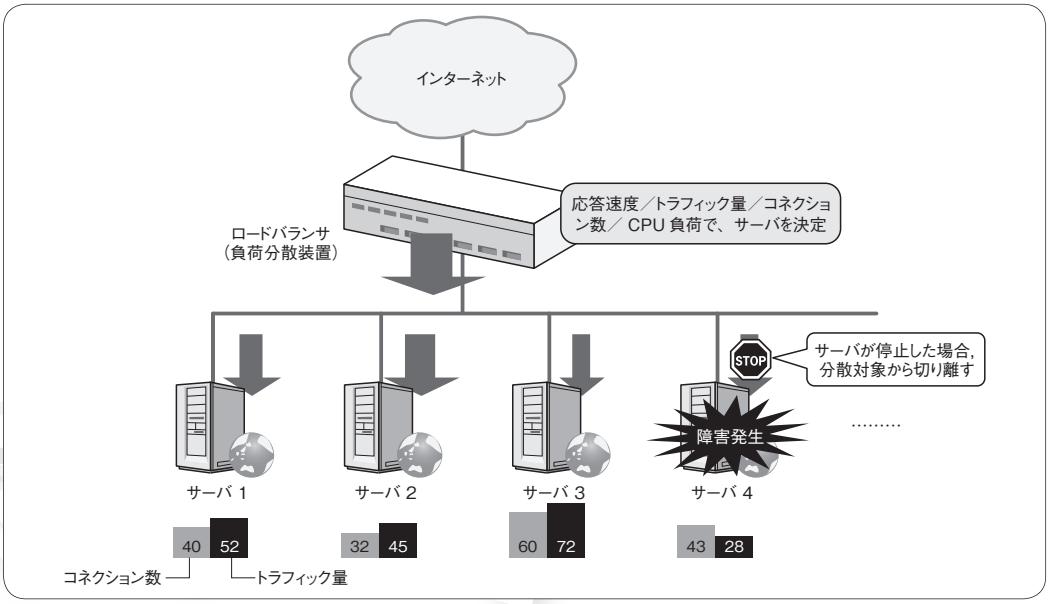
こうしたTCPレベル(L4[注2])の負荷分散に加え、HTTPヘッダやURLといったアプリケーションレベル(L7[注3])で割り振るサーバを変更できる「コンテンツスイッチング方式」もあります。URLに「img」が含まれていたらサーバ1に、拡張子が「.php」だったらサーバ2へといった割り振りが可能です。画像やHTMLといった静的コンテンツと、PHPやJavaといった動的なコンテンツを分けることができます。またキャッシング機能と組み合わせれば、さらにレスポンスを高めることも可能です。

ラウンドロビンやコンテンツスイッチングは、あらかじめ設定したとおりに動作する「静的」なものですが、状況に応じて「動的」に動作する方式もあります(**図2**)。応答速度、トラフィック量、コネクション数、CPU負荷といったデータを、この後解説する「ヘルスチェック」でモニタリングし、割り振るサーバをロードバランサがその都度決定します。「最速応答時間方式」では応答が最も早いバックエンドサーバを優先し、「最少コネクション方式」や「最少トラフィック方式」では、接続中のコネクション数やトラフィック量が最も少ないサーバを優先し割り振ります。

注2) OSI参照モデルのトランスポート層(第4層)。

注3) OSI参照モデルのアプリケーション層(第7層)。

▼図2 動的な負荷分散方式



動的な負荷分散では、各サーバのモニターデータをロードバランサ内部に保持する必要があるため、ロードバランサの負担が大きくなります。高レスポンスを実現するには、より高性能なロードバランサが必要になります。

ロードバランサによっては複数の方式を組み合わせることもできます。たとえば最速応答時間方式と最少コネクション方式を組み合わせ、サーバの負荷状況をより反映させたり、ラウンドロビン方式と優先順位方式を組み合わせ、ある一定量まではプールされたサーバをラウンドロビンで分散し、閾値を越えたときだけ、オフロード用のサーバを使用したりします。

ヘルスチェック

ロードバランサがサーバ障害を検知し、分散対象から切り離すのに「ヘルスチェック」が使われます。ヘルスチェックにはPINGチェック(L3[注4]チェック)、TCPチェック(L4チェック)、アプリケーションチェック(L7チェック)が用いられます。

PINGチェックでは、サーバにPINGパケットを送信し応答の有無で死活を判断します。TCPチェックではサーバのサービスポート(WebサービスならTCP 80番)に対して接続確認を行います。PINGチェックではネットワークの到達性しか監視できませんが、TCPチェックならサービスの接続性まで監視できます。それでもアプリケーションが正常に動作しているかまでは監視できません。たとえばWebサーバとしてサービスは起動していても、コンテンツを正しく配信しているかはPINGやTCPでは確認できません。アプリケーションレベルまで監視するには、アプリケーションチェックが必要になります。一般的に利用されるのが、ヘルスチェック用のダミーコンテンツを使った方法です。各サーバにデータ量の小さい監視専用のコンテンツを用意し、ロードバランサが一定間隔で正常性を確認します。レスポンスコードや応答速度を調べることもできるため、サーバの状態をより明確に検知できます。またWebシステムのように、DBサーバやアプリケーションサーバといった複数のサーバが連携するシステムでは、各システムの状態をヘルスチェックページに動的に埋め込むことで、システム全体としての正常性まで確認できます。

PINGチェックやTCPチェックは、サーバ側に特別な準備はいりませんが、アプリケーションチェックでは、サーバ側の作りこみが必要になります。またアプリケーションのログに、ヘルスチェックのログが出力されてしまい、ポーリング間隔が短いと大量のログが発生することになります。

ヘルスチェックでサーバ停止を判定する際、ダウン検出回数を適切に設定しないと、検知時間が長くなり、障害中のサーバにリクエストを割り振ってしまう危険性が大きくなります。サーバ停止の検出時間は「ヘルスチェックのポーリング間隔×検出回数」で決まります。ポーリング間隔を長くするとサーバ停止を検出する時間が長くなり、ポーリング間隔が短いとサーバ負荷が高くなりレスポンスが遅くなったときに、停止と誤認してしまいます。

TCPポートを使ったサービスのヘルスチェックは容易ですが、UDPポートを使ったサービスだと信頼性を保証できないため、設定が難しくなります。

セッション維持

HTTPのようにコネクションレスなプロトコルだと、アクセスするたびに新たなセッションとして扱われます。たとえばログインページでユーザ認証に成功したとしても、クライアントが次の画面に遷移すれば、違うセッションとして扱われ、認証に成功した情報を引き継ぐことができません。そのためWebシステムではセッション情報をサーバに保持し、同一クライアントからのリクエストならセッション情報を引き

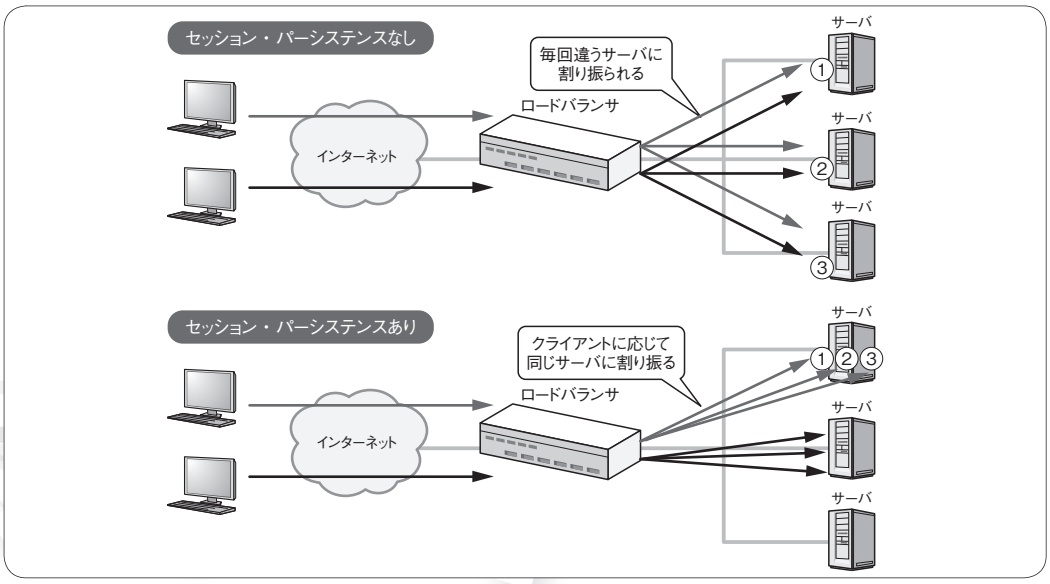
注4) OSI参照モデルのネットワーク層(第3層)。

継ぐようにしています。ロードバランサでリクエストを振り分ける際、最初に振り分けたサーバと別のサーバに振り分けてしまうと、最初に作成されたセッション情報が利用できません。Webサーバ側でセッション情報をほかのサーバと共有する方法[注5]もありますが、そうした手法は複雑です。こうした問題をロードバランサ側で解決するには、同一クライアントからのリクエストを継続して同じサーバに振り分けるようにします。それが「セッション維持(セッション・パーシステンス)」です(**図3**)。

リクエストが同一クライアントのものかどうか判断するのに、クライアントのソースIPアドレスを使う「ソースアドレス・アフィニティ・パーシステンス方式」と、HTTPヘッダのCookie情報に識別用IDを埋め込む「Cookieパーシステンス方式」があります。クライアントがプロキシサーバを使っていたりNAT環境下にあると、ソースIPが変更になる可能性があるため、クライアントをソースIPアドレスで識別するのが困難になります。ただしロードバランサは、ソースIPアドレスと割り振ったサーバとのマッピング情報を管理するだけで済むため、負担が軽くなります。Cookieパーシステンス方式はクライアントの識別がより正確に行えますが、ロードバランサ側でCookie情報にサーバIDを埋め込むため、より負担が重くなります。またアプリケーションデータが暗号化されたHTTPSの場合、Cookieパーシステンス方式は利用できません。HTTPSのセッション維持には、ソースIPアドレスかSSL Session-IDを利用します。ただし、BIG-IPのようにロードバランサでHTTPSを終端させられる場合は、データの中身を見られますのでCookieパーシステンスも利用できます。

注5) Apache Tomcatのようなアプリケーションサーバでは、メモリ、ディスク、RDBMSにセッション情報を保存するセッションレプリケーションが提供されている。

▼図3　セッション・パーシステンス



最新ロードバランサの動向

ADCへの進化

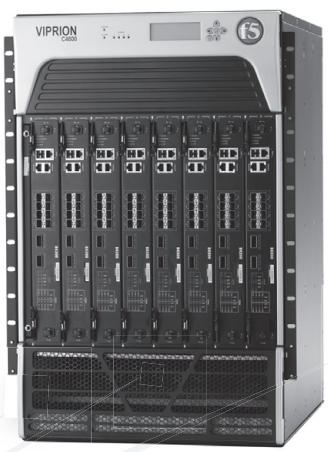
ECサイトやオンラインバンキングといったより高い信頼性が求められるWebサービスの普及とともに、ロードバランサが広く使われるようになりましたが、最近ではソーシャルゲームやモバイルアプリといったリッチコンテンツが

急拡大し、ロードバランサにより多くの機能が求められるようになっています。またユーザエクスペリエンスの重要性が叫ばれるようになり、快適にコンテンツを配信することも一段と重要になっています。単純なトラフィックの分散に加え、ネットワークやアプリケーションの最適化、コンテンツの圧縮、DDoSやDoSといった攻撃に対する防御、アプリケーションレベルでのファイアウォール、SSLオフロード、地理的に離れたデータセンタ間の広域負荷分散といったものが求められています。こうした要望に応えられるようアプリケーションレイヤ(L7)での機能を中心に拡充したのが「ADC(Application Delivery Controller)」です。最新ロードバランサ／ADCの動向をBIG-IPを例に解説します。

BIG-IP

ミッションクリティカルなサービスを支え、ロードバランサの進化を牽引してきたのがF5 Networks社(以降、F5)の「BIG-IP」プラットフォームです。ロードバランサ市場で大きなシェアを誇り、ADCの分野でもほかを大きくリードしています。通信キャリアで使用されるハイエンドなシャーシ・ブレードタイプの「VIPRION 4800(**写真2**)」から、日本市場に特化したローエンドモデルの「BIG-IP 800」まで、さらにXenやVMware ESXiのようなハイパーバイザにインストールできる仮想アプライアンスの「Virtual Edition(VE)」といったものなど、多種多様な製品がラインナップされています。これらの製品は、扱えるコネクション数やトラフィック量といったパフォーマンスやキャパシティが異なるのみで、機能面では共通化が図られています。それを実現しているのが基盤OSの「TMOS(Traffic Management Operating System)」です。F5の全BIG-IPプラットフォームにはTMOSがインストールされており、各機能はTMOS上で動作するモジュールとして提供されています。たとえば負荷分散は「LTM(Local Traffic Manager)」モジュール、Webアプリケーションファイアウォール(WAF)は「ASM(Application Security Manager)」モジュールといったものが用意されています。これらのモジュールは、専用のアクセラレーションハードウェアとともにインストールされた状態で出荷され、実際にモジュールを使用する段階で、ライセンス情報を投入し制限を解除します(**図4**)[注6]。たとえば「BIG-IP LTM」として購入したものに、「ASM」のライセンスを購入することでWAF機能が利用できるようになります。

▼写真2　F5 Networks社のVIPRION 4800



BIG-IPプラットフォームに明確にADCをうたったモデルはありません。それはモジュールを組み合わせることでADC相当として機能するためです。F5では、Fast(早い)／Secure(安全な)／Availability(可用性)を実現したものがADCと位置づけています。

高速配信

年々肥大化するコンテンツをより高速に遅延なく配信し、ユーザエクスペリエンスを高めることがロードバランサの重要な課題となっています。リクエストの結果が返ってくるまでにかかる遅延時間を減らすよう、サーバ側で行う処理をロードバランサ側で高速に処理したり、ネットワークを最適化しトラフィックを効率化した

注6) BIG-IP 800はモジュール追加に非対応。

りとさまざまな手法がとられています。

SSLオフロードは従来から使われてきた高速化手法です。コンテンツの暗号化やキーの作成といったHTTPSで行われる処理を、サーバに代わってロードバランサが行います。サーバとロードバランサとは通常のHTTPで通信するため、サーバの負担を減らすことができます。またSSLサーバ証明書のインストールはロードバランサ側だけで済みます[注7]。最近は鍵長が2048ビットに拡張したSSLが一般的になっており、よりSSL処理が重くなっています。BIG-IPのようにSSLオフロード専用のハードウェアを搭載した製品なら、ほかの処理に影響を及ぼすことなく高速処理が可能です。IPv6対応でも、サーバ側はIPv4のままにし、ロードバランサ側でIPv6に対応することでクライアントとIPv6通信ができるようになります。

トラフィックの効率化はネットワークレベルとアプリケーションレベルの両面で行われています。ネットワークレベルでは、TCPに関わるパラメータを調整し遅延処理やコネクション制御を最適化します。最近は端末やネットワークが多様化し、最適な値もそれぞれの状況に応じて変わってきます。BIG-IPには端末やネットワーク環境ごとに最適化されたプロファイルが用意されており、チューニングやコンフィギュレーションにかかる手間を削減し設定を瞬時に行えるようにしています。また無駄なトラフィックを削減するよう、キャッシュ機能を搭載し、同一のリクエストに対しサーバに代わってロードバランサが代理応答できるようにしています。

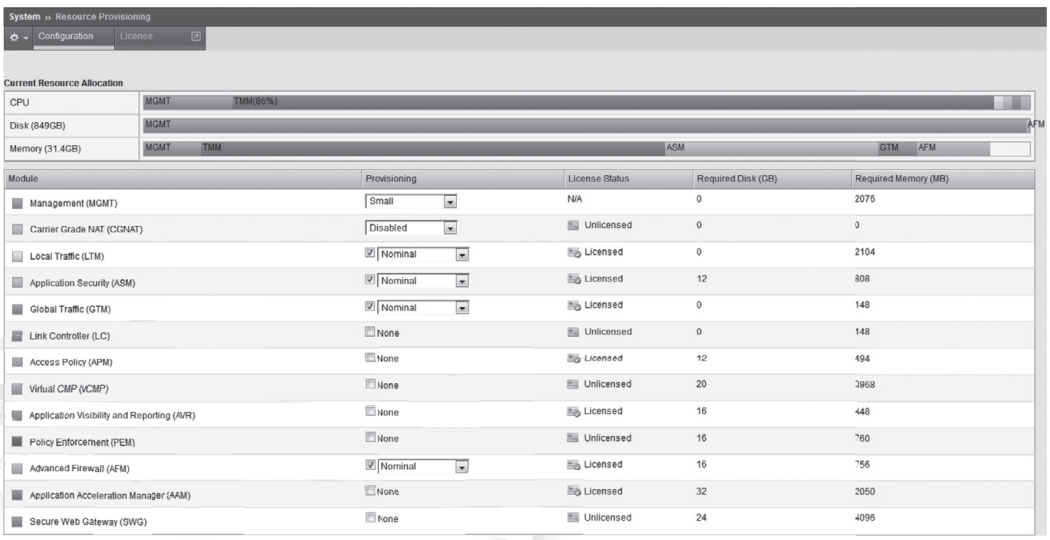
アプリケーションレベルの効率化では、HTTP圧縮機能でコンテンツを圧縮転送したり、プロトコルの優先度で帯域を確保したりできるようにしています。BIG-IPでは、圧縮効果の高いファイルのみを圧縮したり、遅延の大きい回線に対するレスポンスのみを動的に圧縮したりとよりインテリジェンスに機能するようになっています。また圧縮をハードウェアで処理させ、レスポンス遅延を最小限にしています。

セキュリティ機能

サーバやアプリケーションの数が多くなると、セキュリティ管理にかかるコストが大きくなります。ロードバランサはゲートウェイに設置されるため、セキュリティ管理を一括して行うことができ、人員や時間を削減できます。DoS/

注7）SSLサーバ証明書は全サーバにインストールする必要はありませんが、台数分購入する必要があります。証明書発行機関によっては、割安なメニューが用意されている場合もあります。

▼図4　Web UIでライセンスを投入しモジュールを使用可能に



DDoSのようなL3やL4レベルの攻撃から、SQLインジェクションのようなL7レベルの攻撃までマルチレイヤで対応するため、ネットワークFW、IDS/IDF、WAFを別々に設置する必要がありません(**図5**)。攻撃防御には、正しいと定義されたトラフィックのみアクセスを許可する「ポジティブ・セキュリティ」と、脅威を検知するためのデータベース(シグネチャ)を使った「ネガティブ・セキュリティ」を同時に設定でき、既知や未知の攻撃を防御します。BIG-IPは手間のかかるDDoS攻撃元の特定や設定の自動化が可能です。またDoS/DDoSをハードウェアで処理するためパフォーマンスが劣化しません。

WAF機能では、ポジティブ／ネガティブ・セキュリティに加え、SQLインジェクション攻撃のような高度な処理、情報マスキングによる情報漏洩対策が可能です。Webアプリケーションをセキュアにプログラミングしたり、最新パッチを全サーバに適用したりするには、コストや時間がかかりますが、ロードバランサで対策することでリスクを低減できます。BIG-IPはセキュリティ基準のPCI DSS要件6.6に対応し、専用エンジンとハードウェアで、ほかの通信に影響することなく高速に処理します。またIPアドレスをもとに位置を特定する「ジオロケーション」を搭載しているため、特定地域や国単位でアクセスを拒否することが可能です。

拠点間冗長

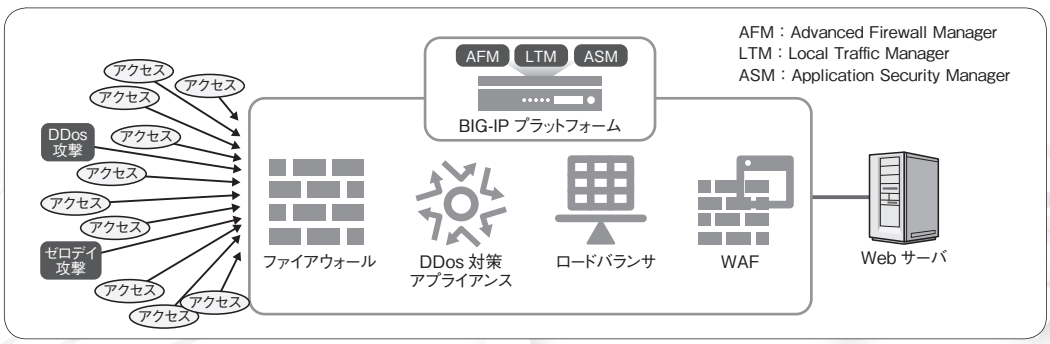
震災以降、ディザスタリカバリーやBCPに対する関心が高まっています。データセンタレベルでアクセスを分散する「広域負荷分散」機能でそうした要望に応えることができます。BIG-IPはDNSを使った方法で広域負荷分散を実現しています。クライアントはリクエストを送信する前に送信先IPアドレスをDNSサーバに問い合わせます。その際、正常に動作しているデータセンタのIPアドレスを返すことで、障害でダウンしたデータセンタへのアクセスを防ぎます(**図6**)。またBIG-IPは「ジオロケーション」機能でクライアントから一番近いデータセンタを割り出し、遅延と修正を最小限にします。

DNSを利用するため、DNSデータのキャッシュ時間(TTL)を極端に短く設定する必要があります。そのためDNSサーバへのアクセスが多発し、DNSサーバにかかる負荷が大きくなります。サービス品質を低下しないようBIG-IPプラットフォームがDNSサーバとなり、DNSのパフォーマンスを強化します。また同時にDNSに対する攻撃も防ぎます。

仮想アプライアンスとハードウェアアプライアンス

ネットワークもサーバも仮想化が主流となり、BIG-IPからもXenやVMware ESXiといったハイパーバイザ上で動作する仮想アプライアンス製品の「Virtual Edition(VE)」がリリースされています。ただし、すべての処理をソフトウェアで行うため、ハードウェアアプライアンスには及びません。データセンタのゲートウェイに

▼図5　BIG-IPのセキュリティ機能

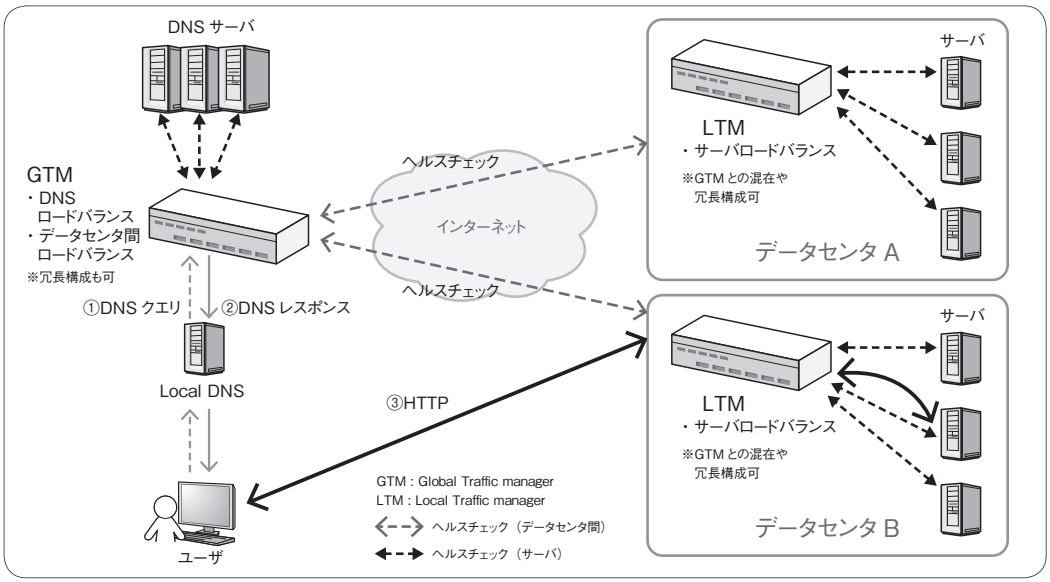


ハードウェアアプライアンスのロードバランサを、IaaS基盤には仮想アプライアンスを配置し、全体のコストとパフォーマンスを適正にします。多数のロードバランサが同時に配置されると、管理問題が発生しますが、BIG-IPは後述するシステム連携機能で、設定や運用管理にかかる手間を抑えることができます。

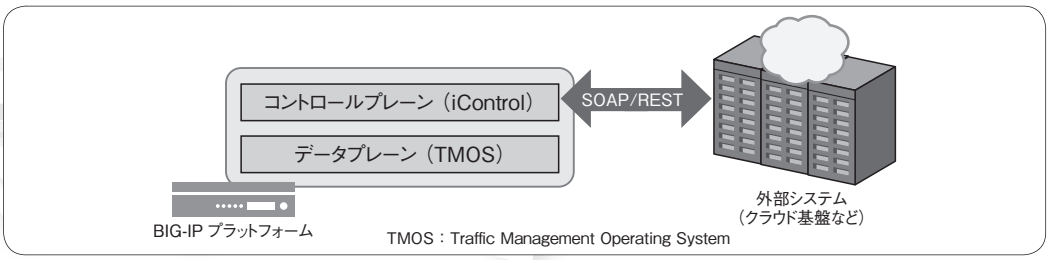
他システムとの連携

大量の機器が導入されると、設定や運用にかかるコストが問題になります。また最近は特定ベンダに依存する「ベンダロックイン」を避ける傾向にあります。こうした問題を解決するには、他ベンダとのシステム連携機能を解放し、プログラムで設定を変更できるようにします。どのベンダとも相互に接続できる「SDN/OpenFlow」が注目されているように、他ベンダとのシステム連携が重要になっています。BIG-IPはOpenFlowプロトコルには対応していませんが、VXLANやNVGREといった新世代のネットワーク仮想化プロトコルに対応し、旧来のネットワーク仮想化プロトコルであるVLANとのゲートウェイとして利用できます。また、内部でSDN(Software Defined Network)のモデルと同じように「コントロールプレーン(制御)」と「データプレーン(トラフィック転送を処理)」に分かれています。コントロールプレーンはSOAPやRESTベースのインターフェースが公開されているため、BIG-IPを操作するアプリケーションをプログラミングし、外部からBIG-IPを設定できます(**図7**)。クラウドシステムと連携したオンデマンドなインフラ構築が可能です。SD

▼図6　BIG-IPの拠点間冗長



▼図7　外部システムとの連携が可能



CHAPTER 2

ハードウェアよりも自由度が高い？ ソフトウェアベースのロードバランサ

LINE㈱　佐野裕（さの ゆたか）Twitter @sanonosa

本稿ではソフトウェアベースのロードバランサについて紹介します。ハードウェア専用機とソフトウェアベースとの違い、オープンソースを用いたロードバランサの構築例、およびWindows Serverでのロードバランス機能などについて紹介します。

はじめに

ロードバランサは、クライアントからのリクエストを複数のサーバに分散させる目的で用いられます。

ロードバランサでは基本機能として、リクエストの振り分けによる負荷分散機能やノードサーバのヘルスチェック（死活監視）機能があります。この目的を達成するために市販のロードバランサ製品を購入して使う方法のほか、サーバにオープンソースなどを用いてソフトウェア的にロードバランス機能を実現して使う方法があります。

ロードバランサはWebサーバの負荷分散で用いられることが一般的ですが、ロードバランサがサポートしていればHTTPやHTTPS以外のプロトコルでも負荷分散ができます。たとえば実際の現場ではSMTPサーバやMySQLのSLAVEサーバへのリクエストを負荷分散するという用途でのロードバランサの採用例をよく見かけます。

ハードウェア専用機とソフトウェアベースとの比較

ロードバランサを導入する際、市販のアプライアンス製品・ハードウェア専門機を購入して使う方法と、サーバにオープンソースなどを用いてソフトウェア的にロードバランス機能を実現して使う方法があるというのはすでに記したとおりです。ハードウェア専用機とソフトウェアベースのいずれにおいても、ネットワークインターフェースから入ってきたリクエストをノードサーバのいずれかに振り分けるという意味では基本的に同じです。

ここでハードウェア専用機とソフトウェアベースでの主な違いを比較します。

1　初期投資費用

ハードウェア専用機のロードバランサは数十万円から数百万円単位とけっこう高価です。ましてや冗長化を考えると2台以上必要となるためさらに高価になります。それに対してソフトウェアベースであれば安価なサーバにLinuxを入れてオープンソースで構築すれば数万円から十数万円程度でロードバランサを構築できます。

2　処理能力

ハードウェア専用機とソフトウェアベースで用いられるサーバで同等スペックのハードウェアが用いられる場合、ハードウェア専用機のほうがロードバランスの機能に最適化されている分、一般的にハイパフォーマンス・ハイキャパシティと言えそうです。ただし後述するSSLオフロード機能などについては専用回路で処理するハードウェア専用機が多く、ソフトウェアベースと比較して圧倒的に高い処理性能を持つと推

定されます。

また、ハードウェア専用機は一般的に処理能力が仕様として公表されているため、どの程度のキャパシティや応答速度があるかあらかじめ把握することができます。それに対してソフトウェアベースでは、用いるサーバや設定する環境によって性能が異なるため処理能力を一概に述べることができません。よって処理能力を把握するためには自身でベンチマークテストを行って性能検証する必要があります。

3 付加機能

ハードウェア専用機ではSSLオフロード機能、HA構成、セッション維持機能、キャッシュ機能、ソーリーサーバ転送機能などといった諸機能が標準、もしくはオプションで提供されていることが一般的です。それに対してソフトウェアベースではリクエストの振り分け機能やヘルスチェックといったロードバランサとして最低限の機能しか提供していないものが多く、ハードウェア専用機並みにさまざまな機能を用いるためには複数のオープンソースを組み合わせるなどの方法で実現する必要があります。

各付加機能の概要は次のとおりです。

■SSLオフロード機能

SSL通信は大量の演算処理が伴うためWebサーバ側でCPU負荷がかかります。そこでSSL通信の演算処理をロードバランサ側で代行し、SSL通信をハードウェア的に処理することでWebサーバ側の負荷を減らします。この機能のことをSSLオフロード機能と呼びます。SSLオフロード機能ではクライアントとロードバランサの間はHTTPS通信を行い、ロードバランサとWebサーバの間はHTTP通信を行います。

■HA構成

ロードバランサの冗長化を行うことで、片方のロードバランサに障害が起きてもロードバランサとしての機能が止まらないように構成できます。

■セッション維持機能

同一クライアントからのリクエストを同じノードサーバに振り分けるしくみです。たとえばECサイトで買い物かごに入れた商品をサーバ側で一時保存しているようなプログラムを使っている場合、クライアントからのリクエストが発生するたびに異なったWebサーバに振り分けられてしまうと買い物かごの中身に矛盾が生じます。そこでセッション維持機能を用いると同一クライアントからのリクエストを同じWebサーバに振り分けてくれるようになるのでそういった矛盾が起きなくなります。

ロードバランサがクライアントと振り分け先サーバの関係を判断するための方法はさまざま存在します。代表的なものはCookieに書き込むもの、URLを書き換えるもの、もしくは接続元のIPアドレスを参照するものなどがあります。

■キャッシュ機能

Webサーバの代わりにロードバランサ側でコンテンツデータをキャッシュしておき、リクエストに応じてロードバランサがWebサーバに代わってレスポンスを返すしくみです。レスポンス速度向上とWebサーバ負荷軽減の効果があります。

■ソーリーサーバ転送機能

同時リクエスト数が一定の閾値を超えたときに、通常のWebサーバの代わりに「ただ今混雑しています」のようなメッセージを表示するソーリーサーバと呼ばれるWebサーバに振り分ける機能です。

4 ロードバランサ運用の管理主体

ロードバランサ運用の現場では、ハードウェア専用機の場合はどちらかというとネットワーク技術者が構築と管理を担当することになるのに対して、ソフトウェアベースの場合はサーバ

技術者が構築と管理を担当する会社が多いと思います。サーバ技術者の立場で見た場合、ハードウェア専用機を導入すればネットワーク技術者がロードバランサの面倒を見てくれるので楽と考えることもできます。一方、ソフトウェアベースで構築すればすべてサーバ技術者側で管理できるので、たとえば障害時自分たちの管轄内で完結するので問題の切り分けがしやすくなるといった考え方もできます。

5　柔軟性

ハードウェア専用機は基本的に中身がブラックボックスです。利用者側で機能追加はできませんし、障害発生時に利用者側でソースコードを分析することもできません。よって何か発生すればすべてハードウェアベンダに問い合わせる必要があります。複雑な障害分析の場合、ハードウェア専用機が海外製品であれば、分析結果が出るまでに数週間から数ヵ月かかることがざらにあります。その点ソフトウェアベースであれば機能拡張やソースコード分析は自分たちの思いのままです。

ソフトウェアでロードバランスを実現する方法

ソフトウェアでロードバランスを実現する方法はいろいろあります。最近比較的よく使われるものとして次が挙げられます。

・LVS(Linux Virtual Server)
・Apache
・UltraMonkey-L7
・Nginx

著者の主観は次のようなものです。「LVS」はLinuxカーネル上で実現されたL4レベルでのロードバランスを実現しています。国内で最もよく使われているため技術情報の入手が容易という安心感があります。「Apache」を用いたロードバランスは設定すべき記述が多くて管理が難しいという印象があります。「UltraMonkey-L7」はL7レベルでのヘルスチェックを行える国内では最も有名なオープンソースと言えそうです。そして「Nginx」はシンプルな設定でロードバランス設定が可能という印象があります。

どれを採用するかは考え方しだいです。パフォーマンスが高いもの、設定がシンプルで簡単なもの、まわりで使っている人が多くていろいろ質問ができるものなど、価値観はいろいろです。

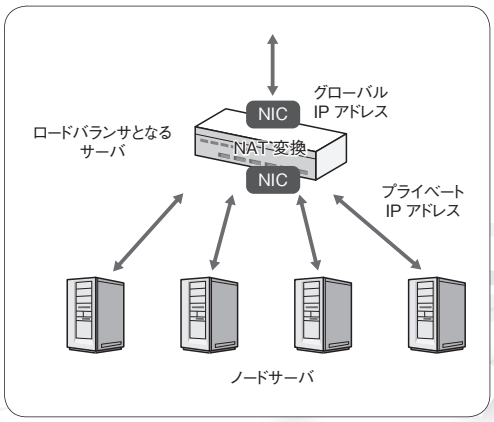
ソフトウェアロードバランスの構成

ソフトウェアでロードバランサを構築する構成としてよく使われるものにNAT(Network Address Translation)構成、フラットベース構成、そしてDSR(Direct Server Return)構成などがあります。それぞれの違いを簡単に紹介します。

NAT構成

ロードバランサとなるサーバに2つのNICを用意し、それぞれにグローバルIPアドレスとプライベートIPアドレスを振る構成です(**図1**)。この構成の特徴は、一般的なNAT構成同様にグローバルIPアドレスがロードバランサのアップリンク側だけに振られ、ノードサーバ側はプライベートIPアドレスが振られることでノードサーバのIPアドレスをインターネット側から隠ぺいします。

▼図1　NAT構成



すべてのリクエストとレスポンスの通信がロードバランサとなるサーバを介すことになるため、通信量が増えれば増えるほどロードバランサがボトルネックとなりやすい構成とも言えます。たとえば1GbpsのNICを用いている場合は物理的制限が受信側と送信側それぞれ1Gbps(125MB/秒)となることに注意する必要があります。ただ1Gbpsというのはけっこう大きな通信量となりますので、そこまでの通信量が発生するにはよほど利用者の多いシステムとなるでしょう。

L2フラットベース構成

ロードバランサとなるサーバに1つのNICを用意し、ノードサーバと同じサブネットでIPアドレスを割り振る構成です(**図2**)。この構成では、リクエストを一度ロードバランサで受け取り、再度ロードバランサからノードサーバにリクエストを投げます。

既存ネットワークにロードバランサとなるサーバを追加する構成となるため導入が最もしやすい構成と言えます。しかしNAT構成と同様にすべての通信がロードバランサとなるサーバを介すことになるためロードバランサがボトルネックとなりやすい構成と言えます。

DSR構成

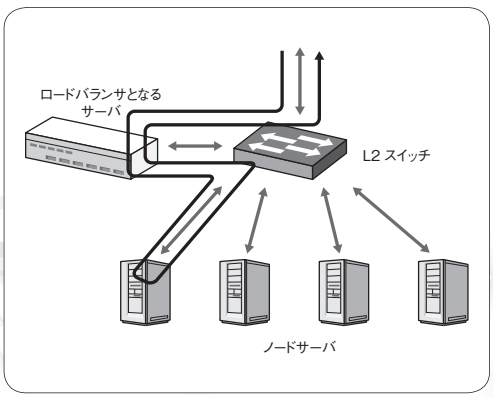
ロードバランサとなるサーバに1つのNICを用意し、グローバルIPアドレスを割り振ります(**図3**)。DSR構成下でのロードバランサの挙動としては、クライアントからリクエストが投げられると、ロードバランサは宛先となるIPアドレスを書き換えずパケットの送付先のみ変更してから各ノードサーバにリクエストを転送します。

ノードサーバではNICに割り振るIPアドレスとは別に、ループバックと呼ばれるIPアドレスをNICにエイリアスとして割り当てます。このループバックにはVIP(Virtual IP)を割り振ります。この設定を行うことでノードサーバではVIP向けに飛んできたリクエストを受信できます。レスポンスを返すときはNAT構成のときと違ってロードバランサではなくインターネット側に直接通信を返します。

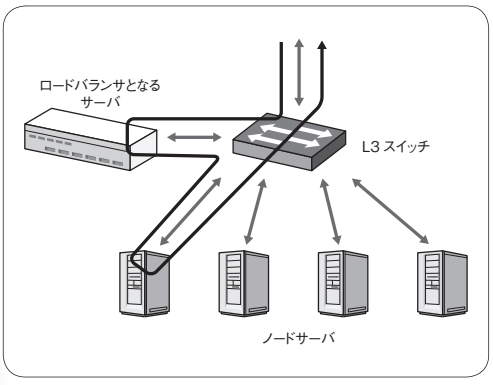
DSR構成を用いる大きなメリットは、ロードバランサの使用帯域を大幅に削減できる点です。一方DSR構成はすべてのノードサーバにループバックIPアドレスを振る必要があるため若干運用コストが上がるというデメリットがあります。リクエスト数がロードバランサのキャパシティに十分収まる小さいサイトではあえてDSR構成を用いる必要はないかと思いますが、リクエスト数が多いサイトではDSR構成の導入を考えるのが良いと言えます。

DSR構成については**Chapter3**でも紹介されていますので詳細はそちらをご覧ください。

▼図2　L2フラットベース構成



▼図3　DSR構成



Nginxによるロードバランスの実例

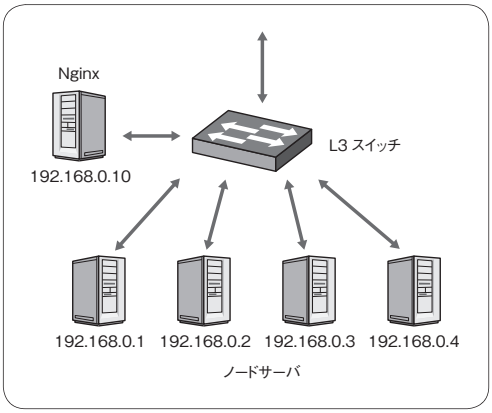
実例として、まずはNginxを用いたL2フラットベース構成での設定例を紹介します(**図4**)。

Nginxは高速なWebサーバとして有名ですが、ロードバランサとしても使うことができます。Nginxの場合は死活監視の設定を行わなくても、Webサーバが落ちている場合は自動的に振り分け先から除外されます。単にリクエストの振り分けのために用いるのであればNginxはソフトウェアベースのロードバランサとしては最も設定が簡単な部類に入ります。

以下はCentOS 6での設定例となりますのでご了承ください。

まずはNginxのインストールを行います。いろいろなインストール方法があります。**図5**は公式ページにあるCentOS用のリポジトリを使う方法です。もちろんソースコードを入手してコンパイルしてインストールしてもかまいません。

▼図4　Nginxによるロードバランス



▼図5　CentOS用リポジトリによるNginxのインストール

```
# rpm -ivh http://nginx.org/packages/centos/6/noarch/RPMS/nginx-release-centos-6-0.el6.ngx.noarch.rpm
# yum install nginx
```

次に設定です。yumでインストールした場合は/etc/nginx/nginx.confを書き換えます(**リスト1**)。設定例では、各Webサーバを「backend」という名前で定義し、80番ポートでlistenするように記しています。

以上の設定を行ったらNginxを起動します。

```
# service nginx start
```

Nginxを起動したら正常に稼働するか確認してみましょう。

```
$  curl http://192.168.0.4/index.html
```

Nginxのロードバランス機能ではリクエストの振り分けに重み付けを行うことができます。たとえば**リスト2**の例であれば、ノードサーバ1,2,3,4がそれぞれ2,2,3,3の割合でリクエストが割り振られるようになります。

▼リスト1　/etc/nginx/nginx.conf

```
http {
  upstream backend {
    server 192.168.0.1:80;    ← ノードサーバ1
    server 192.168.0.2:80;    ← ノードサーバ2
    server 192.168.0.3:80;    ← ノードサーバ3
    server 192.168.0.4:80;    ← ノードサーバ4
  }

  server {
    listen 192.168.0.10:80;
                ↑ listen IPとポート番号の指定
    server_name www.example.com;

    location / {
      proxy_pass http://backend;
    }
  }
}
```

▼リスト2　重み付けの例

```
  upstream backend {
    server 192.168.0.1:80 weight=2;
                              ↑ ノードサーバ1
    server 192.168.0.2:80 weight=2;
                              ↑ ノードサーバ2
    server 192.168.0.3:80 weight=3;
                              ↑ ノードサーバ3
    server 192.168.0.4:80 weight=3;
                              ↑ ノードサーバ4
  }
```

同一のクライアントIPからのリクエストを同じノードサーバで処理したい場合は**リスト3**のように設定します。ただしスマートフォンや携帯からのアクセスの場合、キャリアや回線によって接続のたびにIPアドレスが変わる場合があるため、セッション維持の目的でip_hash機能を用いることは避けたほうが良いでしょう。

LVSによるロードバランスの実例

次にLVSで用いたNAT構成での設定例を紹介します(**図6**)。LVSの場合はノードサーバの死活監視機能が備わっていないため、死活監視機能が必要であれば別途keepalivedなどのソフトウェアを用いる必要があります。

まずはLinuxカーネル内に実装されているIPVS(IPロードバランスソフトウェア)を扱うために、必要となるソフトウェアをインストールします。

```
# yum -y install ipvsadm
```

IPパケット転送を有効にするように、Linuxのカーネルパラメータの設定変更も行います。

```
# vi /etc/sysctl.conf
        :
        :
        :
net.ipv4.ip_forward = 1
        :
        :
        :
```

そして次のコマンドを実行すると設定内容が即時適用されます。

```
# /sbin/sysctl -p
```

次にLVSの設定です。LVSではコマンドを実行することで設定を行っていきます。まずはVIPの追加です。なお、-sオプションでは、たとえばラウンドロビン方式の場合はwrr、最小コネクション方式の場合はlcを記述します。

```
# ipvsadm -A -t (VIPとなるIP):80 -s wrr
```

次にノードサーバを追加します。なお-mオプションを付けるとNAT構成、-gオプションを付けるとDSR構成となります。今回はNAT構成ですので-mを記述します(**図7**)。

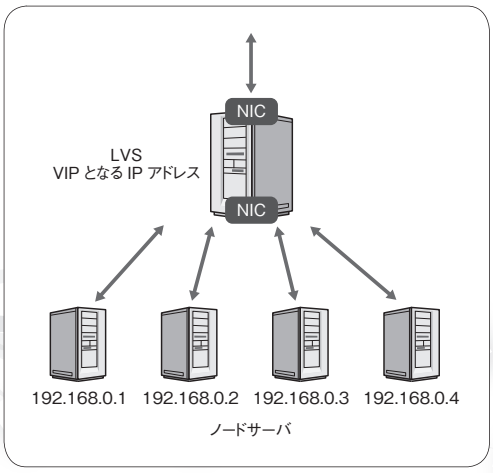
LVSの設定が完了したら正常に稼働するか確認してみましょう。

```
$  curl http://(VIPとなるIP)/index.html
```

▼リスト3　同一クライアントの処理を同じノードで処理する場合

```
upstream backend {
  ip_hash;
  server 192.168.0.1:80;   ← ノードサーバ1
  server 192.168.0.2:80;   ← ノードサーバ2
  server 192.168.0.3:80;   ← ノードサーバ3
  server 192.168.0.4:80;   ← ノードサーバ4
}
```

▼図6　LVSによるロードバランス



▼図7　ノードサーバの追加

```
# ipvsadm -a -t (VIPとなるIP):80 -r
192.168.0.1:80 -m
# ipvsadm -a -t (VIPとなるIP):80 -r
192.168.0.2:80 -m
# ipvsadm -a -t (VIPとなるIP):80 -r
192.168.0.3:80 -m
# ipvsadm -a -t (VIPとなるIP):80 -r
192.168.0.4:80 -m
```

LVSの稼働状況は**図8**のように確認できます。

HA(冗長化)構成

ソフトウェアでロードバランサを作る場合、ロードバランサを実現するサーバ自身の障害についても考慮する必要があります。もしソフトウェアベースのロードバランサが冗長化構成でない場合、ロードバランサに障害が発生するとはサービスが止まることを意味します。

このような場合に備えて、HA(High Availability)構成を組むことを検討するのが良いでしょう。HA構成を実現する際よく使われるものにHeartbeatなどのオープンソースがあります。これらのソフトウェアは、ロードバランサの動作を常時モニタリングし、アクティブであった機器の機能停止を検知すると自動的にスタンバイ状態の機器をアクティブにします。

HA構成の実現方法は実装方法によってさまざまですが、よく用いられる方法としてはネットワーク機器であるルータの冗長化で用いられるVRRP(Virtual Router Redundancy Protocol)と呼ばれるプロトコルを用いるものが挙げられます。

▼図8　LVSの稼働状況の表示

```
# ipvsadm -l
IP Virtual Server version 1.2.1 (size=4096)
Prot LocalAddress:Port Scheduler Flags
  -> RemoteAddress:Port         Forward Weight ActiveConn InActConn
TCP  (VIPとなるIP):http wrr
  -> 192.168.0.1:http             Masq    1      4          0
  -> 192.168.0.2:http             Masq    1      2          1
  -> 192.168.0.3:http             Masq    1      3          2
  -> 192.168.0.4:http             Masq    1      3          0
```

Column　Windows OS標準のロードバランス機能

せっかくの機会ですので、本誌では触れられることが少ないと思われるWindows OS標準のロードバランス機能についても紹介してみましょう。

Windows Serverは、NLB(Network Load Balancing)というロードバランス機能を標準で備えています。ロードバランス機能と言ってもリクエストを複数のノードサーバに振り分けるのではなくて、各ノードサーバにNLB設定を行うことで、ロードバランサを用いずとも各サーバ間で自動的に連携され負荷分散されるしくみとなります。

NLBにはユニキャストモードとマルチキャストモードがあります。

ユニキャストモードでは、各ノードのNICに同一のVIPと仮想MACアドレスに置き換わり、VIPに通信があるたびに各NLBドライバーが協調して毎回違うサーバにリクエストが割り振られるように上位スイッチに対して応答を行います。

それに対してマルチキャストモードでは、仮想MACアドレスとしてマルチキャストMACアドレスが割り振られ、各ノードのMACアドレスはそのまま維持されます。

ご想像のとおりこのしくみはややトリッキーで、NLBを用いる場合はしくみをよく理解しておかないと、環境によっては原因切り分けが困難な障害に見舞われることがあります。著者も過去各ノードと接続されているL2スイッチとの相性問題で苦労した経験があります。

VRRPは、アクティブとなるマスタ機器1台とそれ以外のスタンバイとなるバックアップ機器で構成されます。HA構成を取る機器同士でVRRPグループを構成し、すべての機器に同じ仮想IPアドレスを割り振ります。

マスタ機器は適時マルチキャストでハートビートを送りますが、このハートビートが途絶えた場合、バックアップ機器がマルチキャストパケットを送り、生存している機器の中で最もプライオリティが高い機器がマスタに昇格します。

最後に

ロードバランサは狭義では負荷(ロード)を分散(バランス)する意味でしかなく、ソフトウェアベースのロードバランサはその目的を果たすことのみを追求しているものが多い気がします。それに対してハードウェア専用機の場合は、顧客ニーズが多様化しているからなのか、それともマーケティング的に売れる機能を追い求めているのかわかりませんが、年々高機能化に向かっている気がします。正直ロードバランサというよりは多機能リバースプロキシ装置と呼んだほうが良いのではないかと思ったりもしています。

オープンソースを組み合わせて自分たちの理想とするロードバランサを構築しようとする場合、どの程度手を入れて構築するかの見極めが重要そうです。なぜならものすごく手を入れて構築したシステムは概して運用がたいへん、運用をほかの人に引き継ぐのがたいへん、マシンが老朽化した場合入れ替えるのがたいへんなどといったことが起きるからです。

ハードウェア専用機とソフトウェアベースのロードバランサではそれぞれ利点と欠点がありますのでそれらを十分見極めて導入を検討するのが良いと思います。SD

CHAPTER 3

さくらインターネットにみる クラウド上でのロードバランサの実装と利用

さくらインターネット㈱　大久保 修一（おおくぼ しゅういち）

本稿では、クラウド上でロードバランサを使う際にケアするべき点、ケーススタディとして「さくらのクラウド」上で実際に利用する方法、また、クラウドのロードバランサがどのように実装されているのか、システムの裏側についても紹介します。

クラウドにおけるロードバランサ

最近では「クラウド」という言葉が、以前ほどもてはやされなくなったように感じます。逆にとらえると、クラウドは普及期に入り、そのうえでさまざまなサービスが「当たり前」に動いている時代になったことの表れでしょう。

もはやクラウドのメリットをあらためて説明することもなさそうですが、必要な機能、性能を組み合わせて柔軟にシステムを構成できます。しかも、サーバ、ストレージ、ネットワークといったリソースは、すぐに拡大・縮小できます。アクセス数が急激に伸びてもサーバをすぐに追加して対処可能というわけです。トラフィックの変化を予測しにくい、現在のインターネットサービスを安定稼働させるために、非常にマッチしたインフラであると言えます。

しかし、単にサーバ数を増減してもユーザのエクスペリエンスを向上させることはできません。クラウドのメリットを最大限引き出すためには、ユーザからのアクセスをサーバの増減に追従して適切に振り分ける機構、「ロードバランサ」が非常に重要な役割を担います。

▼表1　負荷分散の方式

#	LB配置場所	接続構成	中継方式
#1	事業者バックボーン	-	NAT、プロキシ
#2	ユーザネットワーク	1アーム	NAT、プロキシ、DSR
#3	ユーザネットワーク	2アーム	NAT、プロキシ

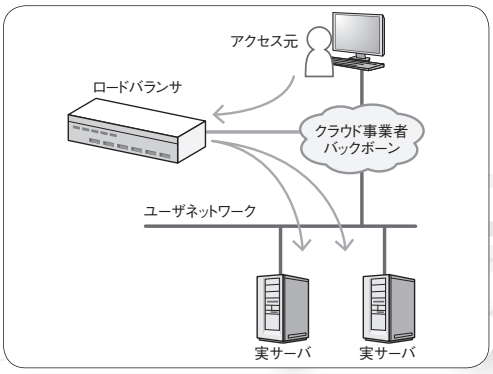
クラウドベンダによって異なる提供方式

ロードバランサと一口に言っても、クラウドベンダによって提供される構成、方式はさまざまです。大きく分類すると、**表1**の3種類の方式が存在します。それぞれの構成図を**図1～3**に示します。

#1のパターンでは、ユーザからのアクセスはいったんクラウド事業者のバックボーンに配置されたロードバランサで終端されます。実サーバ[注1]にはバックボーン経由で振り分けられますので、バックボーンから到達性のあるネットワー

注1）本稿では、ロードバランサから見てアクセス振り分け先のサーバを「実サーバ」と表記します。クラウド上ではそれらも仮想化されたサーバですので、若干しっくりきませんが。

▼図1　#1 事業者バックボーンに配置されるパターン

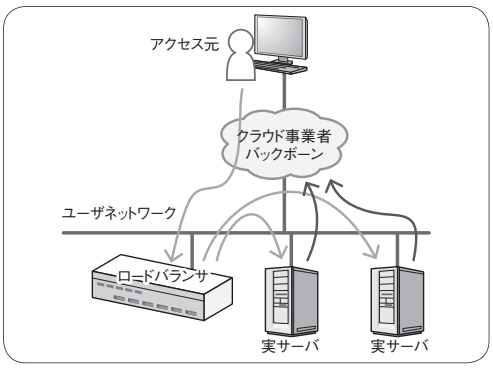


クに実サーバを設置する必要があります。ユーザ自身のプライベートなネットワークでの利用はできません。

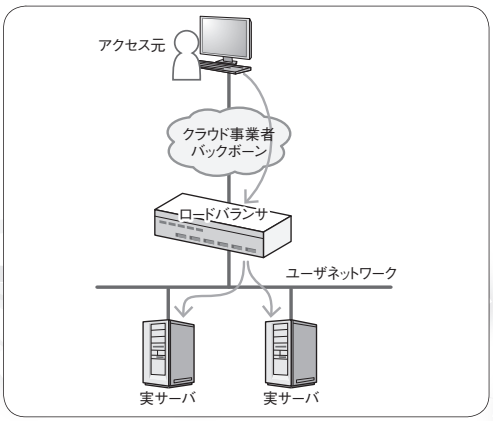
#2、#3のパターンでは、ユーザのネットワークにロードバランサを直接アタッチすることが可能な構成です。外部に面したネットワークはもちろん、プライベートな内部ネットワークでの利用もできます。2アーム構成は2面のネットワークが必要になるものの、実サーバをプライベートネットワークに分離して、セキュアな構成をとることができます。1アーム構成は、のちほど紹介するDSR方式で多く用いられます。

そのほか、クラウド利用者はあまり意識する必要はないかもしれませんが、ロードバランサの機器として、専用ハードウェアを用いるもの、仮想アプライアンスを用いるものなどがあり、機能も性能もさまざまです。とくに、セッション数や新規コネクション性能は大きく変わりますので、クラウド上で提供されるロードバランサが、運用するサイト要件を満たしているか、事前に確認しておきましょう。

▼図2　#2 ユーザネットワークに1アームで配置するパターン



▼図3　#3 ユーザネットワークに2アームで配置するパターン



「さくらのクラウド」のロードバランサ機能

「さくらのクラウド」は、さくらインターネットが提供しているIaaS（Infrastructure as a Service）で、サーバ、ストレージ、ネットワークといった仮想化されたリソースを柔軟に組み合わせてシステムを構築することができます。従来、物理機器で行っていた操作をコントロールパネル上に再現することにより、SDDC（Software-Defined Datacenter）を実現することを目標に開発を行っています。ここでは、さくらのクラウドをケーススタディとして、クラウド上でのロードバランサの活用方法を説明します。

さくらのクラウドのロードバランサは、小規模なサイト向けに手軽で安価に使えるもの、というコンセプトになっており、主な仕様は**表2**のようになっています[注2]。**表1**の分類でいうと#2に該当し、インターネットに面するネットワークでの利用はもちろん、ユーザのプライベートネットワークでデータベースサーバなどの負

注2）詳細はhttp://cloud.sakura.ad.jp/を参照ください。

▼表2　さくらのクラウドのロードバランサ仕様

構成	1アーム、DSR方式
ロードバランサの冗長化	VRRP[注3]を用いた冗長化に対応
仮想IPアドレス(VIP)数	最大4個まで
実サーバ数	1VIPにつき10個まで
性能目安	100Mbps、4000セッション、100cps程度
振り分け方式	Least Connection
対応レイヤ	レイヤ4
ヘルスチェック方式	PING、TCP、HTTP
ヘルスチェック間隔	秒単位指定（ただし10秒以上）

注3）Virtual Router Redundancy Protocol

荷分散にも利用することもできます。

DSR方式について

さくらのクラウドでは、DSRと呼ばれる方式を採用しています。DSRはDirect Server Returnの略で、ロードバランサはアクセス元から実サーバ向けのトラフィックのみをコントロールします。実サーバからアクセス元への返りトラフィックは、ロードバランサを介さず直接ゲートウェイに向けることができます。そのため、リクエストよりも配信トラフィックが大きい一般的なWebサイトのトラフィックバランスであれば、ロードバランサが帯域的なボトルネックになりにくいというメリットがあります。**図2**はDSR方式のトラフィックフローを示しています。

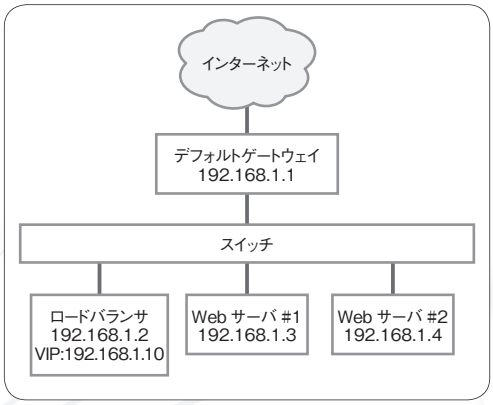
DSR方式を使っているクラウドは珍しいと思われるかもしれません。実は、DSR方式を採用するにいたっては紆余曲折があったのですが、その経緯についてはコラムをご覧ください。

「さくらのクラウド」でロードバランサを使ってみよう

ここでは、**図4**のように複数のWebサーバに対するアクセスを負荷分散するようなケースでの設定例を紹介します。説明で使用する各サーバのIPアドレスは**表3**のとおりです。

なお、さくらのクラウドではロードバランサを配置するにあたって、専用のプライベートネットワークである「スイッチ」、もしくは専用のグローバルネットワークである「ルータ＋スイッチ」が必要です。あらかじめご準備ください。

▼図4　ロードバランサの構成例



手順1　Webサーバの設定

DSR方式では、実サーバ（ここではWebサーバ）に仮想IPアドレス（VIP）を設定する必要があります。CentOSなどでは次のようにループバックインターフェースにVIPを設定します。

```
# cd /etc/sysconfig/network-scripts
# vi ifcfg-lo:0
DEVICE=lo:0
IPADDR=192.168.1.10
NETMASK=255.255.255.255
```

また、Linuxではループバックインターフェースに設定したIPアドレスも、物理インターフェースでARP応答してしまうため、応答しないように設定します。

```
# vi /etc/sysctl.conf
net.ipv4.conf.all.arp_ignore = 1
net.ipv4.conf.all.arp_announce = 2
```

これらの設定を反映するために、いったんリブートするか、次のコマンドを入力します。

```
# sysctl -p
# ifup lo:0
```

手順2　ロードバランサの作成

さくらのクラウドのコントロールパネルにログインすると、「アプライアンス」というタブが

▼表3　ロードバランサ設定例のIPアドレス

機器名	IPアドレス
デフォルトゲートウェイ	192.168.1.1
LB物理IPアドレス	192.168.1.2
Webサーバ#1	192.168.1.3
Webサーバ#2	192.168.1.4
仮想IPアドレス（VIP）	192.168.1.10

あり、そこにロードバランサの追加ボタンがあります。これを押すと、**図5**のような画面が表示されますので、必要事項を入力します。

ここでは「冗長なし」としていますが、冗長化する場合はロードバランサ用のIPアドレスを2個入力します。VRIDは、VRRPを用いてロードバランサを冗長化する場合のペアを示すIDで、複数のロードバランサを同一スイッチに設置する場合はIDが被らないように変更する必要があります。

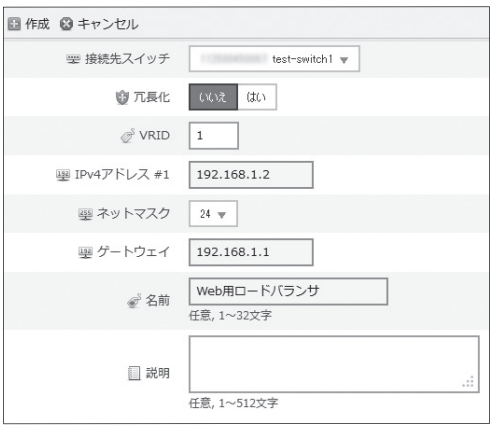
作成後、数分間待つとロードバランサが利用可能となり**図6**のような画面になります。

手順3　VIPの設定

コントロールパネルに「VIP設定」というタブが現れますのでVIPを登録します。追加ボタンを押すと**図7**のような画面が表示されます。

IPアドレスはあらかじめ決めたVIP(192.168.1.10)を、ポート番号は負荷分散を行うTCP待ち受けポート番号(ここでは80)を入力します。チェック間隔は、このVIPに紐付けられた実サーバへのヘルスチェックの間隔を秒数で入力します(10秒以上である必要があります)。

▼図5　ロードバランサの作成画面



▼図6　ロードバランサの作成完了後の画面

手順4　実サーバの登録

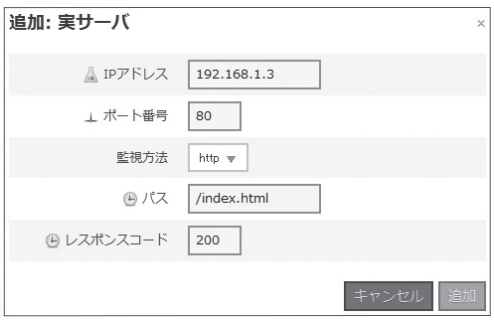
VIPの登録が完了すると、「VIP：ポート番号」のタブが1つ増えますので、そちらからそのVIPに紐付ける実サーバを登録します。追加ボタンを押すと、**図8**のような画面が表示されます。

監視方法は、PING、TCP、HTTPに対応しています。今回はWebサーバの負荷分散を行いますので、HTTPによる実サーバのヘルスチェックを行います。

同じ要領でもう1台のWebサーバのIPアドレス(192.168.1.4)も登録します。上部の反映ボタンを押すとロードバランサに設定がプッシュされ、稼働を開始します。正常に実サーバのヘル

▼図7　VIPの設定画面

▼図8　実サーバの登録画面



Column さくらのクラウドでDSR方式を使っているワケ

DSR方式を採用しているクラウドは珍しく、他社さんのクラウドではHTTPに特化したプロキシに近い方式を採用しているところが多いようです。一方、さくらのクラウドではプロトコルを限定せず、さまざまなアプリケーションで利用可能な汎用性の高いものを目指しました。そういった観点から、DSR以外のNAT方式やプロキシ方式には次のような実装上の課題がありました。

・NAT方式の場合

ソースNATをせず宛先NATのみを行う方式の場合、実サーバのデフォルトゲートウェイをロードバランサに向けなければなりません。そのため、実サーバから外部への通信を通すためにフォワードNAT機能も実装する必要があり、ロードバランサの機能が複雑化します。

ソースNATを行う方式の場合、上記の問題は解決できますが、ソースNAT用のIPアドレスプールの管理機能が必要となってきます。また、実サーバに届くアクセス元IPアドレスがロードバランサのものに置き換わってしまうため、アクセスログを提供する機能も必要となり、やはり機能が複雑化します。

・プロキシ方式の場合

前述のソースNATを行う場合と同じ課題があります。HTTPであればアクセス元IPアドレスをX-Forwarded-Forヘッダに入れて実サーバに伝えることもできますが、そのほかのプロトコルではこのような対応が難しくなります。

DSR方式の場合、アクセス元IPアドレスが変化する問題はありません。また、ロードバランサ以外の機能を実装する必要もなく、実装をシンプルにできました。ただし、DSR方式は良いこと尽くめではありません。次に挙げるような懸念点や制限もあります。

・実サーバに追加の設定が必要

先に紹介した設定例のように、VIPをすべての実サーバに設定する必要があります。もし、実サーバにVIPが正しく設定されていなくても、ロードバランサからのヘルスチェックは通ってしまうため、通信不良が発生してしまいます。

また、物理インターフェースでVIPに対するARPに応答しない設定も必要です。この設定が抜けていると、VIPに対するアクセスを実サーバが直接拾ってしまい、通信がロードバランサを経由しません。一見、正しく通信できているように見えるため、設定漏れに気づきにくいという面もあります。

・ネットワークを分離できない

DSR構成は一面のネットワークのみで組むことができますが、逆に言うと、セキュリティの向上を目的としたネットワークの分離ができないという制限になります。また、グローバルに面するネットワークで使用する際、実サーバに1つずつグローバルアドレスが必要になります。IPv4アドレスが枯渇している現状、コスト高になる可能性もあります。

前者については、手順を適切に案内することでカバーしています。後者については、将来的にL3DSR(異なるネットワークを超えて実サーバを配置可能)に対応したり、対応プロトコルを限定したロードバランサを用意するなど、選択肢を増やすことでカバーしたいと考えています。

スチェックが完了するとステータス欄が緑色になり、UPと表示されます(**図9**)。

設定は以上です。VIPにアクセスして、サービスが正しく応答することを確認してください。正常に疎通していれば、**図9**のコネクション数の部分にロードバランサから各実サーバに振り分けている接続数が表示されます。

「さくらのクラウド」のロードバランサ機能の裏側

さくらのクラウドでは、非常に簡単にロードバランサを設置できることがおわかりいただけたかと思います。ここからは、このシステムがどのように動いているのか、実装の裏側を紹介したいと思います。

システム構成

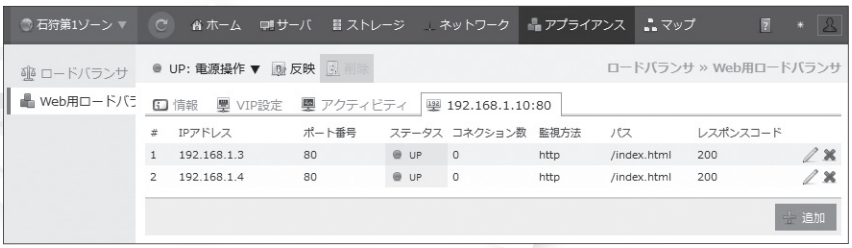
実は、さくらのクラウドのロードバランサ機能は、**表4**に示すような一般的なオープンソースソフトウェアを用いて実装しています。ユーザがロードバランサを作成すると、CentOSにkeepalivedがインストールされたイメージをもとに、クラウド上に仮想マシンを作成、起動します。その後、指定されたネットワークにアタッチし、実サーバのヘルスチェック、VIPに対するアクセスのロードバランシング処理を行います。

実装方法としては、専用ハードウェアを導入し、ユーザごとに論理分割するという選択肢もありました。しかし、さくらのクラウドでは仮想マシンで実装しています。その理由は次の点が挙げられます。

▼表4　さくらのクラウドロードバランサの実装

機器種別	仮想マシンとしてクラウド上で動作
OS	Linux(CentOS)
監視デーモン	keepalived
転送エンジン	Linux Virtual Server(IPVS)
冗長方式	VRRP

▼図9　登録を完了したあとの画面



・特別な機材が不要となる

ロードバランサの仮想マシンを収容するために、通常の仮想サーバを収容するホストサーバ基盤をそのまま利用できます。特別な機材が不要なため、初期投資を抑えられ、その結果安価なサービス提供が可能となります。

・トラフィックが一極集中しない

専用ハードウェアを用いる場合、負荷がそこに集中しボトルネックになる可能性があります。また、ユーザごとのリソース制限の設定が必要となり、収容設計が複雑化します。仮想マシンで動作する場合、クラウド上で分散処理されます。

システム構成は**図10**のようになっています。コントロールパネル上で入力された設定情報は、keepalivedの設定ファイル(keepalived.conf)に変換して、管理用ネットワークを通じて配信しています。設定変更の通知やファイルのやりとりは、SSHとHTTPを使用しています。

DSR方式によるネットワークの問題の対策

クラウドの物理ネットワークは、多数のL2スイッチを組み合わせた巨大なL2ネットワークとして構築されています。ユーザからは仮想化されたスイッチしか見えませんが、物理的にはい

くつものL2スイッチを経由してサーバ間の通信が行われています。

ところでDSR方式の場合、アクセス元→ロードバランサ→実サーバ→アクセス元の通信が三角形となり非対称です。クラウドのL2ネットワーク上で、このような非対称の通信を行うと特有の問題が発生します。**図11**のように、ロードバランサ発のトラフィックが一部のL2スイッチを経由しないことがあり、ロードバランサのMACアドレスがエージアウト[注4]してしまうのです。ここではL2スイッチAでその事象が発生します。

エージアウトすると、ロードバランサ宛のパケットが該当のL2スイッチでフラッディングし、通信が不安定になります。場合によってはほかの通信にも影響を及ぼします。この事象を回避するため、さくらのクラウドではロードバランサから一定間隔でGARPパケットをブロードキャストするような設定を行っています[注5]。

これにより、すべてのスイッチでMACアドレスが定期的に再学習され、通信不良の発生を防ぐことができます。

スペックアップの検討

当初さくらのクラウドでは、小規模なサイト向けに手軽で安価なものを、というコンセプトの元、機能や性能を限定したロードバランサの開発を行いました。しかし、大規模なサイトをクラウド上で運営されるユーザからは、ハイスペックなインスタンスの要望も増えており、性能を強化すべく現在開発を行っています[注6]。

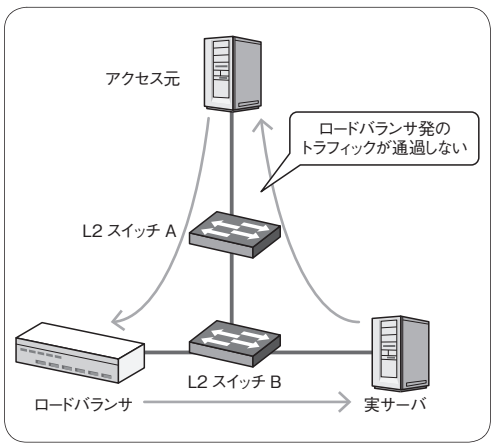
注4） 通信していないMACアドレスを自動的に削除するL2スイッチの機能。デフォルトで5分経過後に削除処理が行われる。

注5） Gratuitous ARPの略で、直訳すると「不必要な、余計なARP」という意味ですが、自身のIPアドレスとMACアドレスの情報を同一ネットワークの機器に伝達する際に用いられます。具体的には、次のようなarpingコマンドをcronで実行しています。

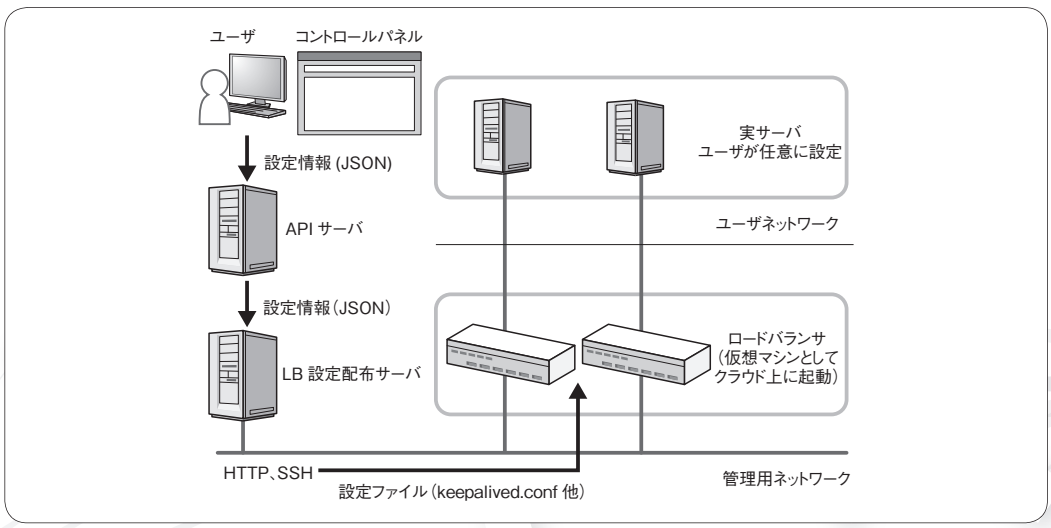
```
% arping -A -c 1 xx.xx.xx.xx ←LBのIPアドレス
```

注6） 順調に進めば、本誌が発売されるころにはリリースできているかもしれません :-)

▼図11　MACアドレスのエージアウトが発生する例



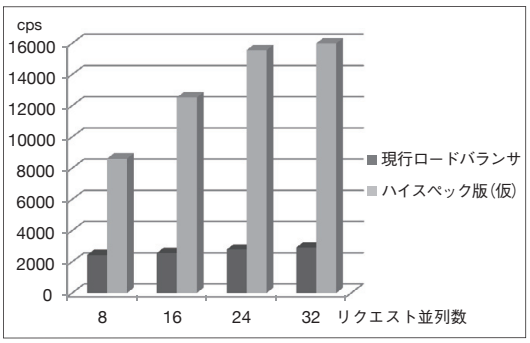
▼図10　ロードバランサシステム構成図



アーキテクチャを変えずに性能を向上するには、単純に仮想マシンのCPUコア数を増やしたり、NICを準仮想化ドライバに対応したりといった方法があります。実際、それらの対策を講じることで新規コネクション性能(cps = Connections per second)を大幅に向上することが可能でした。

現行プランとの性能比較を**図12**に示します[注7]。こちらはWebサーバの負荷テストツールであるweighttpを用いて、リクエスト並列数を変化させた際の性能値を比較しています。現行プランに比べ、ハイスペック版(仮)は最大5倍程度の性能を引き出すことができています。

▼図12　現行プランとハイスペック版(仮)の性能比較



おわりに

本稿では、クラウド上でロードバランサを使用する際の留意点、さくらのクラウドでのロードバランサの利用方法と実装の裏側について説明してきました。冒頭に述べましたように、ロードバランサを組み合わせることでクラウドの真価を引き出せるようになります。ぜひ、設計方法や使い方をマスターし、スケーラブルでダウンしにくいシステムを構築できるようになっていただければと思います。SD

注7) これは限界性能を示したものであり、実際はホストサーバの収容設計などを勘案し、お客様には担保できる性能値を利用目安としてご案内しております。

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。

2014年3月号

第1特集
データベースの諸問題
RDBとNoSQL どちらを選びますか？

第2特集
ネットワークエンジニアのための
プロキシサーバの教科書

一般記事
・さらに踏み込む、Mac OS Xと仮想デスクトップ #1
定価（本体1,219円+税）

2014年2月号

第1特集
入式からはじめませんか?
関数型プログラミング再入門

第2特集
目利きによるトレンド予測
2014年IT業界はどうなるのか?

一般記事
・会社組織を活性化するスパイス「コンパ」

定価（本体1,219円+税）

2014年1月号

第1特集
シェルがわかればシステムがわかる
あなたの好きなシェルは何ですか?

第2特集
未来を作るITインフラ
10ギガビットを実現する ケーブリング技術

特別付録 & 一般記事
・法輪寺鎮守社電電宮 情報安全護符シール Ver.2
・ソーシャルゲームのDevOpsを支える技術（後編）
特別定価（本体1,314円+税）

2013年12月号

第1特集
SDN/OpenFlowの流れを総まとめ!
SDN/OpenFlowで 幸せになれますか?

第2特集
下手でも好印象で効果絶大
エンジニアの伝わる図解術

一般記事
・LinuxとFreeBSDのファイルシステムの良い・悪いところをご存じですか?
・「Mirama」ほか　定価（本体1,219円+税）

2013年11月号

第1特集
思考をコード化する道具
我が友 Emacs

第2特集
コンピュータの加速装置
サーバサイドフラッシュ Fusion-io全方位解説

一般記事
・小規模プロジェクト現場から学ぶJenkins活用（5）
・ソーシャルゲームのDevOpsを支える技術（前編）ほか
定価（本体1,219円+税）

2013年10月号

第1特集
Vimを使いこなしていますか?
Vim至上主義

第2特集
ネットワーク技術力のパワーアップ
ルータの教科書

一般記事
・Key Value Storeをゼロから創る
・小規模プロジェクト現場から学ぶJenkins活用（4）
定価（本体1,219円+税）

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%強の割引になります。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。

今すぐ知りたいSIMのしくみ

～SIMカード事情～

迫 卓見(さこ たくみ)

今までSIM(シム)といえば、携帯電話やスマートフォンに入っているカードという認識はあったものの、それについて考える機会は少なかったと思います。しかし、最近、iPhone5sやNexus5/7など、SIMフリーと呼ばれる端末が普通に手に入るようになりました。また、MVNO業者のSIMを使うと格安に通信できるという話も耳にするようになり、「SIMって何だろう？」という疑問をお持ちの方もいらっしゃると思います。

SIMの概要と変遷

本稿では、「そもそもSIMってのは何だ」というあたりから、どういう機能を持っていて、どういう概念のもので、どういう種類があるのか、など、SIMが生まれた経緯から現状までを含めて解説します。

まずはSIMの用語から入りましょう。SIMはSubscriber Identity (ID) Module＝加入者IDモジュールの略で、それをカード状にしたものが「SIMカード」と呼ばれています。つまり、一種のIDカードと同じもので、容易に情報が盗まれないようなしくみも入っています(**図1**)。

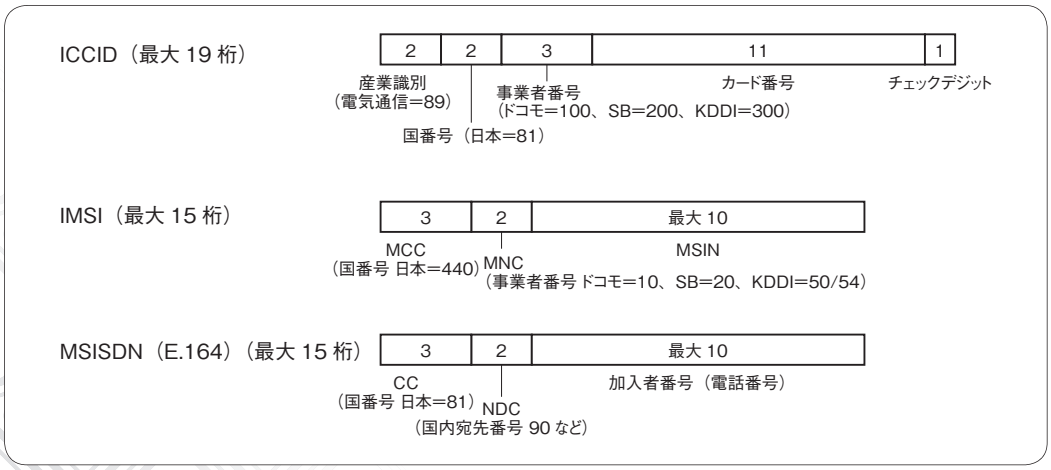
電話番号

このSIMが内部に保持している情報の代表的なものが、電話番号(MSISDN[注1]とも言う)です。国際的に一意に決められており、国番号と加入者電話番号の組み合わせにより世界中に唯一無二の存在になります。たとえば「090-1234-5678」のSIMカードを持っている人は、日本からはその番号が、海外からは国番号を表す「+」記号と日本の国番号(81)を先頭に付けて「+81-90-1234-5678」が、そのSIMへの電話番号になります。

しかし、多くの人が“なぜ090の最初の0が省

注1) Mobile Subscriber Integrated Services Digital Network Number

▼図1　IDの例



略されるのだろう”と思いますよね？　実はこの0は電話番号の一部ではなく、「国内プリフィックス」と呼ばれる、“国内への電話だよ”ということを電話に識別させるための識別番号なのです。ですので、海外から日本にかける場合はこれが「81」に変わるわけです。

電話番号の切り離し

さて電話番号の話が長くなりましたが、このSIMカード、別の角度から言えば、これは「携帯からID情報（電話番号）を切り離したもの」ということになるかもしれません。つまり、このSIMカードを抜いてしまえば、携帯端末本体には電話番号はまったく残らなくなります。なぜこのようなしくみなのでしょうか。のちほど詳しく述べますが、電話番号（つまりキャリアと契約してキャリアから配られる番号）と、携帯端末（電機メーカーが製作して販売する機械）を切り離すことにより、技術的にも社会的にもさまざまなメリットが生まれるからです。

背景

SIMカードは、日本では2001年頃に「3G（The 3rd Generation）」の携帯端末が市場に出現したときに初めて出回りましたが、ヨーロッパではその1つ前の世代（2G）である「GSM（Global System for Mobile communications）」の時代から使われていました（**図2**）。

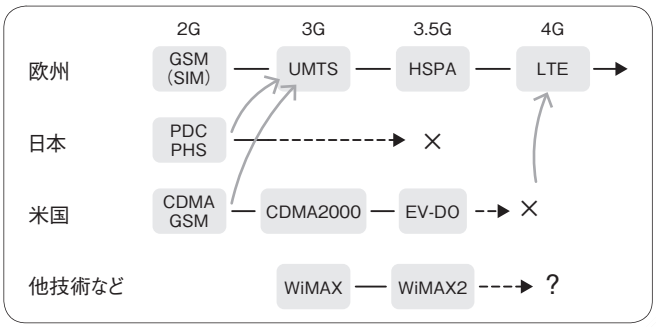
このGSMというシステムが当時非常に優れたもので、ほとんどのヨーロッパのキャリアとメーカーが加入して、国を越えて共同でその規格や仕様を策定し、「どこのメーカーの携帯電話でもどこのキャリアでも動かせる」ようになりました。事実上、初めての国際共通化された携帯システムだったわけです。そのころ日本ではPDC（Personal Digital Cellular）と呼ばれる日本独自の方式を、米国も同じく複数の独自の方式を取っていました。

GSMとSIM

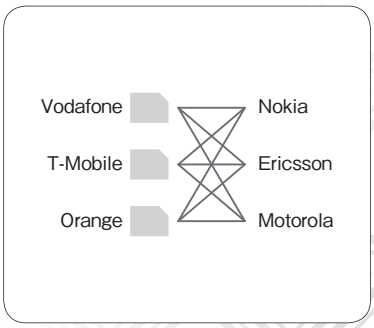
しかし、そのようにするには、「キャリアとメーカーの切り離し」をしなければなりませんでした。なぜなら、キャリアが自分の持つ電話番号を携帯端末に埋め込んで一般ユーザに販売する形態では、メーカーは自社の作った携帯端末をいったんキャリアに納め、キャリアがその携帯端末を一般に販売しなくてはならなかったためです。

これではメーカーは「下請け」となり、キャリアの力が強くなり過ぎてしまいます（日本もそうでした）。そこで、今後のセルラー（キャリア）業界の発展を目指し、業界団体として携帯端末と電話番号の切り離しを行ったわけです（**図3**）。ここで生まれたのが、電話番号を保持したSIMカードだったわけです。端末メーカーは携帯端末を売る。キャリアは通信回線（つまりSIM）を売る。そしてそれらは互換性が確保されています。こうして、ヨーロッパの携帯文化は充実していきました。

▼図2　世界の携帯システムの流れ



▼図3　通信業者と端末メーカーの分離



3GとSIM

2000年頃に3Gのサービスが全世界で始まるにあたって、ヨーロッパ、アメリカ、日本とそれぞれ別の規格で運用されていた携帯システムを世界統一しようという動きになりました。技術的にも複数の候補がありましたが、中でも3G（UMTS：Universal Mobile Telecommunications System）として事実上のスタンダードとして採用された通信技術が、ご存じW-CDMA（Wideband Code Division Multiple Access）方式です。

W-CDMA方式では、従来のCDMA方式の帯域幅を拡張し、高速通信（当時はなんと384kbps）が可能になったわけです。この技術要素そのものの推進を強くしたのが日本陣営です。NTTドコモを軸に、ヨーロッパにこの技術を「ねじ込んだ」形でした。一方、ヨーロッパ陣営はこのコア技術を日本に取らせる代わりに、その他の文化はそのままGSMから踏襲するという、これまた重大な益を取ったことになります。つまり、レイヤ1であるW-CDMAは日本、それより上、とくにレイヤ3（プロトコルスタックのコア）はGSMの拡張で、となりました（**表1**）。ここで、ヨーロッパ勢はSIMという文化を3Gにまで持ち込むことに成功したわけです。

U-SIM

ちなみに3Gシステムは、3GPP（Third Generation Partnership Project）という3G携帯の規格を決める業界団体にUMTSとも呼ばれていました。SIMもそれまでのものと区別するためUMTSのUをとって「U-SIM」と言われていました。しかし、最近ではあまり区別せず単に「SIM」と呼ぶことが多いようです。機能拡張などはされましたが、基本的に2GのSIMの延長です。

▼表1　プロトコルスタック（3G採用時）

L4以上　TCP/IPやSIMなど	欧州のGSM文化をそのまま延長
L3　RNC/MM/CC	
L1～2　W-CDMA	日本の技術を採用

SIMのメリット

SIMという形を導入するメリットは、大きく次の2つがあります。

- ユーザはSIMさえ抜いて別の端末に挿せば、いろんな端末を同じ番号で使えるようになります。つまり端末を複数持ち、好きなものを用途に合わせて使えるようになります。実はこれはメーカーにとってもたくさんの携帯端末が売れることを意味するので、メリットがあります
- メーカーがキャリアの意向に関係なく自由にマーケティングをして携帯端末を作って販売でき、キャリアとの力関係が対等になります。ノキアなどはこのおかげであらゆる形や機能の携帯を自由に作って大きく業績を伸ばしました

SIMの適用されるセルラーテクノロジ

上記のように、もともとSIMというのはヨーロッパのGSMの文化で発祥したセルラー（携帯電話）テクノロジです。それとそれを後継したW-CDMA、そしてその後継であるLTE（Long Term Evolution）などで一般に使われていますが、同じ携帯電話技術でも違う文化の流れではこのSIMはあまり使われていません。先述の日本のPDC（Personal Digital Cellular）やPHS、米国で発達したcdmaOneやその後継のEV-DO（Evolution Data Only (Optimized)）などはその例です。また、そもそも発祥の根源がセルラーテクノロジではない（PC系の）WiMAXやWiFiなどでは当然、このSIMという考え方は採用されませんでした。

SIMに格納される情報

上記のように、SIMにはその携帯電話通信回線をどう契約したか、という情報が凝縮されているのですが、実際にはSIM上にはIDが書か

れ、そのIDを通信事業者が契約情報と関連付けることによってその契約と通信を制御しています。具体的には、次のような情報が書き込まれています。

・ICCID（ICカードID）

国際電気通信連合で定められた、すべてのICカードを一意に識別するためのIDです。通信用のSIMカードのみならず、クレジットカードなどのICカードも対象になります。

・IMSI（International Mobile Subscriber ID＝国際的な端末加入者識別番号）

SIMカードで最も重要な情報です。SIMカードの通信上のIDの根本となるもので、1枚のSIMカードに固有の1つの番号が与えられています。世界で一意に識別できるIDです。MCC（国番号）＋MNC（キャリア番号）＋MSIN（SIMの加入者ID）です。

・MSISDN（Mobile Subscriber ISDN Number＝電話番号）

キャリアと契約するとキャリアから与えられる番号で、いわゆる電話番号です。これはSIMカード固有ではなく、キャリアがどのSIMカードにこの番号を書くかを決定します。IMSIと似ていますが、構成はCC（国番号）＋NDC（キャリア番号）＋SN（加入者番号）です。

・その他

SIM規格やベンダにもよりますが、代表的なものでは簡単な電話帳（電話番号と名前など）などが格納できます。SIMに電話帳を書き込めれば、SIMを抜いて別の携帯端末に挿しても同じ電話帳が使えるようにするためですが、今では電話帳はクラウドに置くのが普通になってきているため、あまり意味はないかもしれません。

SIMの種類

では次に、SIMの種類について述べてみましょう。形態やサイズ、機能など、いろいろなバリエーションがありますが、ざっくり次のような分類ができます。

サイズ

「標準」「マイクロ」「ナノ」の3つが主流です（**写真1**）。そのほかにも規格策定の段階などではいろいろありますが、とくに気にしなくていいでしょう。一般に、サイズの違いによる機能などの違いはほとんどありません。

機能・性能

サイズのほかにSIMを特徴付けるもう1つのものは、機能と性能ですが、これは契約体系に依存します。そのSIMがキャリアとどういう契約がなされているかで、接続スピードや接続テクノロジ（LTEなど）、そしてデータ量など、そのSIMの特徴も変わってきます。契約体系については次項で詳しく述べます。

SIMの契約体系

キャリアと契約するSIMによって、使える機能や性能に差があります。たとえば次のような契約体系がありますが、おおむねすべて通信費用との兼ね合いになります。簡単に言えば、いつでも使える速い通信を契約したければ高くなるということです。

▼写真1　SIMのサイズ比較

左から標準SIM、マイクロSIM、ナノSIM

・基本サービス
音声のみ、データのみ、または両方
・テクノロジ
LTEのみ、3Gのみ、または両方
・スピード
速度無制限(技術限界値まで)なのか、一定のスピードを上限とするのか
・データ量
使えるデータ量は無制限なのか、一定量なのか
・時間帯
どの時間帯が使えるのか。上記スピードと組み合わせて、どの時間帯ならどのスピードで使えるのか、というSIMもある
・その他サービス
テザリング可能かどうか、国際ローミング可能かどうか、NFC対応かどうか、SMS受信可能かどうかなど

キャリアのSIM

たとえばキャリアが通常携帯と一緒に販売しているSIMはどうなっているかというと、上記の分類では音声＋データ、3G＋LTE、スピード無制限、データ量は7GBまで、時間帯無制限、テザリング可、など、ほとんど全部入りが多いようです。一方、MVNO(Mobile Virtual Network Operator：仮想移動体サービス事業者)のSIMではそれでは差別化になりませんので、データのみ速度300kbpsまでで月々980円、などとしています(後述)。

どう制御しているのか

契約体系による機能や性能は、たとえば接続できるアクセスポイント(APN)を使い分けることなどによって制御しています。たとえばNTTドコモであればspモード、iモード、moperaなどがあり、スピードや閲覧可能コンテンツなどが異なります。また、同じキャリアのSIMでも世代によってバージョンが異なる場合もあります。たとえばNTTドコモでは少なくとも5つのバージョンが発行されているようで、国際ローミングが可能なものや無線通信(OTA)により電話番号の書き換えができるものなど、いろいろなバリエーションがあるようです。

契約形態

また、純粋に契約上の分類もあります。

・複数電話番号
1つのSIMに複数の電話番号を搭載するかどうか
・プリペイド／ポストペイド
先払い(プリペイド、契約なし)か後払い(ポストペイド、契約あり)か
・期間
何日間有効か
・枚数
1つの契約で複数のSIMを提供するサービスもある

こうして、さまざまなハードウェア(SIM)とさまざまな契約形態を取ることにより、時代とユーザに合わせたSIMを提供していることになります。

サービス提供者

これらどのような契約体系を用意するかは、キャリアやMVNOのビジネス形態によります。キャリアは保有する周波数(たとえば20MHzなど)の中にできるだけ多くのユーザを収容しなければなりません。しかし全員が速度無制限、時間もおかまいなしにこの周波数帯を使って通信してしまったら、すぐに溢れてしまいます。ですので、キャリアやMVNOはユーザごとに細かくサービスを分類し、それぞれに応じたコストをかけて通信帯域(つまりSIM)を販売しているわけです。

MVNO

また、SIMをMVNOから入手するという手も

あります。MVNOでは、それぞれユーザの嗜好に合わせたきめ細かいサービスを提供しているのが特徴です。たとえばテクノロジやスピードを押さえて格安で提供するなどです。これは、一般商品でいう小売店に似ています。キャリアの持つ周波数帯域の利用権の一部をごそっと買い取り、それを小売りにして(時間や帯域に小分けして)ユーザに使わせているわけです。キャリアは各ユーザの細かい嗜好まではわかりませんので、キャリア(問屋)としてもMVNO(小売り)に帯域を売るのはメリットのあるものなのです。逆に言えば、MVNOとはこのSIMを売るビジネスだと言ってもそう外れではありません。

MVNOのSIM

上記のとおり、MVNOは大手キャリアから帯域を買ってそれを小売りしているので、結果的にはユーザは大手キャリアの周波数を使って通信していることになります。そのため、キャリアでロックのかかった携帯端末であっても、MVNOのSIMが利用可能であったりするわけです。また、最近は大手キャリアでも単体でSIMを販売しているケースもあるようです。

周波数との関係

よく"このSIMはどの周波数に対応しているか"という疑問を聞きますが、SIM自体はユーザのIDを保持しているだけですので、対応周波数はとくに関係ありません。ただ、そのIDがキャリア上でどのように契約されているか、となると、上述のとおりそれはキャリアのビジネス形態によるので、確かにその場合は「このSIMはどの周波数に対応している」ということもできなくはありません。いずれにせよ、周波数はSIMとの関係ではなく、それに紐づけられた契約との関係になるわけです。

SIMと携帯端末の関係

さて次に、SIMと携帯端末の関係について触れておきましょう。

SIMフリー端末とは

SIMフリー端末の説明は今さら必要ないかもしれません[注2]。逆に言えば、キャリアによりSIMロックされていない端末、ということになります。

SIMフリー端末の導入

SIMと端末を個別に導入するメリットは何と言っても通信費の削減です。ただこれは、すでにSIMフリー端末を持っているか格安で入手できる場合だけコストメリットがあります。通信費を抑えても端末コストがバカ高いなら、キャリアの普通の端末を普通の通信で購入するのと同じですね。

また、端末やSIMによっては、「端末では通信できるのにPCなどからのテザリングができない」などということもあるようです。購入前に事前によく調べておく必要があるでしょう。

テザリングとは

テザー(tether)とはもともと「ロープでつな

注2) このSIMフリーという言葉は筆者はあまり好きではありません。シュガーフリー、鉛フリーのように、フリーという言葉は「(本来あってはならないものが)ない」という意味があるので、SIMフリーというとそれは「自由にSIMが使える」という意味ではなく「SIMが入ってない(必要としない)」というような意味合いになるからです。これは和製英語ですね。英語ではSIM Unlocked端末などと言うのが一般的です。

COLUMN SIMは貸与

SIMは通常、キャリア(Mobile Network Operator)から「貸与」の形でユーザに渡されます。そう、SIM自体は本来は利用(契約)を終了したらキャリアに返却しなければならないのです。ただ、単価が安いし再利用も難しいので事実上はそのまま破棄する人も多いでしょう。

COLUMN　SMSとは

SMSについて簡単に触れておきましょう。SMS(Short Message Service)を理解するには、まず携帯電話の歴史では音声とパケットデータの2つが並行して走っていることを理解しなければなりません。

回線交換とパケット交換

「電話機」というものが発明されてからずっと、電話回線は文字通り「回線交換」、つまり実際に(物理的に)線を相手に回して(つないで)接続する方式が取られていました。CS(Circuit Switch)とも呼ばれています。この方式では物理的に線がつながっているので通信は安定しますが、データ(音声など)がないときでもずっとつなぎっ放しですので、リソースがもったいないという欠点もありました。

そこで近年になって発達したのが「パケット交換」(PS：Packet Switch)です。「データ(音声でもほかのデータでも)を細切れ(パケット)にして隣の交換機まで送り、あとはその交換機が次の交換機に送ってくれるのを信頼する」というやり方です(**図4**)。これであればデータがないときはリソースを使わずに済むので効率はいいですが、通信品質は安定しません。いまだにLINEやSkype(パケット交換)の音声が携帯の電話(回線交換)に比べて安定しないのはこのためです。

SMSとパケット系テキストメッセージの違い

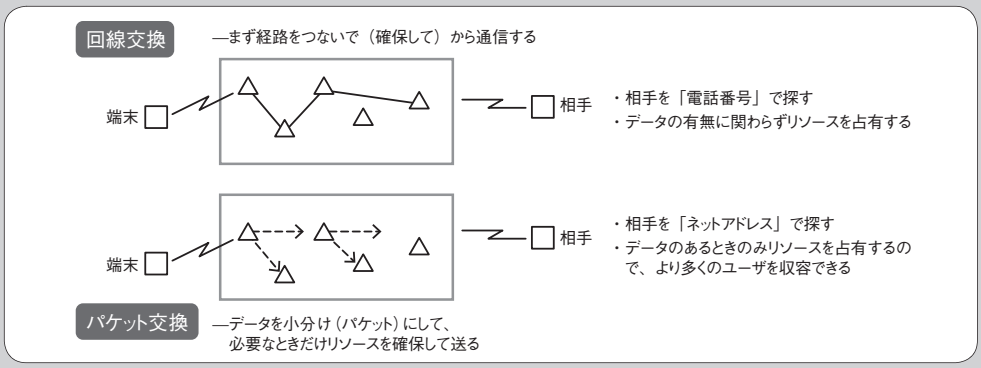
さて、音声(データの一種です)はこの回線交換を、テキストメッセージ(メールなど)はパケット交換を使うのが一般的ですが、このパケット交換があまり発達していない時代(GSMのころでしょうか)にも「テキストメッセージを送りたい。"YES"とか"OK"程度の短いものでいい」という要望がありました。そこで昔の人達は、「回線交換の通信路の隙間に短いテキストメッセージを入れてしまえ」と発想したわけです。これがSMSです。ですから、SMSは回線交換網(つまり電話番号)さえあれば、メールアドレスなどなくても、通信ができるわけです。SMSでは150バイト前後(日本語だと全角ですので70文字前後)までのやりとりができます。

SMSとアンテナバー

SMSは上記のように回線交換を使いますが、端末によっては通信品質(アンテナバー)の制御をするのにこの回線交換を使って実現したりしていますので、パケット専用(回線交換なし、つまりSMSもなし)のSIMを入れるとアンテナバーが立たなかったりすることがあります。また、回線交換契約がない(つまりSMSもない)SIMを用いると、携帯によっては回線交換網への登録処理(レジストレーション)に失敗してこれを繰り返すので、本来使わなくていいプロセッサパワーを使って電池の持ちが悪くなったりします。さらに、iPhoneやAndroidアプリの中にはこのSMSを使ってアクティベートしたり制御したりするものがあるので、SMSがないとこれらができなくなってし まいます。

このような理由から、パケットを中心とした端末利用であっても、SMSの有無は意外と重要な要素になるのです。MVNOによっては、今あるSIMに対してわざわざ「SMS対応！」と謳ったSIMを出しなおしたりしている理由はそこにあるわけです。

▼図4　回線交換とパケット交換



SMSの料金

パケットし放題のプランなのに、SMSだけ別料金が取られるのはこういった事情があるためです。また国際間のSMSは一通100円程度かかることもあります。パケットとは上記のような背景や根本的なところから違うので、同じテキストメッセージでも注意が必要です。ちなみに日本ではキャリア間のSMSがつい数年前に解禁されましたが、諸外国ではSMSが導入された時点からずっと可能でした。

ぐ」というような意味合いの言葉で、それがPCの世界に入って来てテザリングと呼ばれています。ひと昔前は、PCをUSBケーブル(つまりロープ)などで携帯につないで、これをモデム代わりにしてネットにつなぐのが主流でしたが、今では局所ワイヤレスの技術も進みコストも下がってきたので、WiFiやBluetoothによるテザリングが一般的です(**図5**)。

テザリングのメリットと問題点

このテザリング、ユーザからみると次のようなメリットがあります。

①モデム(モバイルルータなど)を別途持ち歩かなくても携帯経由でネット接続が可能になる
②別機器(別SIM)がないため、契約する通信回線が1つで済む
③WiFiテザリングなどの場合は1つの端末で5から10台のPCやモバイル機器を収容できる

などがあります。ただ、キャリアにとってはこれは大問題です。

①今まで複数端末、つまり複数回線(SIM)を契約してもらえたのが1つに集約されてしまう
②1つの回線(契約)で5から10人も収容できるのでさらに契約および収益が減ってしまう

ですのでキャリアは長らくこのテザリングに及び腰でした。しかし、競争の激化に伴い2年間テザリング無料、などとするキャリアも出てきたため、今では"テザリングがタダでできて当たり前"という感覚になってきています[注3]。

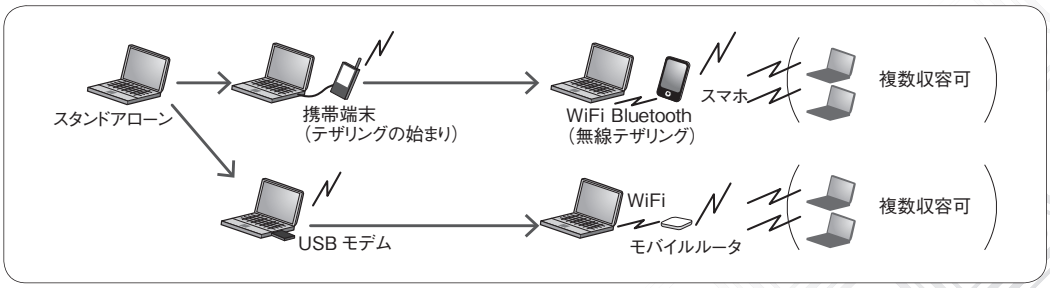
携帯端末のIMEI制限

一部の端末(たとえばiPhoneなど)とSIMの関係はやや特殊です。というのも、これらは3GやLTEといったテクノロジを搭載し、SIMなどを挿入して使うのですが、そのSIMをAndroidなど別の機器に挿入して使おうと思っても使えないことがあるためです。

これは、SIM自体の制限ではなく、キャリアの「SIM契約上の」縛りになります。1つの例では、そのSIMではその端末でしか使えないよう、端末の固有のID(IMEI：International Mobile Equipment ID＝端末識別番号、など)と

注3) と言っても有料だったり、速度制限や通信量制限などがあったりします。

▼図5　PCの広域通信の遷移



紐づけて端末とSIMを販売します。これにより、そのSIMはその端末専用の契約となるわけです。これは、その端末の通信プランを用いてほかの通信機器で想定外の使われ方をされるのを防ぐ目的があるのだと思われます。せっかくSIMという技術を導入しているのに事実上「電話番号埋め込み」の電話になってしまうのですね。ちょっともったいない気がします。

SIMの技術要素

ここで、SIMの技術要素についても少し触れておきましょう。SIMは6～8本の接点(I/O)と、小さなマイクロプロセッサと、必要最小限のメモリで構成されています。つまり、SIMはそれ自体でコンピュータなわけです。

ピンアウト

SIMの形状や規格にもよりますが、SIMの金色をした部分には6～8本の接点があり、おおむね以下の用途に使用されています。

・電源(Vcc/Vpp/GND)
・I/O(データの入出力)
・クロック(内部機器を制御するタイミング信号)
・リセット(内部機器を制御するリセット信号)

これらが内部のマイクロプロセッサや不揮発メモリに接続され、外部(携帯端末)からのコマンドに応じて必要なデータの入出力をしているわけです(**図6**)。この金色の部分以外のところはただのプラスチックケースですので、標準SIMからマイクロSIMなどに(無理矢理)SIMカッターなどで切り欠きして形状変更することできます[注4]。しかし、あまり無茶をすると精密機械ですので壊れてしまいます。注意して行いましょう(貸与なので、本来はカットなどを勝手にしてはいけません)。またナノSIMからマイクロSIMや標準SIMにするゲタのようなものも売られていますが(**写真2**)、抜けなくなるなどのトラブルも多いようです。

セキュリティ機能

またSIMは前述のようにユーザのIDを保持した極めて重要なデバイスです。これを守るため、SIMには次のようなセキュリティ機能が搭載されています。

・認証機能(Authentication)
そのIDが確かに本人であることを証明する機能
・秘匿機能(Confidentiality)
やりとりするIDを暗号鍵により外部から隠蔽する機能
・完全性保持機能(Integrity)
IDが完全な状態であることを保証する機能

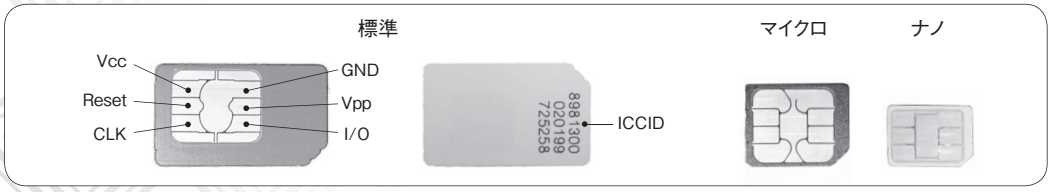
これらはSIM内部に保持した暗号鍵と暗号アルゴリズムにより実現されています。

海外事情

海外に出張や旅行に行ってまず困るのが、携帯端末の利用、とくにその料金ですね。一般に、現状では海外で自国の携帯端末を使うと(ローミングという)べらぼうに高い請求がかけられま

注4) SIMカッターなる製品も売られていたりします。

▼図6　SIMの構造



す。これは技術的に難しいからとかではなく、単に国際キャリア間の取り決めによるものと考えて良いでしょう。

海外ローミングのしくみ

では海外でローミングを行うとどうなるか、SIMおよびユーザIDの観点で見てみましょう。まず空港に降り立って電源を入れると、携帯端末はSIMの固有ID(IMSI)を用いてネットワーク(キャリア)に「私は今ここにいます」という信号を投げます。しかし、その「私」はその国では登録されていないため、当該キャリアは他国に問い合わせをし、ようやくその「私」は日本登録であることを知るわけです。そこで当該キャリアはこれはローミングだと判断し、端末にローミングマークを出させると同時に高額な請求の対象として通信料をカウントし始めるわけです。

海外で安く携帯端末を使う

ではどうすればこの高額な請求を避けられるかというと、それはその国で登録されたSIMを用いて通信をすれば良いだけです。つまり、その国のキャリアのSIMを購入することになります。前述のとおりSIMはヨーロッパで発達した文化ですので、ヨーロッパでは比較的SIMの入手は簡単ですが、現時点では米国ではSIM単体での入手はまだまだ難しいようです。数年前ですが、筆者も米国のAT&Tなどでこれを購入しようと思ったことがありましたが、断念してしまいました。

日本で海外のSIMを入手する

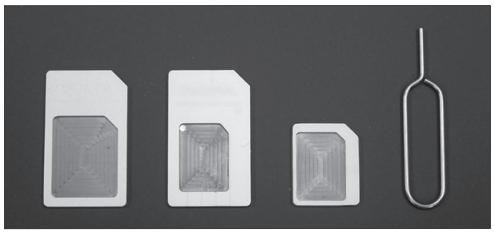
このように、現地でのSIMの入手は困難な場合がありますが、それを回避する方法の1つは、日本で海外SIMをレンタルしてから渡航するという方法です[注5]。ただ、SIMだけレンタルするのはAPN設定や渡航国の周波数を搭載した端末の入手など何かと一般人には難しいので、すでにそれらが設定されているモバイルルータと一緒にレンタルできるようになっており、これでユーザは現地についた瞬間にモバイルデータ通信が行えます(**写真3**)。非常に便利ですね。

同じく海外携帯(音声電話)のレンタルもありますが、こちらは電話番号まで変わってしまうので、いつもの番号で電話を受けることができません。音声電話のレンタルがあまり流行らなかったのはそのためでしょうか。

最後に

以上、今回はSIMの比較的底辺な知識を記述させていただきました。詳しく知りたい方は、3GPPドキュメントなどを参考にされると良いかと思います。今までいろいろな雑誌で「どこのSIMはどういう使い勝手でどう安い」という記事をご覧になって「そもそもSIMってなんだよ」と思われた方にとって、今回のSIMの歴史や背景、周辺知識などが皆さんの理解の助けになっていることを願っています。SD

▼写真2　SIMのサイズ変換用アダプタ



▼写真3　レンタルルータの例(MiFi)

Novatel Wireless社製

注5) 成田空港などでレンタルルータのビジネスをよく見かけるようになりました。

短期集中連載

Mac as a desktop Service -MaaS- !?

さらに踏み込む、Mac OS Xと仮想デスクトップ #2

複数のOS環境を必要とするMac使いのエンジニアにとって、仮想デスクトップ環境をMacに構築することはもはやあたりまえのことのようです。筆者もその一人ですが、日常的に使っているうちにOS間を行き来するオペレーションに煩わしさを感じるようになりました。この短期連載では、筆者がこの煩わしさから解放されるために行った、普通とはちょっと違ったアプローチをご提案したいと思います。

後藤 大地（ごとう だいち）（有）オングス 代表取締役

前回のおさらい

この短期連載では、**図1&2**のような仮想デスクトップ環境をMac上に構築することを目的としています。先月号の第1回目では、このモデルの全体像と、ファイルを共有するための土台作りとしてNFSを使ったファイルシステムの構築方法を解説しました。

第2回となる今回は、アプリケーションをシームレスに運用するために必要な、ssh(1)とX Window Systemを使ったMacとUNIX系オペレーティングシステム（OS）との接続のしくみを解説します。

環境構築[2] CUIもGUIもシームレスに連続させる

MacとUNIX系OSとのやり取りには、ターミナルで作業するものに関してはssh(1)[注1]、UNIX系OSとMac OS Xとでシステムクリップボードを共有するためにはX Window Systemを、UNIX系OSでXアプリケーションを実行する場合にもX Window Systemを利用します。

ssh(1)でUNIX系OSにログインする作業に関しては多くの技術者が日常的に実施していることでしょう。ここでは次のポイントを押さえることで、ssh(1)を使って安全かつ利便性を持った状態を作り上げます。

- **公開鍵認証によるログインのみを許可し、それ以外のログインは許可しない**
- **秘密鍵はハードウェアに対して1つ作成するといった運用をする**
- **Mac OS XとUNIX系OSは相互にログインできるようする**
- **認証エージェント（ssh-agent(1)など）を活用するなどしてパスワード入力の手間を減らす**
- **~/.ssh/configを適切に記述するなどしてコマンド入力の手間を減らす**

ssh(1)を多用する方でも、毎回オプションなどをすべて入力している方をよく見かけます。固定化された作業環境であれば~/.ssh/configに設置をまとめておいたり、シェルのエイリアスの機能を活用するなどして入力の手間を省いたほうがよいでしょう。コンテキストを加味した処理をする必要がある場合には、ラッパスクリプトを開発します。環境融合のためのシェルスクリプトに関しては次の第3回でいくつか解説します。

2回目となる今回は、ssh(1)とX Window Systemについて紹介するとともに、状況に応じて技術を選択する指針などを説明します。

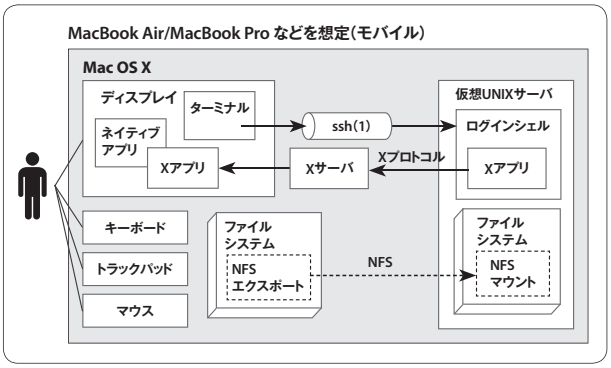
注1）各コマンドのうしろについている括弧書きの数字は、manコマンドで見ることができるマニュアルに記載されている章番号を表しています。

クリップボード共有で使うX Windows System

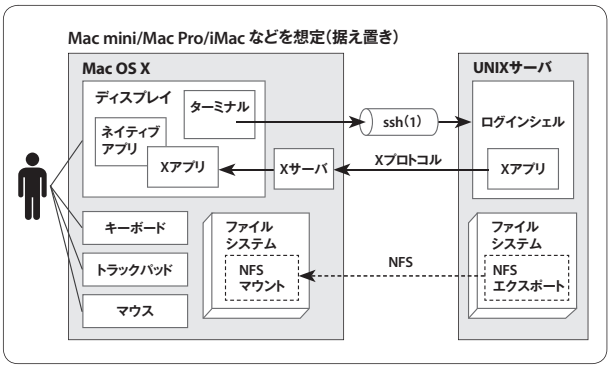
OS間の連帯でもっとも重要になってくるのがシステムクリップボードの共有です。GUIはMac OS Xに固定するという方針をとっていたとしても、マウスを使うよりもコマンド(pbcopy(1)、pbpaste(1)、xsel(1)など)でシステムクリップボードを操作したほうが便利ですので、やはりシステムクリップボードを共有する機能が必要です。この機能にはX Window Systemを利用できます(詳しくは後述)。Xアプリケーションを使わない場合でも、システムクリップボードを共有する機能としてX Window Systemを利用できます。

デスクトップ(ワークステーション)系のLinuxディストリビューションしか使ったことがないと体感として理解しにくいところがありますが、X Window Systemはネットワーク透過なクライアントサーバモデルのウィンドウシステムです。Xアプリケーションが動作するホストと、実際の画面が描画されるホストは同じホストである必要がありません。この“ネットワーク透過である”という特徴を利用すると、ほかのさまざまなことにも応用が効きますので、使ったことがないのであれば一度試してみるとよいでしょう。

▼図1　MacBook AirやMacBook Proのようなモバイル環境でのモデル



▼図2　Mac ProやiMac、Mac miniのような据え置き環境でのモデル



リモートログイン「ssh」

それでは、もうちょっと具体的に説明していきます。まず、UNIX系OSとMacの間はssh(1)経由で相互に行き来できるようにします。このとき、公開鍵暗号方式でのログインのみを許可するように設定します。UNIX系OSが備えているログイン認証でのログインは禁止するとともに、rootアカウントでのログインも禁止します。sshd(8)の設定としては**リスト1**のようになります。

Linuxディストリビューションによっては初期設定が緩い状態になっているものがあります。このあたりの設定は確認しておきましょう。次に、利用するアカウントに紐付ける公開鍵と秘密鍵のペアを生成します。生成にはssh-keygen(1)を使います。

図3のように実行すると次のファイルが作成されます。

▼リスト1　公開鍵暗号方式でのログインのみを許可

```
RSAAuthentication yes
PubkeyAuthentication yes
PermitRootLogin no
UsePAM no
ChallengeResponseAuthentication no
```

▼図3　ssh-keygen(1)で秘密鍵と公開鍵のペアを作成

```
% ssh-keygen
Generating public/private rsa key pair.
Enter file in which to save the key (/Users/daichi/.ssh/id_rsa):
Created directory '/Users/daichi/.ssh'.
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /Users/daichi/.ssh/id_rsa.
Your public key has been saved in /Users/daichi/.ssh/id_rsa.pub.
The key fingerprint is:
74:16:c0:54:e3:15:4c:4e:de:50:9f:b2:c5:60:a3:d3 daichi@example.co.jp
The key's randomart image is:
+--[ RSA 2048]----+
|       ooo+o**.  |
|        .. B=o+..|
|        . +ooE.+.|
|       . o  . +  |
|        S    .   |
|                 |
|                 |
|                 |
|                 |
+-----------------+
%
```

▼図4　作成した公開鍵をログイン先の~/.ssh/authorized_keysファイルへ追加（パーミッションに注意）

```
% cat /pathto/id_rsa.pub 〉〉 ~/.ssh/authorized_keys
% chown 600 ~/.ssh/authorized_keys
```

- **秘密鍵……~/.ssh/id_rsa**
- **公開鍵……~/.ssh/id_rsa.pub**

公開鍵のほうを、ログインする先ホストの~/.ssh/authorized_keysファイルに追加します(**図4**)。このファイルはパーミッションを適切に設定しないと機能しませんので注意してください。

次にログイン先ホストで使用する秘密鍵ファイルを~/.ssh/configに設定しておきます(**リスト2**)。この設定を相互に実施することで、お互いにシームレスにログインできるようになります。秘密鍵にパスフレーズを設定していない場合にはログイン時にパスワードは聞かれませんし、パスフレーズを設定した場合でも認証エージェントに登録するなどして(ssh-add(1))利便性を高めることができます。

公開鍵秘密鍵ペアの作成の指針は、1つのマシンに対して1つの公開鍵秘密鍵ペアを生成するというのがシンプルでわかりやすい指針です。仮想ゲストごとに細かくペアを生成してもよいのですが、厳格にすればするほど利便性が低下します。逆に、すべてのマシンで同じ鍵を共有するのはやめたほうがよいでしょう。1マシンに対して1鍵を基本とし、必要に応じて複数の鍵を使うようにするというのが無難なところだと思います。

Mac OS Xでは「システム環境設定」の「共有」で「リモートログイン」を選択し、ここでログインするユーザを設定します(**図5**)。

ウィンドウを共有「X Window System」

現在のUNIX系OSで広く採用されているウィンドウシステムがX Window Systemです。X Window Systemはウィンドウの描画、ウィンドウの移動、マウスやキーボードなどの入力などを扱うプロトコル(以降、Xプロトコル)を定

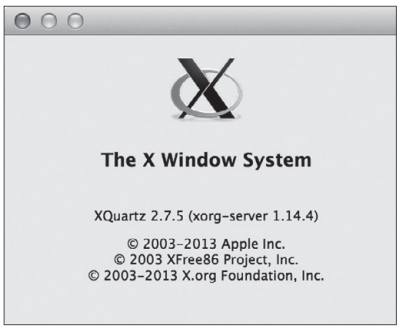
▼リスト2　~/.ssh/configファイルにログイン設定を追加

```
Host parancell
        Hostname        ログイン先IPアドレスやホスト名
        IdentityFile    ~/.ssh/id_rsa
        ForwardAgent    yes
```

▼図5　Mac OS X側でsshd(8)を起動

▼図6　XQuartzをインストール

▼図7　XQuartzを実行



めています。このシステムはクライアントサーバモデルのウィンドウシステムで、GUIアプリケーションがXクライアント、ディスプレイに描画するサーバがXサーバとなります。XサーバとXクライアント(GUIアプリケーションなど)は同じマシンで動作する必要はなく、別々のホストで動作し、ネットワーク経由で連動できるという特徴があります。X Window Systemの実装系の1つがX.Orgです。X.OrgはLinuxディストリビューションや*BSD系ディストリビューションで広く採用されています。

Xプロトコルはネットワークを経由して利用できますので、今回構築するようなシステムとは相性がよいと言えます。GUIアプリケーションはMac、CUIのツールはターミナルからゲストにログインして利用するとしても、UNIX系OSでしか提供されていないGUIアプリケーションを使いたいことがあります。こういった場合にはXプロトコルを使ってMac側にUNIX系OSのGUIアプリケーションを表示させて利用するといったことができます。

また、GUIアプリケーションを利用しないとしても、システムクリップボードを共有する目的でXのしくみを利用できますので、使えるようにしておくとなにかと便利です。

Mac OS X Mavericksでは「XQuartz[注2]」というX Window Systemの実装系を利用します。**図6**のようにインストーラを使ってインストールします。インストール後、いったんログアウトしてから再度ログインした後で利用できるようになります(**図7**)。

XQuartzの設定

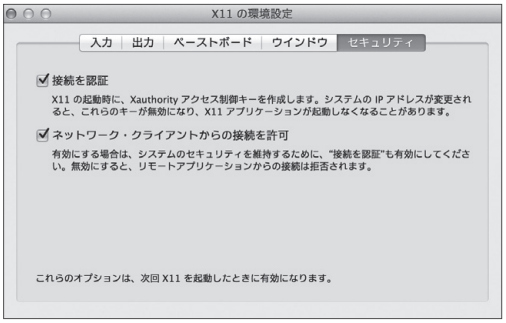
今回の話ではMacからではなくゲストやUNIXサーバからXQuartzにアクセスしてXアプリケーションを利用したいので、まず設定の

注2)　http://xquartz.macosforge.org/landing/

変更を行います。X11の「環境設定」から「セキュリティ」を選択し、「ネットワーク・クライアントからの接続を許可」にチェックを入れて、XQuartzを再起動します(**図8**)。

この設定を変更する前では、XQuartzはXサーバのデフォルトポートである6000番を閉じています。設定を変更してXQuartzを再起動した後は、**図9**のように6000番ポート、ないしは6001番ポートやそれ以降のポートにアクセスできるようになります。

▼図8 「ネットワーク・クライアントからの接続を許可」にチェックを入れてからXQuartzを再起動



Xサーバへのアクセス許可情報を設定

Xサーバにアクセスするには、Xサーバの許可情報をXクライアントが持つ必要があります。Xサーバの許可情報は**図10**のようにxauth(1)コマンドで確認できます。この場合ですと「192.168.1.38:0」がディスプレイ名、「MIT-MAGIC-COOKIE-1」がプロトコル名、「e144eef320ed1e7fdc4bc1086a3dbf7f」が16進数表記の鍵となります。この許可情報はXサーバが起動するときに生成され、ユーザの~/.Xauthorityファイルに保存されます。

このXサーバにアクセスしたいホストで、**図11**のようにxauth(1)コマンドを使ってこの許可情報を登録します。こうすることで192.168.1.38:0[注3]に対してXクライアント(GUIアプリケーションなど)が接続できるようになります。許可情報を登録したホストでXサーバホストと

注3) これは192.168.1.38の6000番ポートという意味です。192.168.1.38:1であれば192.168.1.38の6001番ポートという意味になります。

▼図9 オプションを変更すると6000番ポートにアクセスできるようになる

```
% nmap 192.168.185.1

Starting Nmap 6.40 ( http://nmap.org ) at 2014-01-03 23:07 JST
Nmap scan report for 192.168.185.1
Host is up (0.00013s latency).
Not shown: 964 closed ports, 31 filtered ports
PORT     STATE SERVICE
22/tcp   open  ssh
111/tcp  open  rpcbind
6000/tcp open  X11  ←新しく6000番ポートが開いている

Nmap done: 1 IP address (1 host up) scanned in 5.40 seconds
%
```

▼図10 Xサーバの許可情報

```
% xauth list
192.168.1.38:0  MIT-MAGIC-COOKIE-1  e144eef320ed1e7fdc4bc1086a3dbf7f
%
```

▼図11 Xアプリケーションが XQuartzにアクセスできるように、クライアント側でディスプレイ名と接続用のプロトコルおよび鍵を登録

```
% xauth add 192.168.1.38:0 MIT-MAGIC-COOKIE-1 e144eef320ed1e7fdc4bc1086a3dbf7f
```

▼図12　登録したディスプレイ名、プロトコル、鍵を確認

```
% xauth list
192.168.1.38:0  MIT-MAGIC-COOKIE-1  e144eef320ed1e7fdc4bc1086a3dbf7f
%
```

▼図13　Xアプリケーションに接続先を知らせるために環境変数DISPLAYを設定してから、Xアプリケーションを実行

```
% export DISPLAY=192.168.1.38:0
% xclock &
%
```

同じようにxauth(1)を実行して、エントリが追加されたことを確認します(**図12**)。

図13のように登録したディスプレイ名を環境変数DISPLAYに設定してから、Xアプリケーション(xclock(1))を実行してみましょう。何もウィンドウが表示されていないMac OS X Mavericksの画面に、xclock(1)が表示されるようになりました(**図14**)。

X Window Systemを使うと、このようにネットワークを経由してアプリケーションだけをMac側に表示させるといったことを実施できます。

ssh接続をする際にxauth認証を使わない設定

ssh(1)でログインしているホストの場合、xauth(1)で許可情報を登録せずとも、ssh(1)が提供しているX11フォワーディングという機能を使ってXアプリケーションをXサーバに接続させることもできます。Xサーバが動作しているホストで**リスト3**のような設定を~/.ssh/configファイルに追加します。

Xサーバが動作しているマシンから、この機能(ForwardX11およびForwardX11Trusted)を有効にした状態で別のホストへログインしま

▼リスト3　ssh(1)のX11フォワーディング機能を使用する

```
Host virtualhost
        Hostname                192.168.185.50
        IdentityFile            ~/.ssh/id_rsa
        ForwardAgent            yes
        ForwardX11              yes
        ForwardX11Trusted       yes
```

▼図14　xclock(1)が実行された後のMac OS X Mavericks

す。ログイン先ホストでXアプリケーションを実行すると、ログイン元のホストでXアプリケーションが表示されるようになります。

sshのX11フォワーディングとX

ssh(1)のX11フォワーディングを使うか、それともXもともとの機能を使うか、どちらがよいかはケースバイケースです。信頼できないネットワークを経由する場合にはssh(1)を使う必要があるでしょうし、信頼できるネットワークを経由する場合にはパフォーマンスに応じてどちらを使うか選ぶことになります。

ここでは2つの異なるパターンを示します。ひとつ目はMacホストと仮想環境の間で、X(**表1**)とssh(1) X11フォワーディング(**表2**)の双方でXアプリケーションを実行した場合の例です。Mac側がXサーバになっています。ssh(1)は通信負荷が高くなりますので、ssh(1) X11フォワーディングのほうが負荷が

高そうですが、ここではXをそのまま使ったほうがCPU負荷が高くなっています(**表3**)。これはVMwareのNAT変換処理が高負荷になり、全体の負荷を引き上げているためです。**図15**ではグラフの左側がX、右側がssh(1) X11フォワーディングの結果です。

同じことを、今度は仮想環境ではなく物理マシンとMacマシンの間で実施します(**表4〜5、図16**)。負荷が低かったので、こちらではXアプリケーション(vlc)を2つ起動してあります。この結果は先ほどとは逆で、ssh(1)の通信負荷が高くなり、Xをそのまま使ったほうが負荷が

▼表1　Xプロトコルを直接使った場合

Mac OS X プロセス	CPU使用割合	ゲストOS プロセス	CPU使用割合
vmware-vmx	100%	vlc	60%
vmnet-natd	78%		
kernel_task	36%		
X11.bin	45%		

▼表2　ssh X11フォワーディングを使った場合

Mac OS X プロセス	CPU 使用割合	ゲストOS プロセス	CPU 使用割合
vmware-vmx	100%	sshd	87%
ssh	60%	vlc	10%
kernel_task	20%		
X11.bin	13%		

▼表3　Macマシン CPU平均使用率比較

Xプロトコル CPU平均使用率	ssh X11フォワーディング CPU平均使用率
70%	54%

▼図15　仮想環境とホスト間のX：Xプロトコルとssh X11フォワーディング

column 1　xauth(1)による許可情報の登録やXサーバ起動の自動化

これまで紹介してきたような機能を、人間が毎回手で入力して処理していては本末転倒です。自動化できる処理はスクリプトとして記述して自動化し、ユーザは本来しようとしていた作業に注力できるようにします。その要となるコマンドの1つがssh(1)です。ssh(1)コマンドはホストを指定して、そのホストでさらにコマンドを実行させるといったことができます(**図A**)。この機能を使うことで融合対象のUNIX系OSをローカルホストのように扱うことができます。最初にssh(1)のネットワークを構築するのは、こうした作業をするための布石です。

xauth(1)による許可情報の登録ですが、ssh(1)とxauth(1)の機能を連携させることでコマンド一発で登録できます。たとえば**図B**のようなコマンドが実行されるように仕込みます。

XQuartzの起動なども同じ要領で実施できます。コンテキストを加味して、事前処理やトンネリング処理を実施するようなssh(1)ラッパースクリプトを作成するなどしておくと、このあたりの処理を自動化させることができます。このあたりについては次回にいくつか例をあげて解説します。

▼図A　ssh(1)によるリモートホストでのコマンドの実行

```
ssh 対象のホスト そのホストで実行するコマンド
```

▼図B　xauth(1)を使った許可情報の登録

```
/opt/X11/bin/xauth extract - 192.168.1.38:0 | /usr/bin/ssh 対象のホスト xauth merge -
```

低くなっています(**表6**)。

このように利用するハードウェアやネットワーク、構成によってシステム全体の負荷は大きく変わります。このため、一概にどちらがよいといった判断はできず、どの操作においてどちらが負荷が低いか測定して、適したほうを使用すればよい、といったことになります。

なお同様の作業を、今度はMacのネイティブアプリ(VLC.app)で実施すると、負荷はもっと低くなります(**表7**)。**図17**のグラフでは左側が物理ホストとMacマシン間をssh(1) X11フォワーディングしたときの負荷、右側がネイティブアプリ(VLC.app)を動作させたときの負荷です。物理マシンの負荷も考えると、この手のアプリケーションはネイティブアプリケーションをそのホストで実行したほうが効率がよいことがわかります。

第2回のまとめ

今回はMacとUNIX系OSを融合させる基礎技術として、ssh(1)とX Window Systemを紹介しました。X Window SystemはUNIX系OSを扱う多くの技術者にとっては慣れ親しんだ技術ですが、若手の技術者は普段は意識することのない技術ではないでしょうか。最近はssh(1)でログインすることなく利用できるWebアプリケーション系のサービスも増えていますので、ssh(1)を使わない若手技術者も増えているように思います。

基本的な内容でしかありませんけれども、こうした基本部分を押さえておくことで、細かいストレスに対処できる気の利いたシステムを構築できます。短期連載最後となる次回では、実際にそうしたシステムを構築するための具体的な設定やスクリプトの組み上げ方などを説明します。 SD

▼図16　物理サーバとMacBook Pro間のX：Xプロトコルとssh X11フォワーディング

▼表4　Xプロトコルを直接使った場合

Mac OS Xプロセス	CPU使用割合	UNIXサーバプロセス	CPU使用割合
kernel_task	40%	vlc	20%
X11.bin	25%	vlc	16%

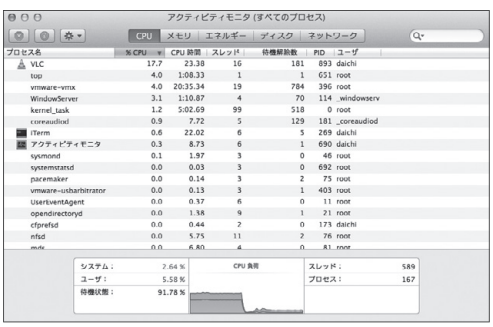
▼表5　ssh X11フォワーディングを使った場合

Mac OS Xプロセス	CPU使用割合	UNIXサーバプロセス	CPU使用割合
ssh	78%	sshd	31%
ssh	57%	sshd	18%
kernel_task	45%	vlc	17%
X11.bin	29%	vlc	14%

▼表6　Macマシン CPU平均使用率比較

Xプロトコル CPU平均使用率	ssh X11フォワーディング CPU平均使用率
29%	64%

▼図17　Mac OS Xネイティブアプリケーションを実行した場合



▼表7　Macマシン CPU平均使用率

CPU平均使用率
18%

シェルスクリプトではじめるAWS入門

新連載

――ゼロから初めるAWS API

第1回　AWS API事始め

Writer　波田野 裕一（はたの ひろかず）／運用設計ラボ　operation@office.operation-lab.co.jp

AWS APIとは

世界最大のクラウドプラットフォームサービスであるAmazon Web Services（以下、AWS）は、2006年のサービス開始からほぼ8年となる現在（2014年2月時点）においても、クラウドコンピューティングサービスにおける先駆者として進化を続けています。

このAWSをはじめとするクラウドプラットフォームサービスは、次の3つの「感覚」に大きな変化をもたらし、従来あった「システム構築の常識」を大きく変えてしまったことから、多くのユーザ企業から注目を集め、急速にその利用者を増やしてきています。

1.「コスト感覚」の変化

従来のオンプレミス環境（On-Premise：会社などの構内に設置するシステム環境）では必要だった初期投資のための資金、減価償却や資産除却などの資産管理の工数が不要になったうえ、稼動中の余剰リソースの即時廃棄によるコスト圧縮がいつでもできるようになりました。

2.「時間感覚」の変化

オンプレミス環境では月単位で想定する必要があったリソース調達やシステム構築におけるリードタイムが、日単位もしくは時間単位にまで短期化しました。また、システムのライフサイクルもオンプレミス環境では年単位で考えていたものが、プロジェクトやサービスの要求に合わせて四半期単位から週単位など極めて短いものとなってきています（たとえば、検証環境用途であれば日単位でライフサイクルが終了するものもあります）。

3.「設計感覚」の変化

オンプレミス環境では調達、構築におけるキャッシュアウトやその工数、さらには事業部門への費用配賦に大きく影響するためにシステムのサイジング設計が重要視されていましたが、稼動開始後のリソース調整を前提としたスケーラビリティ確保型のシステム設計が重視されるようになり、加えてインフラストラクチャレイヤーにおいては「修復」よりも「ゼロから即時作りなおし」を前提にした耐障害設計などに意識が変化してきています。

AWSのAPIの原点

AWSの最初のプロダクトであるAmazon Simple Queue Service（Amazon SQS）、Amazon Simple Storage Service（Amazon S3）は、リリース当初はまったく利用ツールが提供されておらず、純粋に「APIだけ提供する[注1]」形でリリースされたそうです。

Amazon社が当初からAWSを「APIを主軸としたプログラマブルなプラットフォーム」として意識していたことがこの点からも垣間見ることができます。

注1）URL http://aws.typepad.com/aws_japan/2013/01/aws-management-console-improvements-tablet-and-mobile-support.html

AWS APIの充実

これらの特徴はクラウドプラットフォームサービスの導入に踏み切るうえで非常に大きな動機となりますが、エンジニアリングの視点から考えた場合、これらのサービスが提供するプロダクトのほぼすべてをAPI(Application Programming Interface)経由で操作できる、つまり「ITインフラストラクチャをプログラマブルにした」という点にも着目すべきでしょう(最近では「Infrastructure As Code」と表現されることが多いようです)。

AWSではほぼすべてのプロダクトにおいてAPIが提供されており、ユーザはAPI経由で各プロダクトのほぼすべてのリソースと機能を操作することができるようになっています。

Webブラウザから操作するおなじみの「AWSマネジメントコンソール[注2]」も、その背後ではAWS APIに対するコールが行われています。

AWS APIを学ぼう

前節で説明した通り、AWSではその提供するプロダクトのほとんどを(意識していなくても)API経由で操作します。AWS各サービスを深く理解して活用するうえで、AWS APIの知識は欠かせないと言ってよいかと思います。

AWSでは新規サービスや新機能のリリースが頻繁に行なわれ、Web操作画面である[AWSマネジメントコンソール]の画面構成が変更されることも珍しくはありません。そのため[AWSマネジメントコンソール]での操作を前提としたドキュメントは陳腐化が早く、残念ながら作成の手間の割に報われないのが現状です。APIベースでAWSに関する知識を蓄えておけば、このような画面変更に惑わされることなく各種作業ができるだけでなく、たとえばAPIを利用した自作のマネジメントコンソールなどを用意することにより、AWSのサービス変更による画面上での影響をコントロールすることも可能になります。

さらに、デプロイ系のプロダクト(AWS Elastic Beanstalk、AWS CloudFormation、AWS OpsWorksなど)のAPIを利用すれば、あらかじめ設計しておいたシステム構成をボタン1つでリリースできるようになり、そもそもマネジメントコンソールでの作業はほとんど不要になるのかもしれません。このように独自のアイデアを基にAWS APIを活用することで、自分達の目的に特化した革新的なITインフラストラクチャをAWS上に実現することさえ可能となるのではないでしょうか。

加えてAWS APIはAWSで利用されているだけではなく、オープンソースのクラウド基盤ソフトであるOpenStackやCloudStackにおいても Amazon ElasticCompute Cloud(Amazon EC2)やAmazon S3互換のAPIが提供されるなど、AWSの外部に対しても影響を与えています。AWS APIの知識は、AWSが提供するサービスを本格的に使いこなしていくうえで役立つだけでなく、他社のITインフラストラクチャや商用のAPIサービスを活用し、さらには将来独自APIを作成するうえでの参考モデルになるなど、決して無駄になることはないでしょう。

AWS APIの全体像

AWSでは30を超えるクラウドコンピューティングサービスのほぼすべてにおいてAPIを提供していますが、これらのAPIを利用するには、2種類のAPI(リクエスト方式)と3種類のデジタル署名方法を理解する必要があります。

注2) URL http://docs.aws.amazon.com/ja_jp/IAM/latest/UserGuide/Using_AWSManagementConsole.html

2種類のAPI

AWS APIには、REST形式のリクエストに対応しているREST APIとクエリ形式のリクエストに対応しているQuery APIの2種類のAPIがあります。

・REST API

REST APIとは、そのすべてのリソースをURI(Uniform Resource Identifier)形式のユニーク(一意的)なアドレスで表現し、特定のリソースに対する操作はそのリソースのURIに対する4種類のHTTPメソッド(GET/POST/PUT/DELETE)により行なうAPIを言います。

・Query API

Query APIとは、APIに対するリクエストをクエリパラメータで記述し、GETメソッドまたはPOSTメソッドにより送信するAPIを言います。

ほとんどのプロダクトはREST APIかQuery APIのどちらか一方だけを提供[注3]していますが、Amazon CloudSearchのように操作内容によって利用できるAPIが異なるものもありますので、注意してください。

3種類の署名方法

AWS APIでは、各リクエストが正規のユーザから送信されたのものであることを認証するために、リクエストメッセージに共通鍵暗号方式によるデジタル署名を添付する必要があります。

2014年2月現在、次の3種類のデジタル署名作成方法(署名バージョン)が提供されています。

- Signature Version 2
- Signature Version 3
- Signature Version 4

AWSの各プロダクトは、1つもしくは複数の署名バージョンに対応しています。複数の署名バージョンが提供されているプロダクトにおいては、最も新しいバージョンの署名を使うことが推奨されています(新しいバージョンのほうが署名の作成手順も複雑になっています)。

AWSのAPI一覧

AWSにおける2種類のAPI方式、3種類の署名バージョンとAWSの主要プロダクトの対応関係を、AWS公式ドキュメントを参考に筆者が表形式で書き起こしました(**表1**)。この表は、AWS APIのAPI方式と署名方式が明記された一覧としては本邦初公開ではないかと思います。AWSの公式情報ではない点に留意のうえ活用ください。

次回は

次回は、AWS APIを実際に利用するうえで必要な知識と環境について解説していきます。

SD

注3) 従来は上記2種類のAPI以外にSOAP APIも提供されていましたが、2014年3月末にサポートを終了する旨の公式アナウンスが出ていますので、本記事では省略します(http://docs.aws.amazon.com/AWSEC2/latest/UserGuide/using-soap-api.html)。

▼表1　AWS API一覧

インフラストラクチャーレイヤ		REST	Query	Signature (v4)	Signature (v3)	Signature (v2)
ネットワーキング	Amazon Route 53	○	N/A	N/A	○	N/A
	Amazon Virtual Private Cloud (Amazon VPC)	N/A	○ (EC2のAPIを利用)	N/A	N/A	○ (EC2のAPIを利用)
	Elastic Load Balancing	N/A	○	○	N/A	△
コンピューティング	Amazon Elastic Compute Cloud (Amazon EC2)	N/A	○	N/A	N/A	○
	Auto Scaling	N/A	○	○	N/A	△
ストレージ	Amazon Simple Storage Service (Amazon S3)	○	N/A	○	N/A	△ (2014年2月以降に開設されるリージョンでは非対応)
	Amazon Glacier	○	N/A	○	N/A	N/A
	AWS Import/Export	N/A	○	N/A	N/A	○
	AWS Storage Gateway	N/A	○	○	N/A	N/A
データベース	Amazon DynamoDB	N/A	○	○	N/A	N/A
	Amazon Relational Database Service (Amazon RDS)	N/A	○	○	N/A	△
	Amazon ElastiCache	N/A	○	N/A	N/A	○
	Amazon Redshift	N/A	○	○	N/A	N/A

アプリケーションサービスレイヤ		REST	Query	Signature (v4)	Signature (v3)	Signature (v2)
メッセージ	Amazon Simple Notification Service (Amazon SNS)	N/A	○	○	N/A	△
	Amazon Simple Queue Service (Amazon SQS)	N/A	○	○	N/A	△
メール配信	Amazon Simple Email Service (Amazon SES)	N/A	○	N/A	○	N/A
ワークフロー	Amazon Simple Workflow (Amazon SWF)	N/A	○	○	N/A	N/A
メディア変換	Amazon Elastic Transcoder	○	N/A	○	N/A	N/A
コンテンツ配信	Amazon CloudFront	○	N/A	○	N/A	N/A
分散処理	Amazon Elastic MapReduce	N/A	○	○	N/A	N/A
データ連携	AWS Data Pipeline	N/A	○	○	N/A	N/A
検索	Amazon CloudSearch	○ (Doc/Search)	○ (Conf)	○	N/A	N/A
ストリーミング	Amazon AppStream	○	N/A	○	N/A	N/A
分析	Amazon Kinesis	N/A	○	○	N/A	N/A

デプロイ/アドミニストレーションレイヤ		REST	Query	Signature (v4)	Signature (v3)	Signature (v2)
モニタリング	Amazon CloudWatch	N/A	○	○	N/A	△
アイデンティティ & アクセス	AWS Identity and Access Management (IAM)	N/A	○	○	N/A	△
	AWS Security Token Service (AWS STS)	N/A	○	○	N/A	△
デプロイ & マネジメント	AWS Elastic Beanstalk	N/A	○	○	N/A	△
	AWS CloudFormation	N/A	○	○	N/A	△
	AWS OpsWorks	N/A	○	○	N/A	N/A
	AWS CloudTrail	N/A	○	○	N/A	N/A
	AWS Support	N/A	○	○	N/A	N/A

○（対応）　△（非推奨）　N/A（非対応）

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スタッフ
及川 卓也　OIKAWA Takuya
Twitter @takoratta
三廻部 大　MIKURUBE Dai
Twitter @dmikurube

第28回 2013年の振り返りと2014年の方針

この3月で東日本大震災発生から4年目を迎えます。節目となるこの時期にHack For Japanは毎年スタッフミーティングを開催し、前年の振り返りとその年の方針を話し合っています。

Hack For Japanでは、東日本大震災発生後の活動開始時より、1年ごとに活動の継続の要否を決定することを方針としております。今回のスタッフミーティングでも「継続ありき」という前提の議論ではなく、活動停止も選択肢に入れ話し合いました。

昨年の活動実績が少ないことなども鑑み、果たして存在意義はあるのかという厳しい視点を持ち議論しましたが、2014年も活動を継続するという結論に達しました。すでに、Hack For Japanブログでもその方針は表明しておりますが、長くHack For Japanの活動を見守っていただいている読者に向けて、この連載の上でも改めて、今後の方針をお伝えしたいと思います。

2013年の活動実績

まず、昨年の活動を振り返ってみましょう。昨年、私たちは以下のような活動を行いました。

教育：東北Tech道場・イトナブとの協力活動

2013年のHack For Japanは「教育」を軸の1つとして活動してきました。被災地そのものを活性化させ、さらに本当に必要とされているものを生み出すには、現地の産業自体が活性化して必要なものを自分たちで作れる・伝えられる環境を作る必要があると考えたからです。その担い手として、新しい人がITを学べる場を作ることが最初の一歩でした。

そこからつながって、2012年11月から継続して開催されている「東北Tech道場[注1]」や、「イトナブ[注2]」との協力で、7月26～28日に開催された石巻ハッカソン[注3]での講師・サポーターとしての活動を行いました。

International Open Data Hackathon Tokyo

2月23日に世界各地の市民がオープンデータを活用するハッカソンイベント「International Open Data Day[注4]」が行われ、その東京会場[注5]を主催しました。東京会場からは、各種の可視化プロジェクト、千代田区の地域情報を自分たちでまとめるLocalwikiや災害後の情報をまとめるダッシュボードなどのプロジェクトが生まれました。

Open Data for Future

10月2日に行われた「Open Data for Future ——開発者の立場からオープンデータを考える会議——[注6]」に共催として参加しました。ここではHack For Japanスタッフの鎌田が「復旧・復興支援データベースAPIハッカソン[注7]」とその経験を元にしたAPI改善提言書をメインとした発表を行いました。

「ITx災害」会議

Hack For Japanはおもに技術・開発に焦点を当てて活動してきましたが、私たち以外にもITを活用

注1 https://sites.google.com/site/tohokudojo/
注2 http://itnav.jp/
注3 http://itnav.jp/archives/228、本連載の第24～25回
注4 http://opendataday.org/、http://odhd13.okfn.jp/
注5 http://tokyo-opendataday.peatix.com/
注6 http://www.mri.co.jp/news/seminar/other/002418.html
注7 http://blog.hack4.jp/2012/06/api.html（2012年6月2日開催）

した復旧・復興支援を行ってきた団体や個人はたくさんいました。それぞれの活動を振り返ってこれからを考え、次のアクションに結びつけるために、そのような人々が協力して「ITx災害」会議[注8]を10月6日に行いました。

会議は、復旧・復興支援にかかわってきた人たちからのスピーチに続く「アンカンファレンス」を主体にして構成されました。ここでは参加者が自分たちで提案したトピックを「被災者自身による情報発信」「ツールをどうするか」「連携」「IT弱者」などにまとめ、グループに分かれて議論を行いました。

また、スピーチとアンカンファレンスの間には、全日本芋煮同好会による「芋煮」の昼食が用意されました。芋煮を食べながら参加者同士で話すことで、新しいつながりも生まれていました。

さらにこの会議の後の10月26日には、減災にかかわるソフトウェアについて実際に役立つ具体的なアプローチを議論する「減災ソフトウェア開発に関わる一日会議[注9]」が行われています。

みなみさんりく復興マップづくり

10月12日にOpenStreetMap Foundationと「みなみさんりく復興マップづくり[注10]」を共催しました。今回のイベントは、南三陸町のOpenStreetMapの情報を実用レベルまで高めること、南三陸の現状を残すこと、地域の住人が地元の魅力を再発見するきっかけを作ることを目的として行われました。

「税金はどこへ行った？」、Open Spendingプロジェクト

「税金はどこへ行った？」は自治体の税金の流れを可視化するプロジェクトです。2012年の6月と7月に行われたオープンデータ活用ハッカソンにて開始されたこのプロジェクトは、入力した年間収入を元に税額が表示され、その税金が1日あたり、どの分野にいくら利用されているかを表示します。元は、イギリスのOpen Knowledge Foundationが開発したWhere Does My Money Go? のソフトウェアをベースにしており、その後もOpen Knowledge Foundationと協力しながら進められています。2013年にも、2月にInternational Open Data Day、7月と12月にSpending Data Partyを開催し、対応都市を増やしました。この活動はHack For Japanが主催ではありませんが、被災地対応ということで、宮城県石巻市版や岩手県釜石市版をHack For Japanス

注8　http://www.itxsaigai.org/、本連載の第26～27回
注9　http://gensai.itxsaigai.org/
注10　http://fukkomap2013oct.peatix.com/

Hack For Japan
エンジニアだからこそできる復興への一歩

タッフが担当しています。

2014年の方針

震災復興を継続的に支援するためのIT開発を支えるコミュニティとして、2011年3月11日の震災直後に活動を開始したHack For Japanは、当初ITによる震災復興が活動の中心でしたが、その後、今後の災害に備えた活動やハッカー文化を浸透させる活動など、復興支援以外にも活動の幅を広げようとしてきました。

具体的には、次のような活動になります。

- 今後の災害に備えるための活動
  - —— IT防災訓練
  - —— 防災・減災支援ツールの開発
- ハッカー文化（ハッカソンなど）を浸透させる活動
  - —— ハッカソン開催ガイド（ハッカソンパッケージ）
- 若年層へのIT教育を支援する活動
- 被災地のエンジニアと東京のエンジニアの交流促進
- NPOやNGOと連携した海外への活動の発信
- アーカイブ（記録）のための活動
- Hack for Japanの活動および実績の可視化

テクノロジーで社会的課題を解決する人のためのコミュニティ作り

このように活動を広げるに際して、活動のコアとなるテーマは「テクノロジーで社会的課題を解決する人のためのコミュニティ作り」としました。これは未曾有の災害と言われる東日本大震災の復興を支援する中で、戸惑い、試行錯誤を続けるスタッフや参加者を支えたのが、やはりテクノロジーであり、コミュニティであったからです。

4年目を迎える2014年もこのテーマを維持し続けようと考えております。

ただし、2013年は、スタッフ自身もHack For Japanが主体となって行った取り組みは正直少なかったと認めざるを得ません。しかし、これは活動していなかったのではなく、Hack For Japanから派生した団体・コミュニティの活動がアクティブになったことにもよるものでした。

たとえば、アプリケーションやサービスを開発することでより良い政府を作るために設立されたCode for JapanはHack For Japanスタッフの関治之が代表としてリードしていますし、昨秋に行われたITx災害会議はHack For Japanスタッフの多くが発起人となり、その後の活動をおもに技術面から支えています。

このCode for JapanやITx災害などに代表され

るように、Hack For Japanスタッフが積極的にかかわったり、Hack For Japanの活動が何らかのきっかけになり始まった取り組みは少なくありません。私たちの活動の影響だけと限定はできませんが、ハッカソンというイベントがここまで市民権を得たのも、Hack For Japanの副次的な成果と言えるでしょう。

原点回帰

このように、さまざまな方々がテクノロジーで社会的課題の解決を図ろうとしていることを鑑み、Hack For Japanは原点に回帰し、災害への対応を考えるITコミュニティとしての活動に集中します。

たとえば、IT教育も活動の1つとして考えていましたが、昨今では各地で若い世代へのプログラミング教育が行われるようになっています。Hack For Japanではそのような方々と連携し、IT教育においては東日本大震災の被災地を中心に活動したいと考えています。具体的には、東北Tech道場であったり、イトナブなどへの協力です。

また、オープンデータの公開や活用に関しても、Code for JapanやOpen Knowledge Foundationなどが積極的に取り組んでいます。Hack For Japanはそれらの活動を積極的にサポートしつつ、被災地支援の観点での取り組みを模索します。

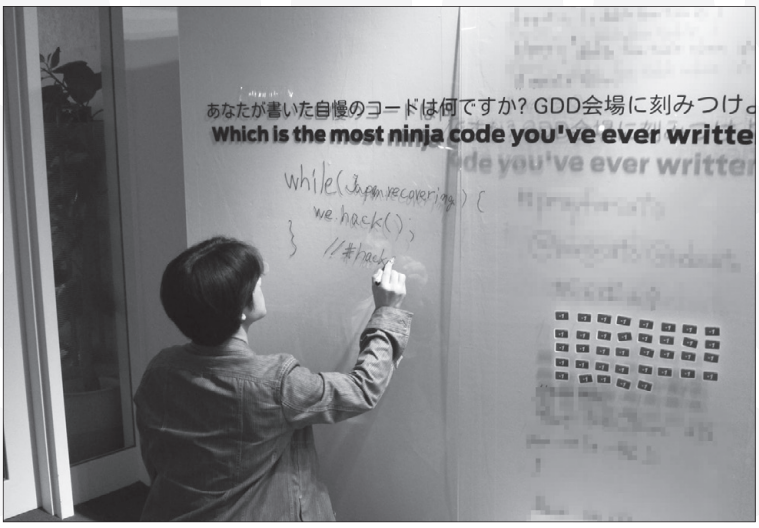
防災・減災についてITでなすべき役割においては、IT×災害がふさわしい場であると考えられますので、Hack For Japanはその活動を技術面で支える役割を担います。

今後の予定

スタッフの話し合いの中では、Hack For Japanは緩やかにつながっているコミュニティとしても十分存在意義があることが再認識されました。たとえば、災害発生時にHack For Japanのコミュニティに連絡すれば、そこでつながり、意義のある活動を迅速に開始できるでしょう。それだけでも、東日本大震災のときよりも状況は良くなっているはずです。

このコミュニティとしてのつながりを維持・強化していくことも2014年のHack For Japanの使命だと感じています。2013年には、ほかの団体やコミュニティの活動を後援などの形でサポートしていながらも、それをHack For Japanコミュニティに共有していなかったため、結果としてHack For Japanの活動が停滞しているように見えてしまったことも否めません。今年はこのコミュニティの活性化もテーマに考えています。勉強会や情報交換会などの開催も検討中です。ブログやTwitter、Facebookでの情報発信にも再度力を入れていく予定です。

2014年も引き続き、Hack For Japanをご支援ください。SD



分散データベース「未来工房」

第10回 Riakはなぜデータをなくさないのか(2)

Writer 上西 康太(うえにし こうた)　Bashoジャパン株式会社　kota@basho.com

先月号では、Riakではなぜデータがなくならないかについて、分散データベースにまつわる整合性と永続化の問題と併せて解説した。Riakがそのような問題をどう解決しているかについて、先月のヒント付きハンドオフに加えて、本稿ではSiblingsとベクタークロックという技術を解説する。

注)本稿は、筆者の意向により常体で表記している。

ヒント付きハンドオフ

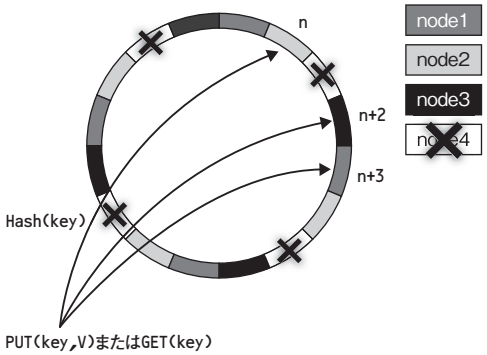
Riakでは、ノードが一時的にアクセスできなくなったときにはヒント付きハンドオフ(Hinted Handoff)によって、**図1**のように、リング上の続くノードにコピーを配置して、とりあえず規定値(通常は3)の数の複製があることを保証するしくみになっていることを先月号で解説した。

一時的にアクセスできなくなる原因としては、スイッチのバッファ溢れやNIC(Network Interface Card)の故障、コンピュータの過負荷などが考えられる[注1]。一時的にアクセスできなくなっていたノードがそのまま復帰した場合、そのノードが持つべきデータは徐々にほかのノードから返却されていく。その様子は**riak-admin transfers**というコマンドで観察できる。

それでは、復帰したあとに別のデータが書き込まれていたり、そのデータが削除されていたらどうなるだろうか？　別のデータが書き込まれた場合に、アクセスできなくなったノードに何も書かれていない場合には問題ないが、それぞれ別のデータが書き込まれた場合は書き込み競合(Write-write conflict)となってしまい、データの整合性を保証できなくなる。また、アクセスできなくなっているノードがいる状態でキーの削除が行われたらどうなるだろうか？　単にエントリを消すだけでは、アクセスできなくなったノードは削除のリクエストを受け取っていないので、復帰したときに削除されたはずのデータが甦ってしまうだろう。

注1) ノード上でRiakだけが動作している場合はこういったことはあまり起きないように設計しているものだが、実際の運用では何が起きるかわからない。

▼図1　障害時の書き込み
node 4はすでに故障とマークされているため、node 1が持っているvnodeへの書き込みが行われる。



Siblings

書き込み競合に対して、Riakでは「とりあえず両方持っておく」という対処をする。これを、同じキーに対する値の兄弟という意味でSibling(シブリング)という[注2]。すべてのSiblingを保持しておいて、読み出し時にすべての

注2) 本来であれば日本語の訳語をあてるべきだが、適切な訳語がないので本稿ではこのままアルファベット表記のままとする。

Siblingから正しい値を決定する。ショッピングカートの例であれば、カート内の商品のUnionを計算すればよい。

仮にSiblingというしくみがなく、上書き可能なしくみでショッピングカートを実現すると書き込み競合は**図2**のように起きる。たとえば、利用者が複数のブラウザや端末(それぞれProcess A、Bとする)から同じショッピングサイトにアクセスして別の製品をカートに入れた場合のアクセスを想定している。リクエストが本当に競合した場合は、このように後から到着したリクエストで上書きが行われてしまう。

それに対して、**図3**のように、Siblingsを両方保持しておくというしくみにしておくと、最終的に**spam**、**ham**という両方の要素が別のSiblingに入っているが、それを結合したリストが結果的にショッピングカートの内容として正しいことをアプリケーション側で決定することができる。

それでは、実際にRiakを叩いてみることで、このようなしくみをどのように利用できるか確かめてみよう。

テスト環境の構築

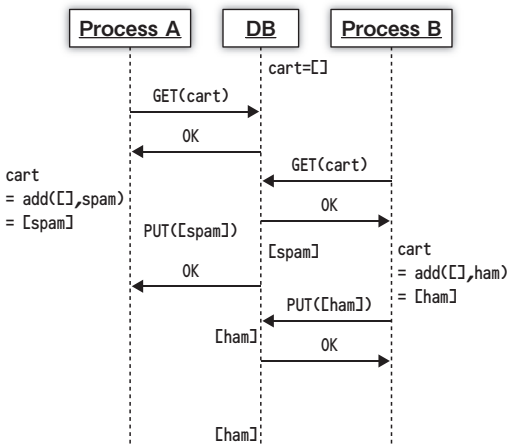
実際に動かしてみるために、簡単に環境を準備する手順を紹介する。Erlang/OTPはhomebrewやaptitudeなどであらかじめインストールしておいてほしい[注3]。

```
$ git clone git://github.com/basho/riak
$ cd riak
$ git checkout riak-2.0.0pre15
$ make devrel
$ dev/dev1/bin/riak start
$ dev/dev2/bin/riak start
$ dev/dev3/bin/riak start
$ dev/dev2/bin/riak-admin cluster \
  join dev1@127.0.0.1
$ dev/dev3/bin/riak-admin cluster \
  join dev1@127.0.0.1
$ dev/dev1/bin/riak-admin cluster plan
$ dev/dev1/bin/riak-admin cluster commit
$ dev/dev1/bin/riak-admin transfers
```

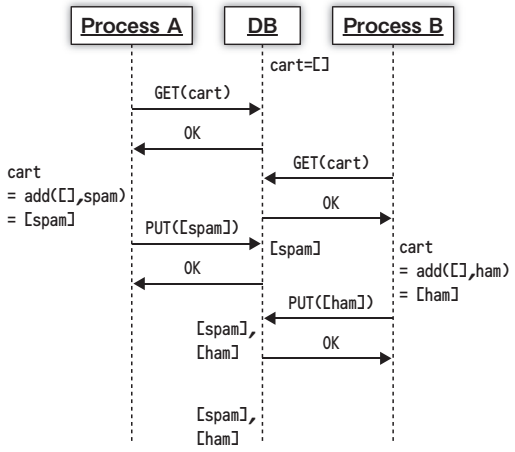
これで3台構成で動作するRiakクラスタが構築できる。**make**のところはとくに時間がかかるので、本稿を読みながら気長に待っていただければと思う。最後の**transfers**コマンドでは、クラスタ内のvnodeの移動の様子が観察

注3) hipeが無効になっていることが望ましい。以降の手順でうまくRiakが起動しない場合はhipeを無効にして試してみてほしい。

▼図2 上書きによる競合解決でショッピングカートを実現した場合
プロセスAとBがそれぞれ別の商品をカートに入れようとしても、先に書き込んだspamは後から書かれたhamによって上書きされてしまう。



▼図3 上書きによる競合解決でショッピングカートを実現した場合
プロセスAとBがそれぞれ別の商品をカートに入れた場合でも上書きが起きず、spamとhamがデータとして残る。



できるだろう。さらに、Siblingsを使えるようにするためには[注4]、

```
$ curl -i -X PUT \
http://localhost:10018/buckets/b/props \
-H 'content-type:application/json' \
-d '{"props":{"allow_mult":true}}'
```

とする。これで、bというバケットでSiblingsが使えるようになった。成功したかどうかを確認するためには、同じリソースをcurlを使ってGETすれば結果を確認できるだろう。

riak-admin transfersですべてのvnodeの移動が完了したことを確認したら、構築は完了である。

Siblingsを試す

実際にSiblingsが生成される様子を観察するためには、2回PUTをすればよい。

```
$ curl -X PUT -d '[0]' \
  http://localhost:10018/buckets/b/keys/k
$ curl -X PUT -d '[1]' \
  http://localhost:10018/buckets/b/keys/k
$ curl -i -X GET \
  http://localhost:10018/buckets/b/keys/k
HTTP/1.1 300 Multiple Choices
X-Riak-Vclock: a85hYGBgzGDKBVIcR4M2cgc...
Content-Type: text/plain
Content-Length: 56

Siblings:
6g2adFxNHcmdhqHdHTC7QI
3VzDuYLNxAINRiIFqeSeaA
```

これだけでそれぞれPUTされたデータが、別のSiblingとなる(最後のGETの結果は、誌面の都合上不要なHTTPヘッダを省略してある)。最後のGETの結果は、このkというキーには6g2adFxNHcmdhqHdHTC7QI と3VzDuYLNxAINRiIFqeSeaAという2個のvtagを持ったSiblingsがあり、このベクタークロックのバージョンはa85hYGBgzGDKBVIcR4M2cgc……であるという意味だ。試しに、それぞれのSiblingの値を取得してみよう。

図4のとおり、Siblingのvtagをパラメータとして渡してやるとそのSiblingの値が返ってくる。ほかのクライアントのライブラリであれば、一度の取得でリストで全部返ってくる場合もある。この例であれば、これをUnionして[0,1]というリストとして扱えば、Siblingsとして簡単に扱うことができる。

ハンドオフでSiblingsを試す

Siblingsがいかに強力かを示すために、ネットワークの一時故障による書き込み競合の例を試してみよう。まず、一時故障をシミュレートするためにdev/dev2/bin/riak stopとして、ノードを1つ落とす。この時点では、相変わらずデータを取得できることがわかるだろう。残っているノードに新しいデータをPUTしてみる。

```
$ curl -X PUT -d '[-1]' \
  http://localhost:10018/buckets/b/keys/k
```

これをGETして確認をすると、3つのSiblingsができていることがわかるだろう。続いて、ネットワーク分断を再現するために、今のノードを落として別のノードを起動する。手順は、まずdev/dev1/bin/riak stopを実行してからdev/dev2/bin/riak startとする。

この時点でGETを試すとSiblingsが2つできていることがわかるだろう。dev2のノードはHTTPポートに10028番を利用しているので、リクエストの際はそれを変更する。さらに、PUTをするとSiblingが増える。

注4) ここまで大事なRiakの機能なのに、なぜデフォルト設定ではないのかという指摘は最もなことだ。実際に開発チームもそうしたいと考えている。しかしながら、これは1.4以前との後方互換性のために諦めざるを得ず、開発チームとしても苦渋の決断であった。

▼図4 Siblingsのそれぞれの値の取得

```
$ curl 'http://localhost:10018/buckets/b/keys/k?vtag=6g2adFxNHcmdhqHdHTC7QI'
[0]
$ curl 'http://localhost:10018/buckets/b/keys/k?vtag=3VzDuYLNxAINRiIFqeSeaA'
[1]
```

```
$ curl \
  http://localhost:10028/buckets/b/keys/k
Siblings:
6g2adFxNHcmdhqHdHTC7QI
3VzDuYLNxAINRiIFqeSeaA
$ curl -X PUT -d '[100]' \
  http://localhost:10028/buckets/b/keys/k
$ curl http://localhost:10028/buckets/b/keys/k
Siblings:
6g2adFxNHcmdhqHdHTC7QI
3VzDuYLNxAINRiIFqeSeaA
4sbxBC6WJFT9e8Qhmf2jfR
```

これで、ネットワークが分断された状態で、別の値**[-1]**と**[100]**が書き込まれたことになる。再びすべてのノードを正常に戻すとどうなるだろうか？

dev/dev1/bin/riak startで止めていたノードを起動し、GETしてみる。

```
$ curl \
http://localhost:10028/buckets/b/keys/k
Siblings:
6g2adFxNHcmdhqHdHTC7QI
3VzDuYLNxAINRiIFqeSeaA
2rpMrBBYwcC3VpM0aLZ3pS
4sbxBC6WJFT9e8Qhmf2jfR
```

ハンドオフが完了するのを待ってからGETすると、Siblingが4個に増えている！――これによって、**ネットワーク分断があっても書き込み競合を起こさずデータをなくさないことが保証される**。読み込み時に値をすべてUnionすれば**[-1,0,1,100]**という正常なデータに戻すことができる。ここまでの流れは**図5**の上半分に対応しているので、そちらを見るとイメージしやすいかもしれない。

しかし、このままでは、書き込みのたびにSiblingsが増えてしまうという問題がある。1回や2回の書き込みで済むアプリケーションならばよいが、データを更新するたびに量が増えてはたまらない。また、上記のシナリオでは削除の扱いがよくわからないだろう。これらの問題を解決するために、ベクタークロックという技術を使う。

ベクタークロック

同じキーに対して複数のバージョン間で正しい値を決定する場合、最も簡単な解決法はそれぞれのバージョンに時刻を付けて、最新のものを採用するという方法だ。もしくは、PostgreSQLをはじめとするRDBMSが採用しているような、システム間でユニークなシーケンス番号を生成する方法だ。シーケンス番号が大きいほうが新しいことが保証される。

しかしながら、このいずれの方法も、Riakのような分散システムでは採用できない。まず、ユニークなシーケンス番号を生成するためには、そのまま設計したのではシーケンス番号を生成する部分がSPOF(単一障害点)になる。うまく工夫して冗長化するにしても、先月号で解説したようなアトミックブロードキャストのしくみがないと正しい番号を生成し続けることはできない。

また、一般に分散システムでは、コンピュータの時計は信頼できないことを前提にする。実際に、現在普及しているコンピュータが持っている時計はさまざまな理由でズレが発生するし、プログラムの動作進行やネットワークの遅延が保証されているわけではないため、時計から読み取った時刻を正確に運用することは難しい。

このような問題に対して、実時間を信用せず、**データの因果関係**を追跡できるようにしたのがベクタークロックだ。

基本的なしくみ

ベクタークロックの考え方はとても簡単だ。1つのキーに付けられるバージョン番号は1つのリニアな整数ではなく、複数の番号が付けられたベクトルだと考えるのである。たとえばRiakでは、そのキーを扱うvnodeの数だけ番号を用意する。それによって、2つのベクタークロックに因果関係があるかないかを特定できる。もし因果関係があれば、因果関係で新しいほうのデータを採用する。因果関係が定義でき

ない場合は、両方のデータをそのままSiblingsとして利用する。

ノードIDをzとすると、あるデータxのベクタークロックを次のように定義するとする。

$$VC(x) = (v_1, v_2, \cdots, v_z, \cdots)$$

このとき、あるキーに対するデータxとyの因果関係$x \rightarrow y$は、

$$VC(x) < VC(y) \Leftrightarrow$$
$$\forall z(VC(x)_z \leq VC(y)_z) \wedge \exists z(VC(x)_z < VC(y)_z)$$

と定義される。式で書くとなんのことだかわからないと思うが、「ベクトルの中の数字すべてが大きいか同じ」と覚えておくとよい。実際にRiakでは、zにはvnode IDを利用し、Vzはただの整数である。次に示すようなデータをベクタークロックとして保持している。ここでは簡単のため、vnode IDを**0,4,8,12,16,……32**とすると、たとえばvnode IDをプロパティ名にして、**a = {0:20, 4:21, 8:20}**といったふうに記述できる。これに対して、**b = {0:22, 4:21, 8:20}**というバージョンがあった場合、bのすべての番号が大きいか等しいので、$a \rightarrow b$といえる。

一方で、**因果関係がない**状態もあり得る。このときは、前述のaに対して、**c = {0:19, 4:21, 8:23}**というクロックがあったとすると、これは上記のような等号での大小関係を定義できない。このとき、$a||c$と表記することがある。

Riakでの使い方

それでは、先ほどの例に戻ってみよう。Siblingsが4個あり、それぞれの値が**[-1]**, **[0]**, **[1]**, **[100]**であることまでGETでわかったとする。今度は、ここから**[0]**を取り除いた**[-1, 1, 100]**という状態にしたいものとする。実は方法は簡単で、新しい値が4つのSiblingに対して因果関係があることをベクタークロックでRiakに教えてやれば良い。

```
$ curl -i -X GET \
  http://localhost:10018/buckets/b/keys/k
HTTP/1.1 300 Multiple Choices
X-Riak-Vclock: a85hYGBgzGDKBVIcR4M2cgc...
Content-Type: text/plain
Content-Length: 56

Siblings:
6g2adFxNHcmdhqHdHTC7QI
3VzDuYLNxAINRiIFqeSeaA
2rpMrBBYwcC3VpM0aLZ3pS
4sbxBC6WJFT9e8Qhmf2jfR
$ curl -i -X PUT \
  -H 'X-Riak-Vclock: a85hYGBgzGDKBVIcR...'\
  -d '[-1, 1, 100]' \
  http://localhost:10018/buckets/b/keys/k
$ curl -i http://localhost:10028/buckets/b/keys/k
HTTP/1.1 200 OK
X-Riak-Vclock: a85hYGBgymDKBVIcMRuucwUzJLtkM...
...

[-1, 1, 100]
```

すべてのSiblingがなくなり、1つの値に無事に更新され、ベクタークロックの値が新しいものになったことがわかるだろうか。

また、この瞬間に**[1]**を削除するという別のリクエストが同じベクタークロックのバージョンに対して実行されたとすると、それもまたSiblingになる。

```
$ curl -i -X PUT \
-H 'X-Riak-Vclock: a85hYGBgymDKBVI...' \
-d '[-1, 0, 100]' \
http://localhost:10018/buckets/b/keys/k
$ curl -i http://localhost:10028/buckets/b/keys/k
HTTP/1.1 300 Multiple Choices
X-Riak-Vclock: a85hYGBgymDKBVIcMRuucwUzJLtk... \
...

Siblings:
2ZW6NOqz1YdeChWy7Sv5xcp
38Ip5UO9iTYfi6HRBQQp3s
```

この一連の流れを図にすると、**図5**のようになる。ここまで、Siblingsとベクタークロックについて解説してきた。これで、ネットワーク分断などの一時的な障害に対して、ほかのシステムではできない状況でも可用性を担保しつつ、書き込み競合などでデータをなくさないようになっているしくみがおおよそ理解できたかと思う。

削除の扱い

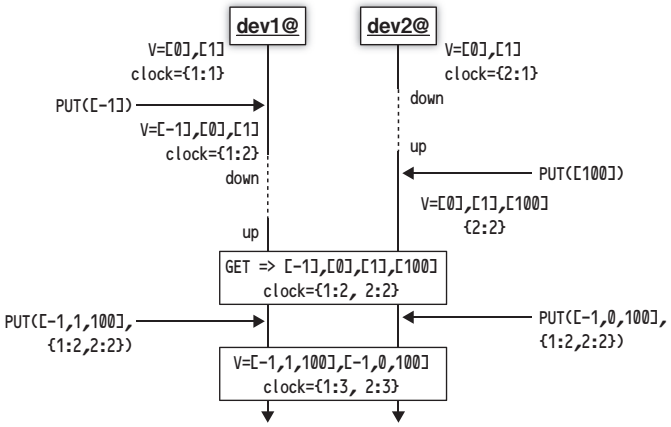
ここまで、データの上書きや更新がどのように動作するかは解説してきたが、削除がどう動作するかは解説してこなかった。単純に削除してベクタークロックも消すような実装であれば、冒頭に少し触れたとおり消したはずのデータがよみがえってしまう。そこでRiakでは、削除の動作は“Tombstone”(墓石)というデータをPUTするという、論理削除に近いしくみになっている。このTombstone自体も更新とみなされるので、ベクタークロックを指定して削除を行うという動作になっており、もしも因果関係がない更新が並行して処理された場合には、TombstoneもSiblingとして登場する。

実際に、3個のSiblingを持っているケースで、削除と更新を並行に動作させて、競合を起こしてみよう。まず、先ほどと同様の環境で、次のように、kというキーに[-1], [1], [100]という値がそれぞれあるものとする。

```
$ curl -i http://localhost:10028/buckets/b/keys/k
HTTP/1.1 300 Multiple Choices
X-Riak-Vclock: ➡
a85hYGBgzGDKBVIcMRuucwXd3s2awZT...
...

Siblings:
5mmjvFqUT3ji4RahBQlmCh
6xaetbhxjVTEaHhM2LCeew
4zSJOKSV5wum1V5Wa2MdvL
```

これをいったん削除する。このとき、DELETEの裏側では、Riak内でベクタークロックの値を取得・指定してTombstoneをPUTしている。直後のGETの結果は404になるが、

▼図5 ここまでのデータ操作の流れと、おのおのの時点でのベクタークロックの値
ベクタークロックのバージョンを指定してPUTすることで、因果関係が明確になっている様子がわかる



コラム コンピュータと相対性理論の意外な関係

コンピュータの中でもそうだが、我々が生活している日常の時刻はたった1つの数[注a]で表現できる。つまり、すべての時刻に順序が付けられるものとしている。ところが、アインシュタインの相対性理論によれば、2つの事象があったときに、お互いの光錐に入っていない場合はお互いを観測することができない。

このような相対論的な事象の因果関係と、分散システムにおけるコンピュータ間の通信は似ているのではないか？ と着想したレスリー・ランポートはベクタークロックのもとになるロジカルクロックを着想したと言われている。物理の教科書に出てくる光錐と、ベクタークロックの因果関係の範囲図が非常に似ているのは、不思議……でもなんでもないのである。

注a) 時間が実際に有理数、実数、整数のどれであるかという議論はここではせず、1次元の整数または浮動小数点つき数で表現できることとする。

X-Riak-Deletedというメタデータに着目してほしい。

```
$ curl -X DELETE \
  http://localhost:10028/buckets/b/keys/k

$ curl -i \
http://localhost:10028/buckets/b/keys/k
HTTP/1.1 404 Object Not Found
X-Riak-Vclock: a85hYGBgzGDKBVIcMRuucwXd...
X-Riak-Deleted: true
...

not found
```

値がtrueとなっており、これがTombstoneを表している[注5]。つまり、削除に成功したということだ。続いて、並行なPUTを再現するために、削除前のベクタークロックを指定してPUTを行う。

```
$ curl -X PUT \
  http://localhost:10028/buckets/b/keys/k \
  -H 'X-Riak-Vclock: a85hYGBgzGDKBVIcM...'\
  -d '[-1, 1, 100]'
```

これをGETしてみると、Siblingが2個できていることがわかる。片方はTombstone、片方はPUTされたデータ[-1, 1, 100]だ。ここからわかることは、DELETEとPUTが競合を起こしたということである。アプリケーション側で、このDELETEとPUTのどちらを採用するか、どのようにマージするかを決めることができる。最も単純な解決は、タイムスタンプを見て新しいほうを採用することだろう。つまり、従来のKVSが提供している強制的な上書きのセマンティクスになる。こういう挙動を嫌う場合は、Tombstoneを無視して、残っているデータを有効なものとして扱うこともできる(図6)。

このように、削除と更新が競合し、消したはずのデータがよみがえる可能性がある場合は、その可能性をSiblingsという手段できちんとアプリケーションに伝えることができるようになっている。なお、デフォルトでは、TombstoneがPUTされて3秒後に物理的な削除がトリガーされるようになっている。そのときに競合がなければ、Riakのレイヤではデータが物理的に削除されることになる[注6]。

CRDT

ここまで、Riakがデータをなくさないためのしくみを説明してきた。複数のバージョンを同時に保持することを許し、値を読むときにそ

注5) HTTPでなくProtocol BuffersのAPIを利用したクライアントでは、このあたりのセマンティクスが場合によってはわかりにくくなっている。Riakオブジェクトが GETできたにもかかわらずデータが空の場合があるが、そのときはメタデータのTombstoneがあるかないかをチェックすると、そのデータが有効かどうかわかる。詳しくはクライアントのリファレンスを参考にしてほしい。

注6) バックエンドのLevelDBやBitcaskで実際に削除されるのはまた別のタイミングになる。

▼図6　削除と更新が競合した結果のSiblings

```
$ curl -i http://localhost:10028/buckets/b/keys/k
HTTP/1.1 300 Multiple Choices
X-Riak-Vclock: a85hYGBgzGDKBVIcMRuucwXd3s2awZTImsfKcPRBxRm+LAA=
...

Siblings:
7JvmhfzirrF76mUh1qEIAf
1HibiDU6qDBaCKKdsxnnOn
$ curl 'http://localhost:10028/buckets/b/keys/k?vtag=7JvmhfzirrF76mUh1qEIAf'
[-1, 1, 100]
$ curl -i 'http://localhost:10028/buckets/b/keys/k?vtag=1HibiDU6qDBaCKKdsxnnOn'
HTTP/1.1 200 OK
X-Riak-Vclock: a85hYGBgzGDKBVIcMRuucwXd3s2awZTImsfKcPRBxRm+LAA=
X-Riak-Deleted: true
...

not found
```

れらを解決して正しいバージョンを得るという方法である。しかしこの方法では、アプリケーションごとに、Riakに保存されるキー(オブジェクト)ごとに、Siblingができることを許すかどうか、Siblingができたとして読み込み時にどのように解決するかを毎度検討するか、さもなければお決まりのパターンをライブラリにしなければならない。

実際に、Riakのライブラリには、そのようにして読み込み時解決の方法を提供するライブラリが多く提供されている。しかし、これは各言語で実装しメンテナンスするコストが比較的高い。そこで、Riak 2.0からはCRDT[注7]と呼ばれる、読み込み時の解決方法があらかじめ決まっているデータ型を導入することでライブラリ側の実装の負担を軽減する方針に変更した。RiakのCRDTでは、次のの4種類の可換データ型が用意されている。

・PN-Counter
Positive-Negative Counters のこと。加算と減算が可能な整数型。ページのクリック数や、ログインユーザ数、「いいね!」数などのリアルタイムなカウントなどがユースケース

・ORSWOT-Sets
要素の削除にも対応したリスト型。ショッピングカートなど、要素のリストを保持することができる

・LWW-Register
Last-Write-Wins Registerのこと。Riakでは、文字列とタイムスタンプを保持し、タイムスタンプが新しいものを選ぶしくみになっている

・maps
上記の型の組み合わせ。JSONのようにプロパティ名とデータ型を指定することで1つのキーに対して複数のCRDT値を保持できるようにする

ほかにもグラフ型というものがあるが、Riakでは対応していない。これらの型を利用すればSiblingが発生しない(Riak側で自動的にマージされる)ため、アプリケーションは通常のKVSだと思って利用するだけでよい。

それだけで、ほかのKVS(Key Value Store)とは異なり、ネットワーク故障や一時故障によっても実際にデータを失わない(あるいは、削除したデータがよみがえらない)、可用性の高いデータベースとして簡単に利用できるようになる。

まとめ

先月号では、分散データベースにまつわる整合性と永続性の問題について解説し、Riakではヒント付きハンドオフによって可用性と併せてそれを実現していることを解説した。とくに整合性については、Siblingsの「とりあえず書き込んでおいて、読み込み時に解決する」しくみと、ベクタークロックの「因果関係を決定するしくみを用意し、Siblings数をきちんと制御する」しくみについて解説した。

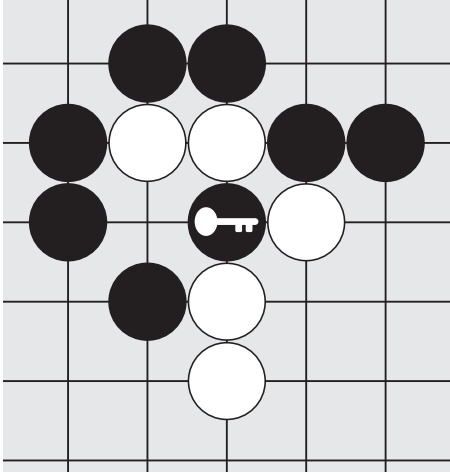
さらに、それらのしくみのために、通常のRDBMSやKVSとのデータモデルと異なり、アプリケーション側のロジックで衝突を解決しなければならないという問題については、Riak 2.0で実装されるCRDTを使えば負担がかなり軽減されることを解説した。

しかしながら、「だから、データがなくならない」というまでにはまだ解説することがある。それは、これまでネットワークやマシンの一時的な故障だけを想定した問題だけを解説してきたことである。実際にディスクやサーバが故障して、そこに保存されているデータが失われたときにコピーの数はどうしても縮退してしまう。次号では、どのようにしてRiakが失われたデータを修復していくのか、そのしくみについて解説していく。SD

注7) Conict-free Replicated Data Types: 定訳はまだないが、あえて日本語にするなら衝突しない複製データ型といえるだろう。

セキュリティ実践の基本定石

すずきひろのぶ
suzuki.hironobu@gmail.com



みんなでもう一度見つめなおそう

【第十回】根深くはびこるDDoS攻撃の脅威

昨年末からNTPサーバを使ったインターネット史上最大級と言われているDDoS攻撃が発生しています。この機会にDDoS攻撃のしくみ、原因、最新動向について考えてみたいと思います。

DoS攻撃

DoS攻撃のDoSとはDenial of Serviceの頭文字からきています。これを直訳すると「サービスを停止させる」あるいは「サービスを不能にさせる」という意味になります。たとえばサーバの機能を止めてなんらかの不都合が生じるような攻撃であれば、とりあえずDoS攻撃と呼ぶことができます。

プログラマであれば、プログラムに誤りがあり(たとえ仕様上正しくても)、データを入力すると異常停止してしまう場面に遭遇したことがあると思います。そんなに特別なこととは思わないでしょう。

しかし、この異常停止を意図的に発生させた場合、サービスを不能にしますからDoS攻撃となります。たとえば、1997年にあったPing of DeathというDoS攻撃手法がこれにあたります。Ping of Deathは、もともとはIPパケットのフラグメンテーション処理のバグが原因です。大きなサイズのICMP Echoを相手マシンに送ることで、そのバグを発現させてOSをハングアップさせます。その結果、システムが提供するサービスは停止します。1回の(異常を引き起こす)データ送付を受けただけとはいえ、システムがダウンしてしまうので引き起こす結果は大きいです。しかし、元をたどれば、そのバグは単純な実装ミスから発生しています。

またプログラムには問題がなくともシステムの容量を越えるデータを与える、あるいは要求することによってシステムを麻痺させるのもDoS攻撃です。たとえば、ツールを使ってWebサーバに対して故意に大量の接続やリロードを行う手法などが挙げられます。TCPの3-wayハンドシェイクの弱点をつくSYN Flood(コラム参照)もこれにあたります。あるいは、ネットワークに大量にデータを送りつけ、ネットワーク帯域を消費し、ほかの通信を行えなくするような手法も同じくDoS攻撃と言えます。

DDoS攻撃

このDoS攻撃を複数の攻撃元に分散させて行うのがDDoS攻撃(Distributed DoS Attack)です。

資源を消耗させるDoS攻撃を1対1で行う場合、攻撃元が一般に引かれたネットワーク回線で、攻撃先がデータセンターのような場合、当然、効果は薄いと言えます。しかし、たくさんの攻撃元から分散して行えば、その効果は絶大です。たとえ1人の仕業でも分散していればDDoS攻撃は成立します。

マルウェア感染型DDoS攻撃

ネットワークに接続されている多数のコンピュータをマルウェアに感染させ、そのマルウェアにあらかじめ指定していた攻撃先へDoS攻撃を行わせます。もしくは、マルウェアをネットワークを介して攻撃命令を行うセンターに接続させ、そこから攻撃先の情報を得てDoS攻撃を行わせるというパターンもあります(**図1**)。このような攻撃命令を出すセ

ンターのことをCommand and Control Server、略してC&Cサーバと呼びます。

ACCSへのDDoS攻撃

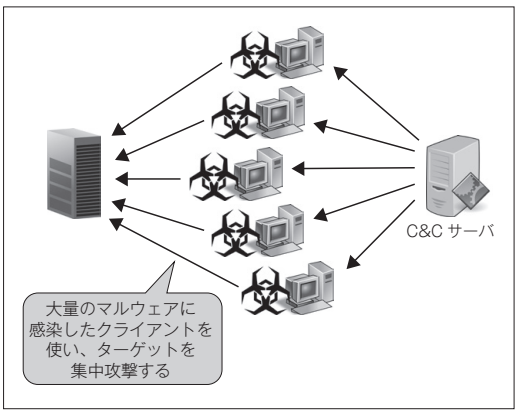
今から10年前の2004年に、日本国内で大規模なDDoS攻撃として有名になったのがantinny系のマルウェアを使ったACCS（コンピュータソフトウェア著作権協会）のサーバへの攻撃です。

antinnyはファイル交換ソフトWinnyを介して感染拡大するワーム型マルウェアです。ACCSにDDoS攻撃を行ったマルウェアはantinnyの亜流に複数のDoS攻撃手法を組み込み、最初からACCSのサーバを狙うだけの機能しか持たない極めて悪質なマルウェアでした。SYNパケットを送るのみのSYN Flood攻撃、実際にTCP接続を行うConnection Flood攻撃、Webサーバのページを大量に取り込むHTTP GET Flood攻撃、反対にWebサーバへ大量のデータを送り込むHTTP POST Flood攻撃など、複数の攻撃手法を組み込んでいました。

それまで国内で、これほど周到かつ大規模なDDoS攻撃を経験したことはありませんでした。そこでISP団体や通信事業者などが協力し、この事件を今後も同様なDDoS攻撃が起こった場合の知見とするためにコストを度外視して対応にあたりました。通信量を計測するために実験的に通信に使われている帯域を大きくしていった結果、最大700Mbpsに達していることがわかりました[注1]。

現在でも700Mbpsレベルのトラフィックに対応できるサーバを借りるのは、たいへんなコストがかかります。もちろん当時としては驚くほどの流量で

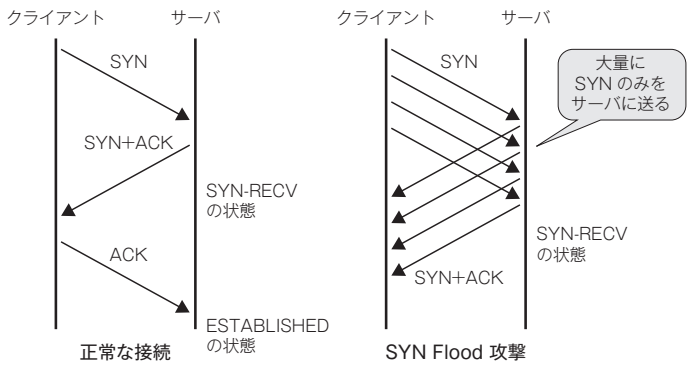
◆図1　マルウェアに感染したクライアントを中央からコントロール



◉SYN Flood攻撃

インターネットの通信プロトコルTCPは、接続の最初に3-wayハンドシェイクと呼ばれる接続手順を行います。このときにクライアントから最初に送られるのが、SYNフラグを立てたTCPパケットです。それに対応してサーバからSYNとACKのフラグを立てたTCPパケットがクライアントに送られます。最後にクライアントからACKフラグを立てたTCPパケットをサーバに送って、それがサーバに届いたら、TCPによる接続が確立したことになります。

◆図2　正常な接続とSYN Flood攻撃の違い



このとき、SYNフラグを立てたパケットだけを大量にサーバに送りつけると、サーバは多くのクライアントの接続準備のみ行われる状態になります。接続準備の上限の量を越えたとき、それ以上新しい接続はできなくなり、結果としてサービスが不能になります（図2）。

注1）当時、家庭に引かれていたのはADSLで、現在のような光ネットワークはまだ広く普及する前でした。

あり、その事実にあらためてDDoS攻撃の危険性を認識したのでした。

増幅型DDoS攻撃

Character Generator ProtocolまたはCHARGENと呼ばれるプロトコルは、1983年に作られたRFC 864で定義されているプロトコルで、TCPかUDPで通信すると文字列を返すという単純な機能を提供しています。もともとはネットワーク上にあるコンピュータとの通信をテストするためにあり、RFC 864にも“A useful debugging and measurement tool is a character generator service.”と書かれています。

このプロトコルを使ってTCPで接続(connection)した場合、延々と続く文字列データが送られ続けます。この機能はUDPでも提供されており、その場合は非接続(connectionless)です。中身が0～512バイトのランダムな長さの文字列の入ったデータグラムが返ってきます。

しかし、UDPは接続しないで(接続という概念がない)、データグラムと呼ばれるデータの塊を投げ合うだけですから、送信元IPアドレスを詐称することが可能です。攻撃ターゲットのIPアドレスを詐称して入れておけば、送信先からの戻りパケットは攻撃先に送られます。この場合、28オクテット[注2]の詐称パケットを送ると最大540オクテットのデータ量に増幅され、攻撃ターゲットに送られることになります。攻撃をしかけた側のデータ送出の最大約19倍に増幅されて攻撃ターゲットに届きます。

さらに前述したように、マルウェアを使った複数マシンから偽造パケットを送出させ、最終段階の増幅と組み合わせれば極めて大量のデータを発生させる攻撃が可能です(**図3**)。

現在、CHARGENはPC、サーバなども含めてデフォルトではインストール自体されていません。しかし、古いルータはデフォルトで設定されており、インターネット上ではまだ現役で動作しているようです。観察していると、それらを探すためと思われるパケットが今でも日常的にCHARGENポートに届きます。しかし、CHARGENはすでに使われない機能ですので、古いルータが引退するにつれ徐々に少なくなっていくでしょう。

現在のDDoS攻撃の脅威

2014年1月時点での最大のDDoS攻撃の脅威は、DNSとNTPの2つです。DNSはホスト名からIPアドレスを引くためのサーバで、インターネットの中では重要な役割を果たします。NTPは時刻同期のためのプロトコルで、サーバだけではなく時間管理が必要なネットワーク機材にも組み込まれています。この2つには次の特徴があります。

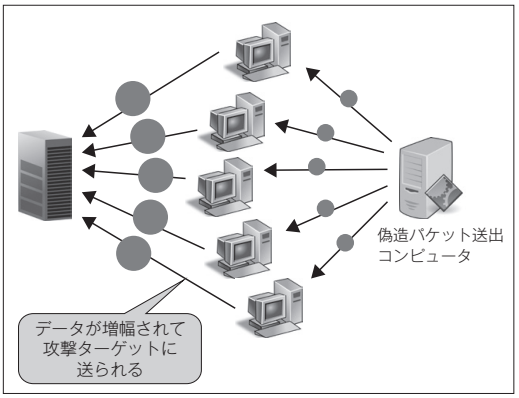
- **サーバとして提供される場合が多く、ネットワーク的に大量のデータを送出できる場所で動作させている**
- **インターネット上のサービスとして重要かつポピュラーである**

DNSオープンリゾルバによるDDoS攻撃

リゾルバ(resolver)とは、ホスト名からIPアドレスを引いたり(解決したり)、IPアドレスからホスト名を引いたりする機能のことです。DNSに再帰的な問い合わせ(recursive queries)を行うと返答してくるデータが大きくなります。

オープンリゾルバとは制限をせずに誰にでも

◆図3　送信元IPアドレスを詐称したパケットを送る



注2)　IPヘッダ(20オクテット)＋UDPヘッダ(8オクテット)。1オクテット(octet)は8ビット。

DNSを使用させることを意味しますが、ここではとくに誰にでも再帰的な問い合わせを可能にさせているDNSサーバを指します。国内外に多数存在しDDoS攻撃の踏み台として悪用されています。

この問題は古くから認識されていて、2006年には、JPRSから「DNSの再帰的な問合せを使ったDDoS攻撃の対策について」という注意勧告が出ています[注3]。ところが、まだこの問題は継続しており、現在でも積極的にDDoS攻撃に使われています。2013年4月に、JPCERT/CCは「DNSの再帰的な問い合わせを使ったDDoS攻撃に関する注意喚起」を出しています[注4]。

運用しているDNSサーバの設定が正しくなく、再帰的な問い合わせに対してレスポンスを返すサーバであることに、システム管理者が気づかないこともあるでしょう。また、再帰的問い合わせには返答しなくとも、古いブロードバンドルータがオープンリゾルバになっていて、ユーザがそれに気がつかないという場合もあります。

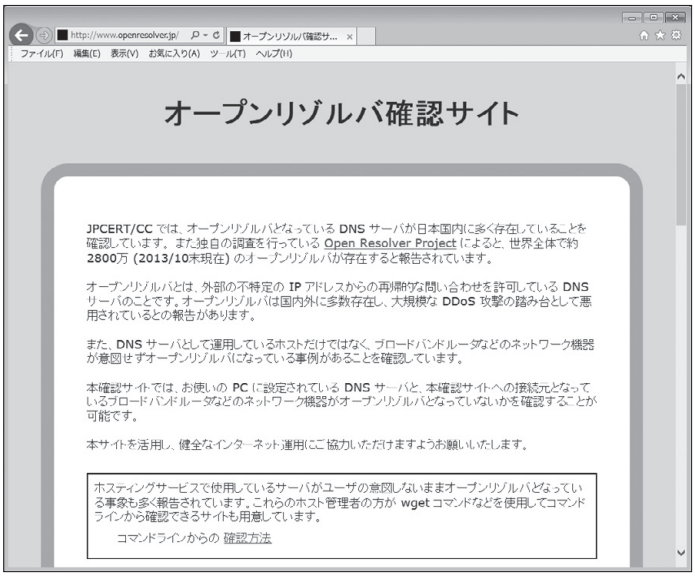
自分でオープンリゾルバか否かをチェックするには、確認用ツールを使う必要があります。しかし、一般のユーザはいちいちそのような手間をかけてはいない、もしくは面倒でやらない、そもそもそのような必要性を認識していない、といった問題が存在しているがゆえに、いまだにこの問題が尾を引きずっているのだと思います。そこでJPCERT/CCでは、チェック用の専用Webサイト[注5]を開設し、専門的知識がなくてもクリックだけでオープンリゾルバの確認がとれるようにしました(**図4**、**図5**)。

図4の説明の中でも紹介されているOpen Resolver Projectは、全世界のオープンリゾルバを検索しデータベース化しています。

JPCERT/CCのサイトには、「日本国内でも3、4年前まで1万を越えるオープンリゾルバが計測されており、現在もまだ数千のオーダーで残っている」というデータが出ています。

数が多いことと、それが順調に減ってきている最大のファクタは、オープンリゾルバのままで出荷されていたブロードバンドルータが原因だったからと筆者は考えています。筆者が以前使っていたブロードバンドルータを引っ張りだして確認したところ、やはりデフォルトでオープンリゾルバになっていました。筆者も利用している最中には気づかず、何年も使っていました。ちなみに、筆者は内部ネット

◆**図4　オープンリゾルバ確認サイト**



◆**図5　オープンリゾルバ確認結果**
(大丈夫な場合、以下の画面が表示される)

注3) http://jprs.jp/tech/notice/2006-03-29-dns-cache-server.html
注4) https://www.jpcert.or.jp/at/2013/at130022.html
注5) http://www.openresolver.jp/

ワーク上にDNSキャッシュのサーバを用意しているので、ブロードバンドルータにデフォルトでDNS機能があったことには気づきませんでした。

史上最大級のDDoS攻撃が発生

ピーク時で400Gbpsというインターネット史上最大級のDDoS攻撃が発生しています。これはNTPサーバを使い増幅させたDDoS攻撃です。これは過去の話ではなく、2014年2月時点ではまだ問題のまっただ中にいます。

NTPサーバにはmonlistという過去にNTPサーバとやりとりしたクライアントの情報をリクエストする機能があります。現在、この機能はデフォルトではオフになっていますが、古い実装だったり、あるいはオンにしていたりするとリクエストに答えます。最大600件の過去のリストが返答されます。

NTPはUDPを使うプロトコルですので、送付元IPアドレスを詐称可能です。また返答するデータは極めて増幅率が高くなっています。さらに、NTPサーバですのでネットワーク資源が十分にある環境に設置されており、攻撃ターゲット側にしてみれば最悪な条件がそろってしまっています。

この件に関しては、すでにJPCERT/CCが2014年1月15日に「ntpdのmonlist機能を使ったDDoS攻撃に関する注意喚起（JPCERT-AT-2014-0001）」として告知しています[注6]。また、BBCが2014年2月11日に“Huge hack 'ugly sign of future' for internetthreats”というタイトルで報道し[注7]、その中でおおよそ400Gbpsという巨大な帯域を使ったインターネット史上最大級のDDoS攻撃であると紹介し、これが一般の人への喚起となり、多くの人がDDoS攻撃の脅威を知るところとなりました。

2004年のACCSへのDDoS攻撃は700Mbpsで、その10年後には400GbpsのDDoS攻撃が発生しています。倍率にして約570倍です。400GbpsものトラフィックはISPそのものも麻痺してしまうレベルです。すさまじいとしか言いようがありません。

NTPサーバ管理者のメーリングリストから事件を読み解く

筆者はたまたま公開NTPサーバの管理者がやりとりするメーリングリストpool@lists.ntp.orgに2010年から登録していて、この問題が発生した初期の段階から経過を知ることができていました。

pool.ntp.orgとはNTPサーバをプールしていてNTPサービスをインターネット上に提供しているボランティアグループです。次のように入力すると、NTPサーバのIPアドレスがいくつか戻ってきます。

```
$ nslookup pool.ntp.org
```

pool.ntp.orgはこのNTPサーバを提供し、メンテナンスをしていくことでインターネット上の時間同期を確実なものにするという努力をしています。

さて、前述のDDos攻撃の兆候は、2013年11月5日にほかの人からのメールの引用という形で紹介されました。“I received an abuse email today（NTPサーバが不正利用されたという連絡がきた）”ということで手短に状況が書かれていました。

- **管理する2台のNTPサーバにIPアドレスが詐称されたクエリが送られてきた**
- **偽装されているポートは80番であった**
- **iptablesによる制限とntp.confによってアクセス元を制限した**

そして、「何かアドバイスがあればありがたい」というメッセージが最後に付けられていました。するとほかの管理者から次の情報がもたらされました。

- **同様の問題があり、10月初旬から2、3週間NTPサーバの提供を停止していた**
- **アウトバンド（外部に送出される）トラフィック量が10GB/時だった**
- **TX/RXのレシオ（ネットワークで入ってくる／出ていく割合）は111:1だった**
- **IPアドレスの多くは/8でランダムなものだった**

それから2ヵ月近く過ぎた2013年12月30日に、

注6） http://www.jpcert.or.jp/at/2014/at140001.html
注7） http://www.bbc.co.uk/news/technology-26136774

メーリングリストにNTPリフレクション攻撃がSymantecによって報告されているという話題が流れました[注8]。

- monlist requestによる増幅攻撃が増えている
- (Symantecでは)設定ファイルでdisable monitorを設定することで防げる

このメーリングリストのメンバーはNTPサーバ管理に対してはたいへん経験豊かな人たちですので、すぐに次のようなアドバイスが流れました。

- (対処済みの)ntpd 4.2.7p26は2010年4月24日にリリースされているものである
- noqueryという設定もある
- Ubuntu 12.04のntpd 4.2.6p3でもdisable monitorは有効

4.2.7p26以降でnoqueryが設定できるという話題は、2011年12月にすでに議論しているので、目新しい話題ではありません。また、monlistを使って増幅を狙う攻撃も2012年6月に報告されていたので、既知の攻撃でした。このときは、21,000リクエスト/分のmonlistのリクエストを2日間に渡って受けたというものです。実際には、事前に対応したので増幅攻撃とはなりませんでした。このサーバは100Mbpsでインターネットに接続していましたが、もし攻撃が成功していたなら、この帯域はすべて使い果たしただろうと予想されていました。

この攻撃の話題は、2012年6月以来聞かれなかったのですが2013年の暮れになって急にDDoS攻撃に使われることとなり、今やインターネットの脅威の一番に挙げられるまでになってしまいました。

monlistに関しては少なくともpool.ntp.orgのメンバーはすでに対応済みのレベルですが、次のような機器は、まだ問題として残っています。

- 企業や大学などの小規模な公開NTPサーバ
- ntpdが動いていてmonlistリクエストを受け付けるネットワーク機器(ルータなど)

これらは外部に動いていることを教えなくても、今はスキャンをすればすぐに見つけられます。ネットワークに接続されたサーバなどのセキュリティチェックをするスキャンツールとしてnmapがありますが、現在ではスキャン速度がnmapの1300倍速いzmapが登場しています。1Gbpsのネットワークでインターネットに接続されていれば、インターネットのIPv4空間すべてをスキャンするのに理論上45分しかかかりません。このようなツールがあれば設定ミスやあるいは、ユーザは知らないけれどもNTPサーバの能力を持っているネットワーク機器などを見つけるのは、難しいことではありません。

IPアドレス詐称とUDP

増幅攻撃が可能であるかどうかに限らず、UDPを使う限りIPアドレス詐称によるリフレクションが発生します。ですからサービスを提供する側はどこかへの攻撃に使われるポテンシャルは常にあります。これを防ぐにはISPがきちんとBCP38(RFC 2827)と呼ばれるイングレスフィルタ[注9]をする以外にはリスクを軽減する方法はありません。

RFC2827は今から15年も前に出されたもので、当時すでにリスクが認知されていたにもかかわらず、インターネット全体としてあまり対応が進まず、15年経って最大の脅威の1つとして現れてしまった、というのが現状だと言えます。

小さくても大きな意味がある

400GbpsのDDoS攻撃の問題に関してはエンドユーザとして直接タッチできないとしても、「管理しているNTPサーバの設定は大丈夫か」、「自分の管理しているネットワークの範囲でオープンリゾルバが動いていないか」を確認するなどやるべきことはあります。そんな小さなことでも見逃せば、悪意のある(悪い意味でよく考えられた)攻撃に利用され、結果として大きな問題につながります。小さいことでもコツコツと積み上げるのが、セキュリティへの大切な道筋だと言えるのかもしれません。SD

注8) NTP reflection attack https://isc.sans.edu/diary/NTP+reflection+attack/17300
注9) ルータから外に出ていくパケットのIPソースアドレスをチェックすること。

プログラム知識ゼロからはじめる
iPhoneブックアプリ開発

最終回

アプリの売り方、そして次につなげるために

GimmiQ（ギミック；いたのくまんぼう&リオ・リーバス）
URL http://ninebonz.net/
URL http://www.studioloupe.com/
イラスト●中川 悠京

プログラミングをしたことのない方にもアプリを作る楽しさを味わってもらいたい本連載。今回は作ったアプリを多くの人に知ってもらう方法、さらに次のアプリへとつなげるためにできることを、筆者たちがやってきた経験に基づいてお伝えします。

一人でも多くの人に届けるために

1年間続いたiPhoneブックアプリ開発も今回で最終回となります。これまでの連載で、アプリを作ることに関してはある程度の知識が得られたことと思います。これまでのチュートリアルを続けてきた方の中には、すでにApp Storeで自分のアプリをリリースしている方もいます。そして、これから本格的にアプリを開発してストアで配信される方が増えていくことを願っています。

そこで最終回である今回は、これまで作ってきたブックアプリに限らず、作ったアプリを一人でも多くの人に届けるために「できること」について、第1弾としてリーバスが、第2弾としてくまんぼうがそれぞれ書いてこの連載を締めたいと思います。

まず最初に、残念ながらアプリをヒットさせるための絶対的な方程式はありません。筆者（リーバス）もこの市場で5年間アプリを売ってきましたが、確実にヒットに導く方法というものには行き着くことができませんでした。しかし、だからといって完全に運任せの世界というわけでもありません。ヒットの的中率を高めるためにできることはたくさんあり、それら1つ1つをこなしていくことで、自分の作品が埋もれてしまうのを防ぐ努力は最低限必要だと考えます。

アプリを船にたとえた場合、あなたは船を造った設計士であると同時に航海士にならなければいけません。ただ船を無鉄砲に大海原に送り出したところで、嵐に巻き込まれて沈没し、日の目を見られなくなるリスクが高まります。 せっかく苦労して生み出した我が子のような船ですから、好調な船旅のスタートを切るための航海術も同時に身につけることが重要なのです。

プレスリリース・レビュー依頼を送る

アプリをリリースするときに基本となるのはアプリ紹介サイトにプレスリリース・レビュー依頼を送ることです。メジャーなサイトであればあるほど、なかなか簡単には取り上げてもらえませんが、やれることはやりたいという方は一通り人気の紹介サイトにPRを送るべきでしょう。ただし、あからさまにテンプレート文だとわかるような内容を一斉送信するような送り方は控えるべきです。基本的な情報はともかく、なぜそのサイトで紹介してほしいか、そのアプリに込めた想いなど、気持ちを込めて書くことが大事だと思います。相手も人間ですから、熱意を感じれば試してみようという気になるでしょうし、逆に適当に送られてきたプレスリリースは迷惑メールと同じような扱いを受けても仕方がありません。

プレスリリースの内容はなるべく短く要点をまとめつつも、他のアプリに勝っている部分は

しっかり強調し、画像、動画リンク、App Storeのリンクなどの基本情報を忘れないようにしましょう。そして、有料アプリであれば、100個までお試しとして配布できるプロモーションコードを付けましょう。アプリは一般公開する前でも、Appleの審査さえ通っていればプロモーションコードを配布できるようになるので、プロモーションコードを送った相手にだけ事前にアプリをダウンロードしてもらうことができます。審査が通ったら、アプリのリリース日を少し先の日程に決め、プレスリリースで事前にプロモーションコードを渡しておけば、リリース後すぐにアプリを紹介してもらえる可能性が高まります。アプリを取り上げてもらえるかどうかはプレスリリースでどれだけ興味をもってもらえるか、そしてアプリの内容によって左右されるため、出せば確実に紹介してもらえるというものではありませんが、出して損することはないのでぜひ試してみてください。

iPhoneアプリ紹介メディアのリスト

今やiPhoneアプリを紹介するメディアやブログは数えきれないほどあるので、すべてを紹介することはできませんが、一般的なメディアとその連絡先をまとめたリストを作ってみました(**表1**)。ほかにもたくさんあるので、ネットで「iPhoneアプリ レビュー媒体一覧」などで検索してみることをおすすめします。

ただ、メディアに紹介してもらうことだけを頼りにしていてはいけません。いくらメジャーなメディアに紹介してもらったとしても、効果は長く持続するものではなく、一時的なものでしかないので、最終的には自分でアプリを売っていく力を身に付けることが重要になってきます。では、どうすればそのような力を付けていくことができるのでしょう？　この先はそのあたりを掘り下げていきたいと思います。

アプリ内課金VS.無料&有料版

近頃は「時代はフリーミアム」という話をよく聞くようになりました。基本は無料で提供し、さらに機能を追加したりおまけ的要素をアプリ内で販売するというビジネスモデルです。それまではお試しの無料版があり、別のアプリとして有料版が存在する形が主流でしたが、今の成功しているアプリの例を見ると、ほとんどがこの「基本無料＋課金」というスタイルになってきています。

ただ、果たしてこれはアプリ市場に入りたての新参者にとっても通用するビジネスモデルなのかというと、必ずしもそうとは思いません。というのも、このフリーミアムが流行りだしたこともあり、無料アプリのランキングで上位に

▼表1　iPhoneアプリを紹介するメディアやブログ一覧

名称	URL	問い合わせ
全般		
AppBank	http://www.appbank.net	http://www.appbank.net/2011/08/01/iphone-news/282433.php
meet-i	http://web.meet-i.com	info@meet-i.com
アップス！	http://www.appps.jp	info@appps.jp
Touch Lab	http://touchlab.jp	http://touchlab.jp/about/
iPhonePLUS	http://iphone.ascii.jp	http://iphone.ascii.jp/about/
女性向け		
iPhone女史	http://www.iphone-girl.jp	app@iphone-girl.jp
iPhone女子部	ttp://www.iphonejoshibu.com	release@iphonejoshibu.com
アプリソムリエ	http://appsomm.jp	http://appsomm.jp/contact/
Girl's App	http://girlsapp.jp	http://girlsapp.jp/contact/
スマホガール	http://spgirl.jp	spgirl@mediba.jp

食い込む敷居はどんどん高くなっています。この状況で上位を狙うのは決して簡単ではありません。時代の流れに身を任せることが必ずしも成功の法則とは限りません。ときには時代の流れに逆行することで成功のチャンスをつかむこともあるのです。

無料化プロモーション戦略

プロモーションとして有力な手段の1つに「有料アプリの無料化プロモーション」というものがあります。これは早い話、期間限定で有料アプリを無料にすることで注目度を上げるという手法です。これは最初から無料アプリである「基本無料」のアプリではできませんが、有料版と無料版が分かれているアプリにとってはかなり有効な宣伝方法になります。無料化アプリのプロモーションを専業にしているメディアも数多くありますし、多くのアプリ紹介メディアにはセール紹介コーナーがあり、必然的に期間限定無料化アプリが優先的に紹介されることが多いのです。

ただし、無料セールは当然リスクも付きものであることを理解しなければなりません。有料アプリを無料化するわけですから、その間ダウンロードされたぶんに関しては収益は生まれません。さらに、無料化すれば必ず注目されるというわけでもなく、これもまた成功するときと失敗するときの波が激しいのです。タイミングを狙うのも簡単ではありませんが、できるだけほかに注目の無料セールアプリが出ていないところを狙うのがポイントでしょう。年末のホリデーシーズンなど、大手メーカーのセール合戦が始まる時期は控えるべきです。

無料セールは犠牲バント

そして、この無料セールの一番の目的は、目先の利益を生むための戦略ではないということも十分理解しなければなりません。無料プロモーションで獲得したユーザは、既存の別アプリ、または将来のアプリへとつなげる犠牲バントと考えることが重要です。もちろん、無料プロモーション中にダウンロードしたユーザが口コミで広げてくれる可能性もありますし、そのアプリが結果的に伸びることも十分あり得ます。ただ、短期的に数万ダウロードされたとしても、プロモーションが終わって数日もすると販売数は元の勢いに戻ってしまうことのほうが多いので、それをプロモーションの失敗と考えないようにしましょう。ユーザがアプリを使い続けてくれる限り、新作アプリを出したときに以前のアプリからの流入を生めることを忘れてはいけません。この流れをうまく利用して雪だるま式にユーザを増やしていけば、新作を出すときのヒット率がどんどん高まります。

野球も1人の選手だけならホームランを狙うしか得点を得る術はありませんが、チームメイトを充実させることで確実に点に結びつけていくことができます。つまり、アプリ開発で成功を収めるためには、それなりにアプリのラインアップを増やしていくことも大きな課題と言えるでしょう。

他のアプリへの誘導手段

無料セールのプロモーションによって自作の他アプリへ誘導するしくみについて書きましたが、無料セールをすれば勝手にほかのアプリへ流れて行くわけではありません。事前に、または後からでも誘導するための手間をかけなければなりません。方法はいくつかあります。

一番ユーザにストレスがないのは、アプリ内でApp Storeを開ける「SKStoreProductViewController」を導入する手法でしょう。ユーザはアプリを離れることなく、その場で有料、または無料のアプリをダウンロードすることが可能になります。難点は、導入はさほど難しくないものの、フレームワークの追加や少々コードを加える手間があることと、サーバ側から操作ができるしくみがない限りは、いちいちアップデートをAppleに提出しなければいけないことです。新作のリリースと同時のタイミングで流入を狙いたい場合は、新作とほぼ同時進行で旧作のアッ

プデート作業も行う必要があり、時間のロスが生じてしまうのがネックと言えます。

広告付きの無料アプリであれば広告枠を使って、広告会社のサーバを通していつでも自由に自社広告を配信することが可能です。新作が出たときにいちいち以前のアプリをアップデートしなくてもすぐに知らせることができるメリットがある反面、普段も広告が表示されている位置なので、他の広告との区別がつきづらく、同じアプリ開発者の新作アプリであるということが伝えづらいデメリットもあります。

どちらの方法で誘導するにせよ、ユーザの目に付きやすい形で知らせることが重要です。手間をかけても相手にされなければ意味がないので、しっかりアプリからアプリへとつなげていくことを意識しましょう。

セルフブランディングのススメ

アプリを広めるための知恵第2弾は、セルフブランディングのお話です。

アプリを作ることと売ること、これは両翼の関係にあり、どちらが欠けてもアプリビジネスシーンでは大きく羽ばたくことはできません。アプリを多くの方に知ってもらうための手段の1つとして、筆者(くまんぼう)はセルフブランディングをお勧めしたいと思います。また、セルフブランディングを活用し人に覚えてもらうということは、アプリ開発者としてもさまざまな運やチャンスを引きよせることにもつながります。

セルフブランディングとは

セルフブランディングとは文字どおり「自分自身」を「ブランド化」するためにプロデュースすることです。このように言うと難しく聞こえるかもしれませんので、もう少しやわらかく「自分を覚えてもらいやすくするためのイメージ作り」だと考えていただいてかまいません。

なぜセルフブランディングが必要なのか

アプリを知ってもらう手段の1つとして

目的はあくまでアプリを知ってもらう、手にとってもらうことなのに、なぜ制作者自身を知ってもらうことを勧めるのでしょうか？　ひとつ想像してみてください、商店街に同じ商品を扱うお店が2つあり一店はあなたの知り合いが経営する店舗で、もう一店はまったく知らない方の店舗だとします。どちらの店舗で商品を購入するでしょうか？　きっと、知り合いの店舗で購入してあげようとする気持ちが働くのではないでしょうか？

また、絵や歌などある作品を購入しようかどうしようか迷っているときに、その作品の制作者のキャラクターを知っている場合と知らない場合とでは、どちらが最後の瞬間に購入するという決断を下す確率が高いでしょう？　あなたが気になっている方が何か作品を出したとしたら、どんな作品なのか見てみたくなるのではないでしょうか？

このように、制作者としての自分を知ってもらうというのはアプリをダウンロードしてもらう最後の一押しや、知ってもらうためのきっかけとして大きな力になり得るのです。100万種類を越えたと言われているアプリ市場で自分のアプリを知って、選んでもらうための最後の決め手として有効なのです。

もちろんアプリ自体がきちんと魅力的なものになっていることは最低条件です。しかし市場には同じように魅力的なアプリが星の数ほどあるということを忘れないでください。その綺羅星のようなアプリ群の中から自分のアプリに興味を持ってもらう、そのための手段の1つとしてセルフブランディングがあるのです。

次のアプリへの橋渡しに

また、筆者が意識していることの1つとして「自作アプリ同士のハブとして自分が機能できるように」というものがあります。アプリをいくつかリリースしている場合、そのうちの1つをダウンロードして気に入ってくれたユーザに、自作の他アプリへと導く橋渡し役を自分がするということです。

アプリ内に他アプリへのリンクを設置するなどアプリ同士を直接つなぐ方法もありますが、これは誘導元になるアプリ使っているときにしか機能しません。せっかくあなたのアプリに興味を持ってくれていたとしても、新作アプリをリリースするまでの期間が空いてしまって誘導元のアプリが起動されなくなっていたり削除されてしまうと、橋渡しができなくなってしまいます。

しかし、制作者自身に興味を持っていただき、TwitterといったSNSや自身のブログなどをチェックしてもらえるようになれば、たとえ前作のアプリが削除されていても新作リリースのつぶやきやブログの記事はそのユーザに届くでしょう。

どうやって人に覚えてもらうのか

それでは、実際にどのような方法で人に覚えてもらいやすくするのでしょうか？　それはセルフブランディングの言葉どおり、自分を「ブランド化」することです。ブランドといってもアプリの値段を上げて高級感をだすということではありません。始めに言ったように「ブランド化」というのは「わかりやすいイメージを作り出す」ことなのです。その方法について説明したいと思います。

ブランドとは約束

たとえば「あの人のアプリはパズルゲームが多いな」「やたらハイスピードでアプリをリリースする人だな」と思ってもらえたらそれはすでにブランドなのです。また、直接アプリと関係がなくても「あの人のブログを見に行くと、いつも美味しそうなラーメンを紹介しているな」「いつもTwitterでおもしろいこと言ってるな」などでもいいのです。

つまり、ある事柄を期待してもらえるようになること、それが「ブランド化」なのです。これを言い換えると**ブランドというのは約束である**とも言えます。「あの人のアプリはパズルゲームが多い」「あの人のブログは美味しそうなラーメンを紹介する」これはユーザ皆さんとあなたの間の「約束」なのです。

たとえばバッグや洋服の高級ブランドメーカーの場合は「あのメーカーの商品は質がいい」という約束ですし、「値段が高い」というのも約束です。「値段が高い」という約束が成り立っているので、そのメーカーの商品を身につけることが他の人から見ても「高級な物」を身につけているとわかることの保証にもなっているのです。

それではこの約束を伝えていくにはどうすればよいでしょう？

自分自身がコンテンツになる

約束を伝えるには、やはりアウトプットし続けるのが王道だと思います。あなたの思い、アプリに込めたメッセージやコンセプトはアウトプットしたほうが伝わるのは言うまでもないでしょう。そしてアウトプットは1回ではなく、継続的にアウトプットし続けたほうが伝わりやすいです。

ただし、必ずしもアプリのコンセプトなど難しい話をしなければいけないわけではありません。先ほどの例にも挙げたとおりラーメンの話でもいいですし、そのほかのあなたが興味を持っていて継続して情報を提供しやすいものでもかまいません。ただ、あまり多岐にわたるとイメージがぼやけてしまい、結果として覚えてもらいにくくなってしまうので注意しましょう。

そして、できればあなたの人となりが伝わる

ような内容を心がけてください。アプリ制作以外であなたがしている活動や興味を持っていることがいいでしょう。人となりが伝わったほうが親しみを持ってもらいやすくなります。自分自身がひとつのコンテンツになるのだということを意識するとアウトプットしやすいかもしれません。

アウトプットする場所としては、やはりネット上がアプリビジネスとは親和性が高くかつ低コストです。ネット上の活動ですぐに始められるものとしてブログとSNSがあります。

▶ブログ

アプリ制作をするのであれば、自分の制作したアプリの情報をまとめておくサイトを作っておくといいでしょう。あなたのアプリを検索してそのページにたどり着いた方は、必然的に他のアプリの情報も目にすることになります。ブログとしてサイトを作り、そこに先ほど言ったようなアウトプットを記事としてしたためておけば、自然にあなたのアプリとあなた自身の人となりが結びつくことでしょう。

また、ブログは記事を書けば書くほど蓄積していき、検索を通してあなたのサイトを訪れる人が増える可能性が高くなっていきます。あなたの人となりを記載した、いわばあなたのパンフレットがどんどん厚くなっていくようなものなのです。

▶SNS

TwitterやFacebookなどのSNSを活用するのも有効です。新作アプリのリリース告知やブログ更新の情報はもちろん、普段の何気ないつぶやきからあなたの人となりは伝わっていくことでしょう。また、アウトプットだけでなくあなたのアプリについて書かれているつぶやきを検索して、アプリに対する反応をリサーチするインプットのツールとしても有効です。

ユーザと直接メッセージをやりとりすることも可能でありブログより双方向性が強いので、人となりを伝えるには適しているメディアといえます。

ブログは蓄積のメディアで、SNSはスピードのメディアと言われるように、2つのメディアの特性をうまく組み合わせることによってその効果は何倍にもなることでしょう。

筆者ももちろんブログ、Twitter、Facebookのいずれも活用しています。ブログ[注1]では自作アプリの情報、大学講師の活動など仕事の話、TVやラジオなどに出演させていただいたときのこと、そして旅行記など、私という人間が何をしてどういうことに興味を持っているのかをわかっていただきやすいように考えて記事をまとめています。

1つとくに意識しているのは、何となく「行ってみたい」や「やってみたい」と思ってはいてもなかなかチャンスがないようなことは積極的に参加し記事にすることです。こうすることで、同じように思っている方に興味を持ってもらいやすくなるようにしています。筆者の記事では「ドバイ・アブダビ旅行期」や「飛鳥乗船記」などがその最たるものです。

実際TVでドバイや飛鳥が紹介されると、検索で筆者のブログにたどり着く方が一時的に増えます。そうするとやはりその日にTwitterのフォロワー数も増加します。Twitterを見てもらえるということは新作アプリの告知も今後その方たちに届けることができるということです。筆者のアプリにまったく接点がなかった方たちが筆者のアプリのユーザになってくれる可能性が出てくるのです。

わかりやすいビジュアルを作る

さて次は、これまでお話ししたブランドイメージをより伝わりやすく、そして強烈に覚えてもらうための方法についてお伝えしたいと思います。それはビジュアルをうまく使うことです。

注1）http://ninebonz.net/

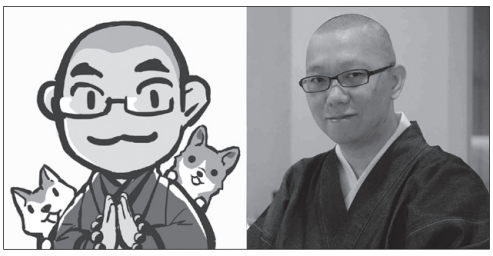
▶100人集まる勉強会で見分けが付くのか？

ここでまたひとつイメージしてみてください。開発者が大勢集まる勉強会やアプリレビューサイト主催の集まりなど100人以上の人が集うイベントで、あなたと会ったことない方、つまりあなたの顔を知らない方があなたを見つけることができるでしょうか？　普通に考えると「そんなことは無理だ」と思うでしょう。しかし、筆者は自信を持って「私のことは見つけてもらえる」と言えます。それはなぜかというと筆者は普段から「和服」を着て活動しているからです。もともと和服が好きでよく着ていたのですが、ここ数年は人前に出るときには意識して着物を着ています。IT系の集まりで和服の人というのはほぼいませんので大変人目を引きます。また、Twitterなどのアイコンも着物を着た自分の似顔絵で統一しています（**図1**）。

こうすることによって、直接会ったことがない方でもTwitterのアイコンなどを見たことがある方は「ああ、あの人だ」と見つけてもらえる可能性がグッと上がるのです。ちなみに筆者がこのビジュアルの力を一番実感したのは、電車の中でほかの乗客から突然「くまんぼう和尚ですよね？」とお声かけいただいたときでした。このときは自分が意識してきたことが少しずつ成果を出していることがわかり嬉しかったです。

もちろん奇抜な格好をしましょうと勧めているのではありません（着物自体はそれほど奇抜ではありませんが、IT系の世界では異質と言えるでしょう）。自ブランドのロゴなどをデザインしてそのロゴをあらゆるところで露出するのもよいでしょう。なんでもいいのでキービジュアルとなるものを決めて、そのビジュアルをアウトプットし続けることで効果は必ずあります。

▼図1　SNS上のアイコンと筆者



▶アプリは知っていても制作者は知らない

ビジュアルの力を得るためにはどんな方法があるでしょう？　筆者がお勧めしたいのはアプリのアイコンやイメージを有効に使うことです。たとえば自分のアプリのアイコンやアプリ内の印象的なグラフィックをプリントしたTシャツを作成し、人前に出るときにはそれを着用するはいかがでしょうか？（**図2、3**）

そういえば大ヒットゲーム「AngryBirds」のROVIOの方にお目にかかったときも、制作者の方はあの赤い鳥がプリントされたパーカーを着用されていました。

Tシャツなどを用意するのが大変であればもっと簡単な方法もあります。アプリのアイコンなどを印刷し、カードホルダーに入れて首からかけておくだけでも「ああ、あのアプリの作者なんだな」と認識してもらえるはずです。日常的には難しいかもしれませんが、先に挙げた集まりなどでは雄弁にあなたがどのアプリの作者なのか語ってくれることでしょう。

それさえもめんどくさいという方は、何でもいいので特徴的な物を1つ身につけるようにするのもよいでしょう。テンガロンハットや胸ポケットにシルクのハンカチでもとにかくなにか「○○の人だ！」と言われるような物を設定してみましょう。

売るだけではなく、作ることへの効果も

さて、ここまでアプリを広める、売ることについての効果を説明してきましたが、人に知ってもらうということはアプリ開発自体へのプラス効果も期待できます。あなたがどんなアプリをリリースしていて、何ができる人なのかを知ってもらえば、そういったアプリを作りたい会社や、同じようなアプリを作っている開発者から

コラボレーションなどの声がかかることもあるのです。

実際に筆者も見つけてもらいやすいように意識し始めてからインタビューやTV、ラジオなどのメディアへの露出のチャンスも増えました。そしてそれが芸能人をフィーチャーしたアプリを作るチャンスを引き寄せ、おかげさまでそのアプリはApp Storeの無料総合ランキング1位をいただくことになりました[注2]。

「利益もチャンスも人が運んできてくれる」という言葉があります。人に知られるということは、それだけ縁やチャンスを引き寄せられるようになることなのです。そして、この連載だってそんな出会いの中でつながったチャンスの1つなのです。今回で無事最終回を迎える当連載ですが、このようなチャンスを与えてくださったSoftware Design誌、そして編集担当のK氏には大変感謝しており、出会えてよかったと心から思います。

知られなければ「居ない」と同じ

厳しい言い方かもしれませんが「知らない」ということは「居ない」と同じことなのです。自分の中の「強み」を見つけそれをアピールすることを恐れないでください。自分の長所をブランド化して、多くの方にあなたのアプリを届けましょう！SD

▼図2　脱出ゲーム「DOOORS」とアプリのイメージがプリントされたTシャツ

▼図3　電卓アプリ「Llumino」とアプリのイメージがプリントされたパーカー

注2）参考「ある個人開発者がApp Store総合ランキング1位を獲得するまで」http://ninebonz.net/thx-all

本連載のサポートページで記事の補足説明をしています。あわせてご活用ください！
› http://www.gimmiq.net/p/sd.html

リオ・リーバス／Leo Rivas
Twitter @StudioLoupe

iOSアプリ開発を中心に電子絵本作家・漫画家として活動中。代表作は、KDDI株式会社に社内導入され、世界で40万ダウンロードを記録する革新的な電卓アプリ「FusionCalc」と、国連主催のWSA Mobile 2013を受賞した、顔の動きで電子書籍が読める「MagicReader」。電子絵本はiBookstore/Kindleストア共に児童書カテゴリ総合1位を獲得。

いたのくまんぼう／Itano Kumanbow
Twitter @Kumanbow

神奈川工科大学非常勤講師。リオさんとはGimmiQ名義で「MagicReader」（手を使わずにページがめくれる電子書籍ビューワ）をリリース。個人ではNinebonz名義で「江頭ジャマだカメラ」（無料総合1位獲得）、「i列車の車窓から―そうだ! 京都行こう!―」（バーチャル旅行アプリ）などをリリース。アプリ紹介サイト「あぷまがどっとねっと（http://appmaga.net/）」の技術サポータ。

第47回 Esenthel Engineを使ってみよう

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！

嶋崎 聡　SHIMAZAKI Satoshi
よこいど／日本Androidの会 横浜支部
Twitter @sato_c　Mail motosumi64@gmail.com

C++で書ける3Dゲームエンジン

横浜支部の嶋崎です。「Esenthel Engine[注1]（以下、EE）」という3Dゲームエンジンを今回は紹介します。現在このジャンルでは、手軽さや情報量で圧倒的に「Unity[注2]」がメインという感じがあります。EEはまだまだマイナーなゲームエンジンですが、もしかしたら……という期待を込めて紹介していきます。

Esenthel Engineの特徴

EEの開発環境（IDE）はWindows/Mac OS/Linuxで動作します。ターゲットはAndroidとiOSを足して5種類となっています。iOS環境についてはコンパイラの都合でMac OS上での開発に限定されますが、Androidはすべての環境で作成可能です。

ライセンスは199ドルです。ライセンス登録を行わなくても無料で試用できますが、数分おきにダイアログが出ますので、少しうっとうしいかもしれません。EE本体のダウンロードやライセンスの支払いはサイトのSTOREから行います。STOREはUnityのAssetStoreと同じようなもので、自分で作ったアプリケーションもSTORE経由で配布できます。配布には開発者登録が必要になります（有料。29.99ドルかかります）。STOREにはこれ以外にデモやソース、各種素材などの参考になるファイルが販売されています。

C++と独自スクリプトを使った開発

EEを使った開発はC++そのままの記述とEsenthel Scriptという独自スクリプトで行います。スクリプトといってもコード記述支援スクリプトで、記述自体はC++に近いのであまり意識せずに書けます。C++コードへの変換も随時行われますので、プログラマが変換作業を意識せずIDEの実行ボタンやビルドボタンですぐにビルドが始まります。

また、このIDEには非常に高速なコード補完機能を持ったエディタがついています。引数などを忘れてしまっても、関数名を数文字叩けば候補とともに引数も表示されます。引数の名前も本来のヘッダで宣言されているものが出ますので、違う名前が出てきて混乱することもありません。

2014年2月に、フォント設定にCustom項目が追加されました。フォント設定ダイアログにあるチェックボックスを選択すれば、日本語表示できます。

C++コンパイラはEEには含まれません。各OS用の実行ファイルを作成する環境が必要で

注1）http://www.esenthel.com/
注2）http://japan.unity3d.com/

す。Android用の場合はAPKを作るためにAndroid SDK/NDKとJDKをインストールしましょう。Windows用実行ファイルを作る場合はVisual Studioが必要です。Express版でも動作を確認していますので、まず試してみたいという方にはお勧めです。Mac OS/iOS用の場合はXcode、Linuxではgccが必要になります。

ビルドするには、EEのPATH設定で各コンパイラへのPATHを設定します。一度設定すればEEで作ったプロジェクトのビルドはEEがやってくれます。Androidアプリの場合はNDKを使ってビルドされます。通常NDKでアプリを作る場合、ツールのインストールや設定、Android.mkを作ったビルドなど面倒な作業が多くありました。しかし、EEはPATHの設定だけで自動でAPKを作ってインストールまでしてくれるので、非常に楽で驚きました。

EEがC++で開発できる利点に、C++用として公開されている各種ライブラリをとくに改造しなくても利用できる点が挙げられます。筆者はTwitterで教えてもらったLinq for C++[注3]ライブラリをコンパイルして動かしてみました。Windowsではすぐに動きましたが、Androidでは`-std=c++11`と指定してC++11設定が必要になるので、EEの出力するApplication.mkに追加する方法を調べていて、まだ試せていません。

また、PC用のLeap Motion[注4](コラム参照)のようにライブラリが提供されているハードウェアならば、リンクしてすぐに使えます。2014年1月のアップデートで、外部ライブラリの設定が各プラットフォームごとにできるようになりました。

プログラマ向けの環境

Unityも使っている筆者が感じたのは、EEはどちらかというとプログラマの視点に立ったツールだということです。Unityはデザイナー視点でコードよりも実行画面やデザイン画面がメインになっていましたので、いきなりEEの画面を見ると取っつきづらいと感じるかもしれません。どのような部分がプログラマ視点かというと「まずコードを書く」必要があることです。新しいプロジェクトを作成しても基礎となるファイルすら用意されません。このあたりは最低限のスケルトンはあったほうがいいと思うのですが、現状ではチュートリアルから基本的な関数をコピーしてくるのが一番手っ取り早いです。そのうち改善されるといいのですが。

IDEにはエクスプローラ風のファイル選択ウィンドウもありません。プロジェクトツリーで右クリックして新たなファイルを作成していきます。モデルやモーションなどのファイルはエクスプローラからプロジェクトツリーにドラッグ&ドロップして追加します。

EE独自の機能として、プロジェクトメンバー間の連携開発機能があります。1台のPCをプロ

注3) http://cpplinq.codeplex.com/
注4) https://www.leapmotion.com/

Column
Leap Motionとは?

LeapMotion社から発売されている(日本国内ではBBソフトサービス㈱が販売)、手の動きを読み取るコントローラです。たとえば、マウスカーソルを空中の指の動きでコントロールしたり、PC画面に映っている物体を直接手で触るような操作も可能になるデバイスです(**写真1**)。

使い方の幅が広いので「これ」と言えるような決定的な操作がないので説明がしづらいですが、手のひら、指といった「手」の動きはほぼすべて取れますので、本当に自分の手が画面のなかで動くような感覚で再現できるという意味では、他のデバイスに比べて直感的に利用できるものと言えるでしょう。

▼写真1 Leap Motion

ジェクトサーバとして起動して、他のメンバーはそこからプロジェクトを取得。各自の手元で編集したソースやモデルなどはサーバ側と同期できます。サーバ側とクライアント側の差分は、自動的にマージしてくれますのでバージョン管理システムを導入していない環境でもメンバーが複数人いる場合に便利でしょう。

Android用アプリを作る場合には、なかなか良いデバッグ環境がないのが悩ましいところではないでしょうか。EEの開発環境ではADTで利用するデバッガなどは使えませんが、ライブラリが共通なので、まずターゲットをWindowsやMac OSなどにして作成・デバッグを行ってから、ある程度動いたらAndroid用に調整するような形で開発するといったことができます。こうするとVisual StudioやXcodeのソースレベルデバッガでデバッグできるのでかなり便利です。

エンジンの紹介代わりのチュートリアルが豊富

EEには最初からたくさんのチュートリアルが用意されています。ファイルの読み込みやテクスチャの表示といった簡単なものから、実際にMMORPGのベースに使えそうなサンプルまで100を越えるファイルがあります。インストール直後に登録されているプロジェクトは、このチュートリアルだけです。初めてEEを起動したときにこのチュートリアルが登録されているだけでも実際のところ安心感がありました。ゲームエンジンに限らず、たいていの開発環境ではサンプルプログラムは用意されています。基本的な機能の説明で終わっているものや中身を理解するのが大変なほど規模の大きいものも多い中で、細かい機能を確認して応用できるサンプルが豊富なのは、開発を進めていくうえでも役に立ちます。ただ、提供されている機能は多岐にわたるため、チュートリアルでライブラリの関数が完全に網羅されているわけではないのが残念です。

EEのドキュメントはVer.1がほとんどでしたが、2014年に入ってからサイトの内容が見直されて、Ver.2用ドキュメントと動画が毎週毎週追加更新されていきます。動画の解説は英語ですが、操作に関しては画面も映っていますので使い方を確認するのに役立ちます。

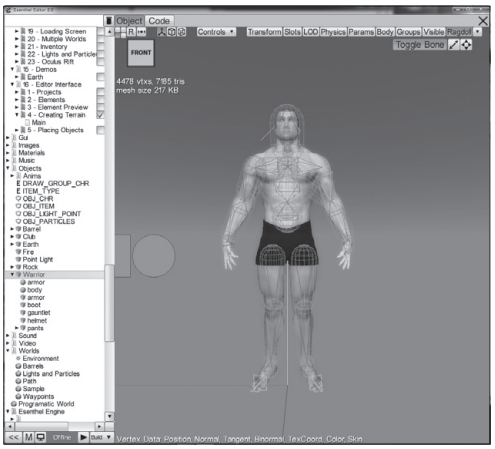
チュートリアル以外でも、自分で作るプロジェクト内でライブラリのヘッダを確認できます。プログラムを書いている途中などいつでも確認できるので便利です。“この関数の使い方がわからない→検索しても情報が出てこない→もうダメだ”といった流れになりがちですが、ヘッダを調べてやりたいことを探せる資料が用意されているということです。

ほかにもEE内でのタッチパネルの扱いなど、スマートフォン向けアプリを作る際の参考になるサンプルがありますので、一通りチュートリアルに目を通してみることをお勧めします。3Dに関するサンプルもAndroid端末で動作することが確認できました。PC用の入力系をスマートフォン用に変更して操作できるようにするにはタッチ操作へ改造する必要がありますが、PCではタッチ操作をマウスに置き換えるという機能もありますので、まずPCで試してからAndroid端末で動かすという手順で大丈夫でしょう。

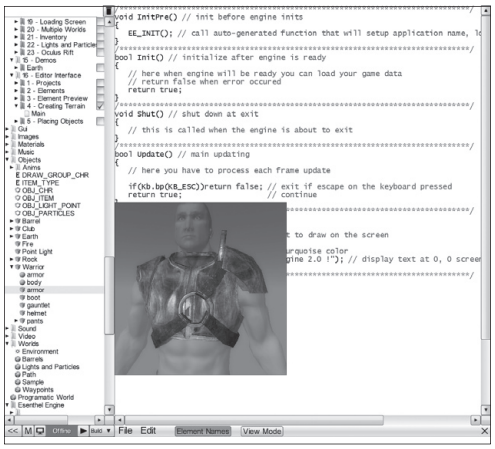
各種エディタと独自GUI

エディタは、テキストエディタ以外にモデルエディタ、ワールドエディタなどがあります。モデルエディタは読み込んだモデルのマテリアルや見た目の確認、調整を行うためのものです。ラグドール設定やアイテムスロット設定などもここで行います。ラグドールは、アニメーションデータではなくモデルにつけている当たり判定を元に物理演算で動きを付ける機能で、キャラクタがやられたときのモーションなどで利用されています(**図1**)。アイテムスロットは、モデルの体につける鎧などの装備を複数用意しておいて、プログラムで簡単に着脱や装備変更をするための機能です(**図2**)。なお、モデルエディタではモデリング自体はサポートしていませんので、別途モデリ

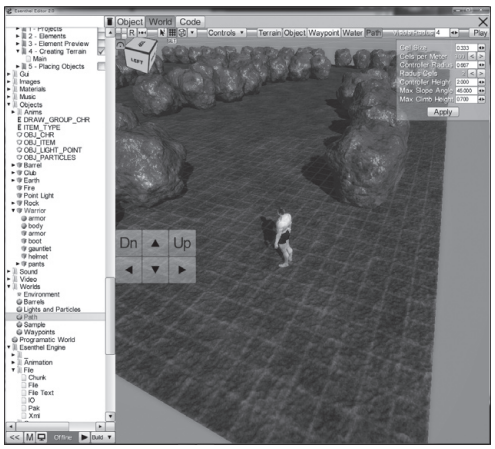
▼図1　ラグドールの設定画面



▼図2　スロット用装備の確認



▼図3　ワールドエディタ。岩と人間が配置されている



ング用ツールが必要になります。

ワールドエディタはゲーム世界の構築用エディタです(**図3**)。世界の広さをブロック単位で指定して、そこに地形用のマテリアルやさまざまな物体を置いて世界を構築していきます。山や谷といった自然の地形を作るのに、マウスの左右ボタンを使って調整をします。また、河や湖といった水の表現をする機能や、MMORPGなどで地形をクリックするとキャラクタがその場所まで歩くようにする機能などもあります。ただ、筆者が設定方法などを調べ切れていないので「すごいのかもしれない」という感想しか残念ながら書けません。水の表現は海の上を走るモデルの実現に使いたいと思っていますが、もう少しかかりそうです。

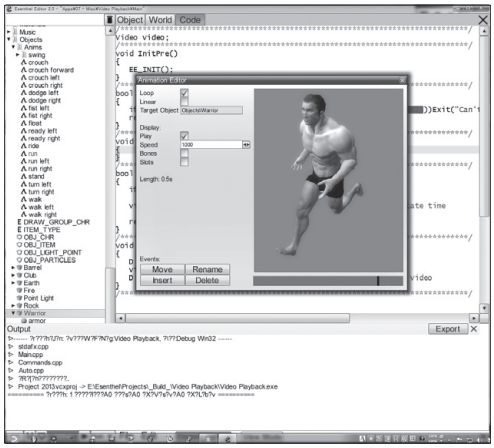
それから、EEにはゲーム中に使えるGUI部品が独自に用意されており、ダイアログの表示やボタン、タブ切り替えというGUI構築に必要な機能も一通りそろっています。このGUIのデザインを見ていると、EE自身もこのGUI部品を使って実装されているように見えます。チュートリアルの最後に「Editor Interface」という項目もあって、自分でエディタの拡張ができるようです。

モデルデータの流用も

昨今のゲームでは、どうしても3Dモデルやモーションを使う必要がでてきますが、EEではfbxなどの3Dモデルデータが読み込み可能となっています。fbxフォーマットへの出力はモデリング用ツールが何らかの形でサポートしているのでとくに困ることはないと思います(コラム参照)。

ゲームのプレイヤーキャラクタでは、ボーンの各部位の名称を、EE側であらかじめ決められているものにすることが推奨されています。合わせておくとEEのゲームキャラクタオブジェクトを利用したときに、プレイヤーの入力に応じてモーションをオブジェクト側で管理、切り替えができるようになります(**図4**)。歩いているモーションから走るモーション、止まるモーションへとジョイパッドやマウスの入力に応じて動いてくれるので、自分でモーションの合成

▼図4　モーションの確認ができる



などを行わなくてもよくなります。実際にゲームで使うキャラクタはそれぞれにモーション調整が必要になると思いますが、プレイヤーキャラクタのタイプを増やす場合にある程度モーションを共有できるかもしれません。

Column

MikuMikuDance用データの流用

国内でオンライン配布されているモデルデータを使おうとするとMikuMikuDance(以下、MMD)で作られたデータ(PMXやPMDといった拡張子)が多いので、これをfbxに変換するために別のツールを紹介します。

MMD4Mecanim注というUnity用パッケージに同梱されているPMX2FBXです。このツールはMMDで利用されているモデルやモーションをfbxにコンバートします。できあがったfbxデータをEEにドラッグ&ドロップするとプロジェクトに登録されるので、モデルエディタで頂点カラーを削除すれば、見た目もちゃんとしたモデルになります。モデルによっては、マテリアルカラーやテクスチャの調整も必要になりますが、だいたいの場合は頂点カラーの削除だけで大丈夫でしょう。

注) Noraさん作のUnity用パッケージ。http://stereoarts.jp/よりダウンロード可能。

ネットワークゲームへの対応

EEは3Dのゲーム向けエンジンなので、RPGやFPSといった箱庭型ゲームを製作することも意識して作成されています。実際にEEのSTOREでは3DRPGのソースが何種類か販売されています。

ネットワーク用関数もクライアント、サーバ共にEEの標準的な関数として用意されています。データベースのアクセスを行う関数も用意されていますので、あとはデータサーバを用意すれば、ネットワーク対応のRPGやFPSなども構築できるでしょう。

先進的なデバイスへの対応

EEはゲームエンジンの値段としては割と安価な部類に入ると思います。その安価なエンジンの割に各種スマートフォン、Oculus Rift(コラム参照)などへの対応がプラグインではなく、本体側で行われているためにさらにお得感があります。Unityの場合はAndroid/iOS向けの出力は別途ライセンスが必要になりますし、Oculus Riftを利用する場合、現状ではネイティブプラグインを使うためPRO版が必要になります。

Unityのようなメジャーな環境では、新しいデバイスが出てきたときに最初からUnity用プラグインとして対応が行われています。EEでは、最初から対応されていることはほとんどありませんが、C/C++用ライブラリがあればおおよそ動くでしょう。前述したLeap Motionも、外部ライブラリの利用について調べるついでに動けばいいなと思って試した程度でしたが、プロジェクト設定にヘッダファイルやライブラリの場所を書いてリンクするだけで使えました。あっさり動いたのでびっくりしたくらいです。

Android向け環境について

Android用ネイティブアプリを作るにはJavaが必要になりますので、敷居が高いと思っている人

もいると思います。筆者もAndroidアプリを作り始めたころはJavaが必須でしたから、使い慣れていないJavaを一生懸命覚えました。しかし、EEはC/C++を知っているプログラマならばサンプルを少し読むだけでも使えるようになるでしょう。NDKでC++を使った開発はできましたが、ビルドして動かせるようにするまでの準備が必要で手間がかかります。EEはNDKを使ったビルド設定を代わりにやってくれるので、プログラムを書いたらビルドボタンを押すだけになり、手間が減りました。

Android向けにアプリを作った場合にどれだけ機能が制限されてしまうのかについては、筆者も現在いろいろと試しているところです。初代Galaxy Note、SH-02E、Nexus 7(2012)など手持ちの端末で動作テストをしている限りは、3万ポリゴンのモデルの単体表示では不満のない速度で描画しています。今後、モデルを増やす、入力判定など処理が増えたときにどうなるかわかりません。ただ、複数のモデルを動かすなら、3,000ポリゴン程度までモデルを簡略化しないと厳しいかなと思っています

まとめ

昨年末にEEを知ってから、年末年始の休みの間もいろいろと試してみました。地形の編集や水の表現など、魅力的な部分がたくさんあるのですが、実際に動かすにはプログラムも書かないといけないらしく、そういった情報を集めるのに苦労しています。EEの歴史は割と古いようですが、Ver.1の頃のライセンス形態が1タイトル1ライセンスだったこともあって日本ではあまり知られていないようです。Ver.2からは1ライセンスで何本でも作っていい形に変更されたので、これから使う人が増えてくるのではないでしょうか。

ゲームエンジンとして評価するには、せめて2〜3本作ってからと思うのですが、今回、EEを少し使ってみただけでもいろいろとおもしろいと感じたので紹介しました。ほかのゲームエンジンと比べてもおもしろい機能が豊富です。

C++を使える方でAndroidアプリを作りたかった方は、ぜひ試してみてください。SD

Column
Oculus Riftとは?

Oculus Rift(以下、Rift)はOculus VR社[注]が開発しているVirtual Reality用ヘッドセットです。2014年1月現在、開発者向けキットの初期バージョンが販売されています。2014年のCESで次期バージョンが発表されました。まだまだ開発者向けキットの段階ですが、300ドルという値段も手伝って話題になっています。すでに発売されているFPSやUnityを始めとした各種ゲームエンジンがプラグインなどの形で対応しているため、手にした日から、すぐにRiftのすごさを体験できるというのも人気の一因でしょう(**写真2**)。

秋葉原では定期的にOcufesと呼ばれるRift用アプリケーション開発者によるイベントなども行われていますので、興味がある方はTwitterのハッシュタグ#ocufesを定期的にチェックしてみてください。

注) http://www.oculusvr.com/

▼写真2　筆者の所属会社で購入したOculus Rift

嶋崎 聡(しまざき さとし)　よこいど／日本Androidの会 横浜支部所属

Androidでゲームを作りたいがために長年敬遠してきたJavaを覚えたまではよいのですが、使い始めたときの環境(Android 1.6)では描画とGC(と自分自身のスキル)のおかげで求める速度が出せないまま、いつのまにか当初の目的を忘れて全然違うものばかり作っています。そろそろゲームを……。

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18回

FreeBSD 10.0-RELEASE公開記念 10.0-RELEASEで学ぶbhyveの使い方

Writer 浅田 拓也（ASADA Takuya）Twitter：@syuu1228

はじめに

bhyveをマージしたFreeBSD 10.0が2014年1月20日についにリリースされました。今回はこれを記念し、仮想マシンのネットワークデバイスについて解説する予定を変更し、bhyveの使い方を復習します。

bhyve とは

bhyveはNetAppにより開発され2011年に発表された、新しいハイパーバイザです。設計はLinux KVMに似ており、大雑把に説明すればKVMのFreeBSD版と言えます。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を流用しているため既存OSとの高い互換性が得られています。しかし、コードの見通しが悪くなってしまっているのが難点として挙げられます。一方、bhyveでは独自に一から実装したユーザランドにより、シンプルで見通しのよいコードを保っています。

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[注1]。

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。

ただし、FreeBSD以外のゲストOSを動作させるためには、次のような問題が確認されています。

- VMwareによるNested VM環境へLinuxをゲストOSとして起動しようとすると、タイマ周りの問題でLinuxカーネルがフリーズする
- 古いCore iシリーズ（Nehalem）はVMでリアルモードをサポートする機能をサポートしないため、grub2-bhyve（後述）でLinuxやOpenBSDカーネルを起動できない
- 一部のバージョンのLinuxカーネルと一部のIntel CPUの組み合わせで非サポートMSRのアクセスによるエラーを起こしてbhyveが異常終了する（後述）

また、bhyveを実行するにはamd64版のFreeBSDを使用する必要があります。32bit版ではサポートされていません。

bhyve が提供する機能

現状のbhyveでは限られたOSをゲストOSとして実行できる最低限の機能だけが実装されています。

注1）svnブランチで開発が進められています。（http://svnweb.freebsd.org/base/projects/bhyve_svm/sys/?sortby=date&view=log）

ハードディスクコントローラとしてはAHCIコントローラのエミュレーションと、準仮想化ドライバであるvirtio-blkをサポートしています。NICコントローラは、準仮想化ドライバであるvirtio-netをサポートしており、Intel e1000のような標準的なデバイスエミュレーションはサポートしていません。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ドライバがインストールされていて起動時に使えるように設定する必要があります。

システムコンソールとしてはPCI接続の16550互換シリアルポートをサポートしており、標準的なビデオデバイスはサポートしていません。また、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ものだけですが、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。

このほか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ゲストOS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CPUに対してVT-x命令を発行するなどハードウェアに近い処理を行うカーネルモジュール(vmm.ko)と、ユーザランドにおいてユーザインターフェースを提供しハードウェアエミュレーションを行うVM実行プログラム(/usr/sbin/bhyve)の2つからなります。

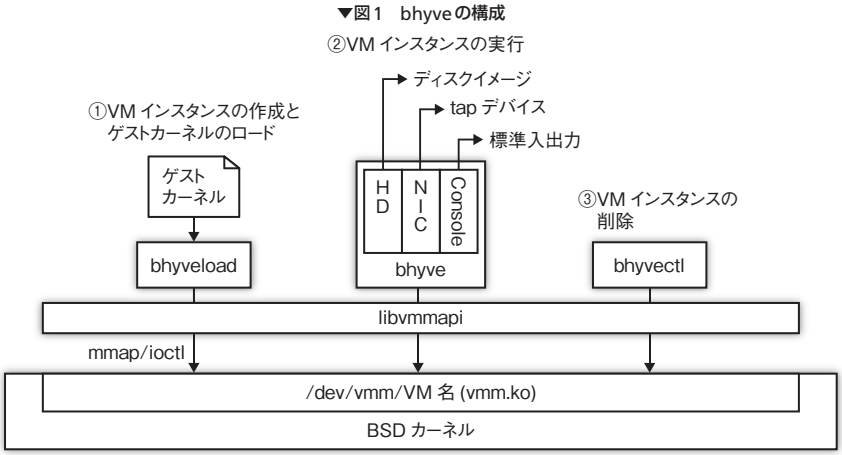
前述のとおり、bhyveはBIOSやUEFIなどのファームウェアをサポートしておらず、ディスクイメージ上のブートローダーを実行できません。このため、ゲストカーネルをロードして起動可能な状態に初期化するゲストOSローダ(/usr/sbin/bhyveload)が付属します。bhyveではディスクイメージ上のブートローダを実行できないため、ゲストカーネルをロードして起動可能な状態に初期化するゲストOSローダ(/usr/sbin/bhyveload)が付属します。

/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

▼図1　bhyveの構成



トマシンの起動は起動していません。また、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vmm.koが提供するデバイスに対してmmapやioctlを発行してゲストマシンの初期化や実行を行います。全体図を**図1**に示します。

grub2-bhyve

前述の/usr/sbin/bhyveloadでは、FreeBSD以外のゲストOSを起動させられません。これを解決するために、GRUB2をFreeBSD上で実行可能なプログラムに改変したgrub2-bhyveが提供されています。grub2-bhyveは/usr/sbin/bhyveloadと異なり汎用OSローダであるため、GRUB2がサポートする各種OSを起動させることができます。

これまでに動作が確認されているOSは次の3つです。

・Linux
・OpenBSD
・FreeBSD（grub2-bhyve経由）

/usr/sbin/bhyveや/usr/sbin/bhyveloadはFreeBSD baseに付属するプログラムであるのに対し、grub2-bhyveはPorts・pkgng経由で提供されるパッケージとなっています。

bhyve-script

QEMU/KVMのコマンド引数はユーザにとってわかりづらく、KVMベースの仮想マシンの構築にはlibvirt + virshなどのフロントエンドが多く用いられています。

libvirtは今のところbhyveをサポートしていませんが、その代わりになるような簡易的なシェルスクリプトがbhyve.orgから提供されています。ダウンロードページへのURLはこちらです（http://bhyve.org/tools/）。

vmrc

vmrcはbhyve-scriptをさらに拡張したもので、bhyveのほかにjailやQEMUもサポートします。

今回はbhyve-scriptを用いたVMの構築方法について紹介しますが、今後はこちらが主流のツールになる可能性がありそうです。

各種ゲストOSのインストール方法

ここからは、FreeBSD 10.0-RELEASEにおける各種ゲストOSのインストール方法を紹介します。なお、すべてのゲストOSはamd64（x86_64）版であり、32bit版はサポートされていません。

事前に必要な作業

grub2-bhyveとbhyve-scriptをインストールします。

```
# pkg install grub2-bhyve tmux
# fetch http://bhyve.org/bhyve-script.tar
# tar -xvf bhyve-script.tar
```

▶ Ubuntu 13.10の場合

```
# cd bhyve-script
# cp vm0 ubuntu0          ← これから作るVMの名前にvm0をコピー（末尾は数値で、他のVMと重複しない値でなければならない）

# vi ubuntu0
NIC= "em0"                ← 使用中のNIC名に変更
VCPUS="1"                 ← 任意のvCPU数に変更
VMRAM="1024"              ← 任意のメモリサイズに変更
VMOS="freebsd"            ← "freebsd"を"linux"に変更
VMOSVER="9.2-RELEASE"     ← "9.2-RELEASE"を"ubuntu13.10"に変更
DEVSIZE="2G"              ← 任意のディスクサイズに変更

# sh ubuntu0 iso          ← ISOファイルがダウンロードされ、ブートローダが起動する。ネットワークの設定画面でDHCPが失敗するため、「Do not configure the network atthis time（ネットワークの設定をしない）」を選択。パーティション作成画面では「Guided - use entire disk（LVMを使わない）」を選択

# sh ubuntu0 start        ← HDイメージからUbuntuが起動する。ただし、このままでは端末を閉じるとVMが強制終了してしまう。これを避けるにはtmuxを使ってバックグラウンドで走らせる必要がある。tmuxを使うには次のようにスクリプトを編集すればよい

# vi ubuntu0
CONSOLE="default"         ← "default"を"tmux"または"tmux-detached"に（tmux-detachedは実行時にtmuxをdetachする設定）変更する
```

▶ Debian 7.3の場合

VMOSVERを"debian7.3.0"にすれば、Ubuntu 13.10とほぼ同様の手順でインストール可能です。

▶ CentOS 6.5の場合

```
# cd bhyve-script
# cp vm0 centos1          ← これから作るVMの名前にvm0をコピー（末尾は数値で、他のVMと重複しない値でなければならない）

# vi centos1
NIC="em0"                 ← 使用中のNIC名に変更
VCPUS="1"                 ← 任意のvCPU数に変更
VMRAM="1024"              ← 任意のメモリサイズに変更
VMOS="freebsd"            ← “freebsd”を“linux”に変更

VMOSVER="9.2-RELEASE"     ← "9.2-RELEASE"を"centos6.5"に変更
DEVSIZE="2G"              ← 任意のディスクサイズに変更

# sh centos1 iso          ← ISOファイルがダウンロードされ、ブートローダが起動する。"Error processing drive: pci-0000:00:02.0-virtio-pci-virtio0"というようなエラーが表示されたら「Re-initialize all」を選択する。パーティション設定の画面では「Use entire disk」を選択する

# sh centos1 start        ← HDイメージからCentOSが起動する
```

▶ FreeBSD 9.2-RELEASEの場合

```
# cd bhyve-script
# cp vm0 freebsd2         ← これから作るVMの名前にvm0をコピー（末尾は数値で、他のVMと重複しない値でなければならない）

# vi freebsd2
NIC="em0"                 ← 使用中のNIC名に変更
VCPUS="1"                 ← 任意のvCPU数に変更
VMRAM="1024"              ← 任意のメモリサイズに変更
DEVSIZE="2G"              ← 任意のディスクサイズに変更

# sh freebsd2 iso         ← ISOファイルがダウンロードされ、ブートローダが起動する
# sh freebsd2 start       ← HDイメージからFreeBSDが起動する
```

▶ OpenBSD 5.4の場合

OpenBSDサポートはまだ実験段階で、カーネルを改変したバージョンの5.4しか動きません。そのため、次のような手順で改変済みディスクイメージを取得・インストールする必要があります。

```
# cd bhyve-script
# cp vm0 openbsd3         ← これから作るVMの名前にvm0をコピー（末尾は数値で、他のVMと重複しない値でなければならない）

# vi openbsd3
NIC="em0"                 ← 使用中のNIC名に変更
VCPUS="1"                 ← 任意のvCPU数に変更
VMOS="freebsd"            ← "freebsd"を"openbsd"に変更
VMRAM="1024"              ← 任意のメモリサイズに変更
```

```
# mkdir -p ./vm/obsd1
# bunzip2 flashimg.amd64-20131014.bz2
# cp flashimg.amd64-20131014 ./vm/openbsd3/openbsd3.img
# sh openbsd3 start       ← rootのパスワードは'test123'でログイン可能
```

▶ インストールのエラー回避方法

Linuxゲストで「WRMSR・RDMSR」などのエラーメッセージと共にbhyveが異常終了する場合、一部のLinuxカーネルと一部のIntel CPUの組み合わせでは「Unknown WRMSR code 391, val 2000000f, cpu 0」「vm exit rdmsr 0xe8, cpu 0」などのエラーがでる場合があります。これはbhyveの実装上の問題で10.0-RELEASEでは修正されていません、次の手順でCURRENT上のパッチを取得・適用することで回避できます。

```
# svn co svn://svn.freebsd.org/base/head
# cd head
# svn diff -r259634:r259635 > ~/msr.diff
# cd /usr/src
# patch -p0 < ~/msr.diff
# cd usr.sbin/bhyve
# make
# make install
# cd ~/bhyve-script
# vi centos1
BHYVECMD="/usr/sbin/bhyve \
-c "$VCPUS" \
```

この行を次のように書き換える

```
BHYVECMD="/usr/sbin/bhyve \
-w \
-c "$VCPUS" \
```

まとめ

今回はFreeBSD 10.0-RELEASEで実際にいくつものゲストOSを実行する方法を解説しました。いよいよbhyveが実用的に使えるようになってきたのを実感してもらえたと思います。

次回は、今号で紹介する予定でした仮想マシンのネットワークデバイスについて解説します。SD

Be familiar with FreeBSD.

チャーリー・ルートからの手紙

第6回 ◇ pkg(8)＆portsハイブリッド運用！

◉著者プロフィール

後藤 大地（ごとう だいち）

BSDコンサルティング（株） 取締役／（有）オングス 代表取締役／FreeBSD committer

エンタープライズシステムの設計、開発、運用、保守からIT系ニュースの執筆、IT系雑誌や書籍における執筆まで幅広く手がける。カーネルのカスタマイズから業務に特化したディストリビューションの構築、自動デプロイ、ネットワークの構築など、FreeBSD/Linuxを含め対応。BSDコンサルティングでは企業レベルで求められるFreeBSDの要求に柔軟に対応。

pkg(8)の凄さはPorts Collectionとの地続き感にあり?!

2年ぶりのメジャーアップグレードバージョンということもあって、さっそくFreeBSD 10.0-RELEASEをインストールして最新技術を楽しんでいる方が多いようです。戸惑う変更もあると思いますけれども、何にせよpkg(8)の強力さに舌鼓を打たれているのではないかと思います。簡単なインストール、手軽に最新版へあがり続けるアップデート作業。これまでのシステムセットアップ作業に比べると格段に時間が短くなりました。管理を次のレベルへ引き上げるいろんなアイディアが浮かんできますね :)

かといってすべてpkg(8)で済むかといえば、実はそういうわけでもありません。デスクトップ（ワークステーション）用途だとデフォルトのパッケージだけで事足りることもありますけれども、サービスを提供しているサーバとなると、次の2点を解決できるのはpkg(8)ではなくPorts Collectionです。

- オプションを指定してビルドする必要があるソフトウェア
- セキュリティ脆弱性対策ですぐにでもアップデートが必要なソフトウェア

オプションを指定してビルドする必要があるソフトウェアはpkg(8)経由ではインストールできませんし、次のパッケージセットが公開されないかぎり、セキュリティ脆弱性を抱えたソフトウェアをアップグレードすることはできません。ここで役立つのがPorts Collectionです。

しかし、pkg(8)が本当の意味で驚異的なのはここからです。pkg(8)とPorts Collectionは完全に地続きになっていて世界の分断がありません。たいていのソフトウェアはpkg(8)でインストールして、一部のソフトウェアはPorts Collectionからビルドしてインストール、またはセキュリティ脆弱性を抱えたソフトウェアはそのときだけ一時的にPorts Collection経由でアップグレードする、といったことをシームレスに実施できます。

Ports Collectionは常に最新版であり続けます。pkg(8)は週一でそのとき最新のPorts Collectionからまとめてビルドされる一連のパッケージの集まりです。こういったしくみがあることで、pkg(8)によるインストールもPorts Collectionからのインストールも、バージョン間の差異などの問題のリスクを低く抑えています。このしくみを利用するためだけにFreeBSD 10.0-RELEASE以降にアップグレードしたいと考える管理者は少なくないでしょう。以降で具体的な操作方法を説明します。

Ports Collectionと一緒に運用する

pkg(8)は便利ですが、前述したとおりPorts Collectionからビルドしてインストールする必要があるソフトウェアもあります。たとえばHTTPSを有効にしたHTTPサーバが必要な場合、Ports Collectionからオプションを指定してビルドし、インストールする必要があります。

各コマンドのうしろについている括弧書きの数字は、manコマンドで見ることができるマニュアルに記載されている章番号を表しています。

このような場合にはpkg(8)のロック機能を使います。pkg(8)ではパッケージセット全体で1つのまとまりととらえるので、アップグレード作業というのは基本的に全パッケージが対象です。個別に指定してアップグレードという作業はしません。代わりに、特定のソフトウェアはアップグレードの対象から外す、という指定ができます。これを活用してPorts Collectionからインストールするソフトウェアとの連携を実現します。

まず、ビルドオプションを/etc/make.confファイルなどに書いておきます。**リスト1**では、NginxをHTTPS機能付きでビルドするオプションを指定しています。

NginxをPorts Collection経由でインストールしたら、**図1**のようにpkg(8)コマンドでNginxをロックします。これでNginxは、**pkg upgrade**でアップグレードされる対象から外れます。**info**サブコマンドに**-k**を指定するとロックされているかどうかを確認できます(**図2**)。ロックを解除するには**unlock**サブコマンドを使います(**図3**)。

pkg(8)でアップグレードを実施したタイミングで、ロックしているサードパーティ製ソフトウェアはそれぞれ個別にPorts Collectionから再インストール(portsnap fetch update && make reinstall clean)します。これでpkg(8)で管理しているソフトウェアも、Ports Collectionからビルドしてインストールしているソフトウェアも、ともに最新版を維持することができます。

セキュリティ脆弱性への対処

pkg(8)には、登録されているサードパーティ製ソフトウェアのセキュリティ脆弱性をチェックする機能があります。**pkg audit -F**のように実行することで調査できます。たとえば**図4**のようにセキュリティ脆弱性が報告されているインストール済みのソフトウェアが表示されます。この情報はroot宛てに送られてくるセキュリティ日報メールでも確認できます。

すでにアップデート版が提供されているようであ

▼リスト1　Ports CollectionからのNginxインストール設定例(/etc/make.conf)

```
.if ${.CURDIR:M/usr/ports*} != ""

### nginx
. if ${.CURDIR} == "/usr/ports/www/nginx"
OPTIONS_SET+=           HTTP_SSL
. endif
.endif
```

▼図1　Nginxパッケージをロック

```
# pkg lock nginx
nginx-1.4.4_2,1: lock this package? [y/N]: y   ←yと入力
Locking nginx-1.4.4_2,1
#
```

▼図2　ロックされていることを確認

```
# pkg info -k dash nginx
dash-0.5.7                      no
nginx-1.4.4_2,1                 yes
#
```

▼図3　Nginxパッケージのアンロック

```
# pkg unlock nginx
nginx-1.4.4_2,1: unlock this package? [y/N]: y   ←yと入力
Unlocking nginx-1.4.4_2,1
#
```

ればpkg upgradeでアップグレードします。図5のようにパッケージが提供されていない場合には、Ports Collectionから最新版をインストールします。パッケージよりもPorts Collectionのほうが常に最新版が提供されています。portsnap(8)でPorts Collectionを最新版へアップデートし(図6)、make reinstall cleanで再インストールします(図7)。

セキュリティ脆弱性を抱えたソフトウェアがなくなると、pkg auditで問題のあるソフトウェアは存在しないという出力が出るようになります(図8)。auditに指定する-Fはセキュリティ脆弱性のデータファイルをチェック前にダウンロードしてくるという指定です。

将来のバージョンで挙動が変わる可能性はありますが、現在の実装ではpkg(8)でロックしたパッケージをPorts Collection経由で再インストールした場合、設定したロックは解除されるしくみになっています。このため、セキュリティ脆弱性対策やバージョンアップのためにmake reinstallした後で、pkg upgradeによる自動アップグレードの対象になってしまうことがあります。手動でアップデートしたソフトウェアでロックする必要があるものは確実にpkg lockでロックするようにしてください。

知ってると便利! パッケージの探し方Tips

pkg(8)のsearchサブコマンドでパッケージが存在するかどうかチェックできます。まず、登録されている全パッケージの一覧を取得する方法を知っておきましょう。この方法がわかれば、あとは

▼図4　セキュリティ脆弱性を抱えたパッケージをチェック

```
# pkg audit -F
Vulnxml file up-to-date.
curl-7.33.0_1 is vulnerable:
cURL library -- cert name check ignore with GnuTLS
CVE: CVE-2013-6422
WWW: http://portaudit.FreeBSD.org/4e1f4abc-6837-11e3-9cda-3c970e169bc2.html

1 problem(s) in the installed packages found.
#
```

▼図5　アップデートが提供されていない場合のメッセージ

```
# pkg upgrade
Updating repository catalogue
Nothing to do
#
```

▼図6　portsnap(8)でPorts Collectionを最新版へアップグレード

```
# portsnap fetch update
Looking up portsnap.FreeBSD.org mirrors... 7 mirrors found.
Fetching snapshot tag from ec2-ap-northeast-1.portsnap.freebsd.org... done.
Fetching snapshot metadata... done.
Updating from Sun Dec 29 03:11:49 JST 2013 to Sun Dec 29 17:41:49 JST 2013.
Fetching 4 metadata patches... done.
 ... 略 ...
#
```

▼図7　Ports Collection経由で再インストール

```
# pkg info -o curl
curl-7.33.0_1                  ftp/curl
# cd /usr/ports/ftp/curl
# make reinstall clean
===>  License MIT accepted by the user
 ... 略 ...
#
```

▼図8　セキュリティ脆弱性対策完了

```
# pkg audit -F
Vulnxml file up-to-date.
0 problem(s) in the installed packages found.
#
```

grep(1)に流し込んで正規表現でふるいにかけるといったことが簡単にできます。図9のように、パターンに'.*'を指定すると全パッケージ一覧を表示させることができます。

たとえばSubversion 1.8系をインストールしたいとして、「svn」で検索をかけるとします。この場合、図10のように表示され、該当しそうなパッケージは見つかりません。

使おうとしているパッケージのPorts Collectionのパスを知っているなら（現時点ではこちらのほうをよく知っているというユーザの方がほとんどでしょう）、図11のように操作することで対象パッケージのパッケージ名を取得することができます。この名前をベースにsearchで検索すれば、関連するパッケージも含めてチェックできます。

Ports Collectionのときには/usr/ports/に移動してからmake searchを実施したり、whereis(1)コマンドでソフトウェアを探しましたが、今後はpkg(8)コマンドのsearchサブコマンドでパッケージを探すといった作業をすることになりそうです。

使い尽くそう、pkg(8)のすべて!

サードパーティ製ソフトウェアの導入やアップグレード作業が桁違いに簡単になりました。新しいサーバの構築やシステム開発もやり方が変わってきます。pkg(8)はJailとの連携機能も提供していますし、chroot(8)やjail(8)と組み合わせて隔離環境へサードパーティ製ソフトウェアをインストールしたり、隔離環境内のソフトウェアをアップグレードする作業も簡単に実現できます。Jailを使ってホスティングサービスを提供している場合には、強くこの新機能の恩恵を受けることができるでしょう。

Webサービスの提供方法も変わってきます。従来よりも仮想環境の構築が簡単になりますので、アプリケーションごとにJailの中で動作させるといったことが今までよりも簡単にできます。より安全な環境をより簡単に構築して提供できるようになります。SD

▼図9　検索パターンに'.*'を指定するとすべてのパッケージが一致する

```
# pkg search '.*' | head
0verkill-0.16_2
2bsd-diff-2.11
2bsd-vi-050325_1
2d-rewriter-1.4
2dhf-2005.05_5
2ping-2.0
3dc-0.8.1_3
3ddesktop-0.2.9_10
3dm-2.11.00.019_1
3dpong-0.5_6
# pkg search '.*' | wc -l
   22613
#
```

▼図10　Subversion 1.8系をインストールしたいけれど見つけられないパターン

```
# pkg search svn
bzr-svn-1.2.2_1
cvs2svn-2.4.0_4
hidesvn-1.2
kdesvn-1.6.0_1
latex-svninfo-0.7.4_4
pecl-svn-1.0.2
psvn-emacs24-20120326.212349_2
py27-hgsvn-0.2.3
pysvn-1.7.8
qsvn-0.8.1_6
rapidsvn-0.12.1
rsvndump-0.6
ssvnc-1.0.29
statsvn-0.7.0_2
svn2cl-0.14
svn2git-1.0.10
svn_load_dirs-1.8.4
svndelta-1.0.6_4
svnkit-1.3.0
svnmailer-py27-1.1.0.d.r1373_5
svntrac-2.0.1_4
svnup-1.02
websvn-2.3.3
#
```

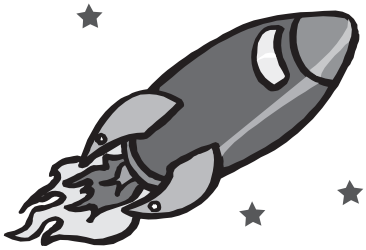
▼図11　Ports Collectionからパッケージ名を確認する方法

```
# cd /usr/ports/devel/subversion/
# make package-name
subversion-1.8.5
# pkg search subversion
git-subversion-1.8.5.2
p5-subversion-1.8.5
py27-hgsubversion-1.5.1
py27-subversion-1.8.5
ruby-subversion-1.8.5
subversion-1.6.23_2
subversion-1.7.14
subversion-1.8.5
subversion-book-4515
subversion-static-1.8.5
#
```

13

Debian Developer　やまねひでき　henrich@debian.org

パッケージの入手先の設定を見直そう



Debian Hot Topics

initシステムは「systemd」を採用?

しばらく巷を賑わせてきたDebianの次期リリースでのデフォルトinitシステム変更についてですが、技術委員会としての判断は「『Jessie』のLinuxカーネルを利用したアーキテクチャについてはsystemdを採用する」ということになりました(ですのでカーネルとしてkfreeBSDおよびHurdは今回の決定の範疇外です)。ちなみにこれで最終決定……ではなくて、このあとで開発者全員による一般投票でひっくり返されるという可能性もなきにしもあらずですので、一応ご留意ください(この投票決議はDebian開発者の誰もが参加できます。技術委員会が今回示した方向性に不満がある人がそこそこ集まれば、Jessieでのsystemd移行はペンディングになる可能性も高いでしょう)。

そして、この決定の余波として、Upstartを開発していたUbuntuのSABDFL[注1]であるMark Shuttleworth氏は、「UbuntuもSystemdへ移行しよう」と呼びかける声明を自身のブログで発表しました。筆者としてはUpstart/Mir/Unityなど、もう十分さまざまな独自実装を行っているUbuntuでそのような判断を即座にするというのは意外でしたが、これからの動向が注目されます。

詳解apt line

さて、一部の方からご要望いただきましたので[注2]、今回からDebianの使いこなし方をみなさんに伝授させていただく予定です。初回は「Debianパッケージの入手先とその区分について」の解説です。

みなさんが何かソフトウェアを使いたくてパッケージをインストールする場合、それはサーバのパッケージ置き場(リポジトリと言います)から取得しているわけですが、その設定は「/etc/apt/sources.list」ファイルに記載されています(この設定を「apt line」と呼びます)[注3]。例として筆者の検証環境(testing)のapt lineを見ると**リスト1**のようになっています。

URLが書いてあるのはわかるけれども、その規則を知らないと暗号のようにも見えますね。どんなことが書いてあるかというと、次のような内容になります。

- コメントアウトされている①の部分は、インストールに使ったメディアの指定です。ここをコメントアウトしておかないと、apt-getを実行するたびにメディアの挿入を要求されます
- ②のdeb/deb-srcはバイナリパッケージ／ソースパッケージの指定です。「パッケージ

注1) "Self-Appointed Benevolent Dictator For Life"(自ら任命した優しい終身の独裁者)の略。Ubuntuの開発に対する最終的な決定権限をShuttleworth氏が持つことを示します。

注2) 引き続き、本連載へのご要望のメールをお待ちしています。

注3) Google Talkなど一部サードパーティソフトの場合は、/etc/apt/sources.list.d/ディレクトリ以下に入手先の設定が記載されています。

を自分でビルドすることはないよ！」という方は、deb-srcで始まる行をコメントアウトするとパッケージデータの更新時間が短縮できます

- ③は入手先サーバのリポジトリURLになります
- ④はどのディストリビューション(＝sqeeuze、wheezy、jessie、sidなどのバージョン)を利用するか、という指定です。lennyやetchなど、あまりに古いバージョンの場合はサーバ側に存在しないためエラーになるのでご注意ください
- ⑤はどのリポジトリ(＝main、contrib、non-freeなど)を選択するか、という指定です。インストール時には「main」だけですが、ここには複数のリポジトリを「main contrib non-free」というように列挙できます
- ⑥のsecurity.debian.orgの指定はセキュリティアップデートの取得に必須ですので、消さないように注意しましょう

これをふまえると、筆者は「ftp.jp.debian.orgからバージョンjessieを取得。リポジトリはmainだけ」という形でパッケージを導入していることが読み取れます。たいていの場合、変更するのは⑤のリポジトリ部分に利便性の点からcontribやnon-freeを追加する、あるいは、④で利用するディストリビューションを変更する(squeeze→wheezyなど)という使い方でしょう。

手動でapt lineを記述する必要がある場合ですが、たとえば「myserver.example.jp/repo/に設置したリポジトリを参照、バージョンはsid、main、contrib、non-freeリポジトリすべてを取得する」というapt lineの書き方は**リスト2**に

▼リスト1　筆者(検証環境)のapt line

```
# deb cdrom:[Debian GNU/Linux testing _Jessie_ - Official Snapshot amd64 NETINST
Binary-1 20131217-09:47]/ jessie main                                          ①

# deb cdrom:[Debian GNU/Linux testing _Jessie_ - Official Snapshot amd64 NETINST
Binary-1 20131217-09:47]/ jessie main

deb http://ftp.jp.debian.org/debian/ jessie main
deb-src http://ftp.jp.debian.org/debian/ jessie main
deb http://security.debian.org/ jessie/updates main
deb-src http://security.debian.org/ jessie/updates main                        ⑥

# jessie-updates, previously known as 'volatile'
deb http://ftp.jp.debian.org/debian/ jessie-updates main
deb-src http://ftp.jp.debian.org/debian/ jessie-updates main

# jessie-backports, previously on backports.debian.org
deb http://ftp.jp.debian.org/debian/ jessie-backports main
deb-src http://ftp.jp.debian.org/debian/ jessie-backports main
```

▼リスト2　apt lineの書き方の例

```
deb http://myserver.example.jp/repo/ sid main contrib non-free
```

なります。参考にしてください。

パッケージの入手先について——どのサーバを選択すべきか

パッケージ入手先のサーバとして、多くの人が**リスト1**にあるようにデフォルトで選択される「ftp.jp.debian.org」を指定していることでしょう(そして、それで正解です)。ftp.jp.debian.orgは、JAIST、WIDE Projectやさくらインターネットなど日本国内の複数のミラーサーバから構成されるCDNであり、利用者のネットワーク的な位置で「おそらくここが一番近いだろう」というところから自動的に選択されるようになっています。とくに理由がない限りftp.jp.debian.orgを利用するのが鉄板です。

では、ftp.jp.debian.org以外を選ぶのが良い場合とは、どのような場合でしょうか？　1つはすぐ近くにミラーが存在している場合です。日本にはftp.jp.debian.orgとして動作するサーバ以外にもミラーサーバが複数あります[注4]。ISPや所属組織がミラーを提供しているのであれば、書き換えて明示的にそちらを利用すると良いでしょう。

また、日本国外への移動／出張などが頻繁にある方も、ftp.jp.debian.org以外を選択する理由となります。日本国内にいる場合はいいのですが、そうでないときは海外からわざわざ日本のミラーまでデータを取得しにいくので、遅くなりますからね。そのような方には選択肢が2つあります。1つは「cdn.debian.net」、もう1つは「http.debian.net」で(2つの違いは実装された時期とその手法です)、双方ともに最も近い国の公式ミラーを選択してくれます(日本にいる場合はftp.jp.debian.orgを指すことになります)。

インストールするリポジトリの違い

さて、main/contrib/non-freeというリポジトリの違いですが、「Debianでのライセンス上の区分(DFSG[注5]に適合するか否か)」によるものです。はるか古代ではデフォルトインストールでmain/contrib/non-freeのすべてが参照できるように設定されていましたが、「Debianはフリーな OS を作るのが目標なんだから、デフォルトはDSFGに適合するもの(= main)だけにするべきだろ」ということで、現在はmainリポジトリのみが設定されるようになっています(ちなみにUbuntuでも同様にmainとなっていますが、Debianでのmainとは意味合いが違っています。Ubuntuでのmainは「Canonical社がサポートをする対象」のパッケージであるということで、Debianでのライセンス上の区分とは関係ありません)。

このため、mainに存在しないパッケージを利用したい場合は、利用者が明確な意志をもってmain以外を指定する、ということが必要になるわけです[注6]。

過激な人は「なんでや、non-freeなんていらないやろ！」といってnon-freeリポジトリそのものをDebianから削除するように要求したりしていますが、現在のところそれに賛同する人は多くありません[注7]。また、upstreamからの賛同の結果、non-freeからmainリポジトリに移動したパッケージもありますので、筆者としてはユーザの利便性という意味でも将来的にmainパッケージへの収録の道筋をつける意味でも、non-freeリポジトリは存在しているべきだと考えています[注8]。

そのような例の1つとして、poppler-dataと

注4) URL http://www.debian.or.jp/using/mirror.html#mirrorlist
注5) Debian Free Software Guidelineの略

注6) 「面倒くさいなー」と思われる方が多いとは思いますが、そもそもそういう理念なり何なりがないとボランティアプロジェクトは存在自体が危うい、という点も理解ください。
注7) どこにでも「過激派」はいるものです(たいてい主張だけをして、必要とされる作業はしないのが何とも……)。ちなみにFSF(フリーソフトウェア財団)では「DebianはフリーなOSではない、なぜならnon-freeリポジトリがあるからだ」と言っていますが、これは見解の違いですね。Debian GNU/Linuxという表記を推奨していることから、GNU/FSFとDebianを混同される方もいらっしゃいますが、2つは別の団体で別の理念と見解を持っていることに留意しておいていただけるとありがたいです、はい。
注8) プロプライエタリなソフトウェアの利用についても同様で、Debianとしては配布などは行わないですが、利用者がプロプライエタリなソフトウェアを使うことを禁じていたり、忌避したりはしません。誤解している方も多いのですがDebianは「利用者がFLOSSしか使うの認めない！」というスタンスではありません。

いうPDFで日本語表示する際に必要になるパッケージがあります。以前はnon-freeなものでしたが、UpstreamであるAdobeがBSDスタイルのライセンスで配布してくれたので無事にmain入りしました。当時、non-freeリポジトリがなかったら、DebianユーザはPDFでの日本語表示が容易にはできなくなっていたということになります。理念は重要ではありますが、それと同時に実用性というのも重要なファクターです。

jessie-updates(あるいはsqueeze-updatesやwheezy-updates)は「stable-updates」と呼ばれるリポジトリで、アンチウィルスなど「常に最新のバージョンが必要になるパッケージ」が多く収録されます。ちなみに、検索するとこれに似たような名前のリポジトリとして「proposed-updates」がありますが、この2つは**表1**のようにその性格と意味合いが異なります。インターネットで検索して適当にコピー&ペーストしている方は気をつけましょう。

安定版を使いつつ、最新パッケージも入れる

stable-backports(jessie-backportsやwheezy-backportsなど)は以前は別サーバで提供されていた「backports」リポジトリです。これはtesting/unstableにある新しいバージョンのパッケージを依存関係を調整してリビルドし、stableで利用できるようにしたパッケージリポジトリになります。何がうれしいかというと、本体は更新頻度が少なくて済むstableを使いつつ、一部の意図的なパッケージだけは機能が増えた新しいバージョンが使える、という点になります[注9](ただし、すべてのパッケージが提供されてはおらず、メンテナができる範囲でbackportsパッケージを提供しています)。

現在の安定版(wheezy)ではapt lineへのリポジトリ追加を行ったうえで、明示的にインストールを指定することで導入可能です[注10]。たとえば、mikutterというTwitterクライアントですが、残念ながらwheezyではパッケージが存在していません。しかし、wheezy-backportsにはあるので導入するには**図1**のようにします(これに限らず、Twitterクライアントは Twitter側の仕様変更が多く、なかなかタイムリーに安定版に放り込むことが難しいので、backportsはそういう意味でも重宝します)。これでwheezyでもunstableのバージョンと同様、ほぼ最新のmikutterが利用できるようになります。一度お試しください。SD

注9) 一方でパッケージメンテナの視点からすると、stableでリリースされたパッケージに対して新規機能を追加するのは原則NGでリリースマネージャらを説き伏せる必要があるためハードルが高いのですが、backportsであればおおよそリビルドするだけで済んでしまうことが多いのでたいへんに楽です。

注10) メーリングリストでの議論(筆者がゴネた、とも言います)の結果、apt-setupパッケージの変更によって次のリリースであるjessie以降はインストール時にbackportsがデフォルトで記載されるようになっています。経緯に興味のある人はdebian-backports MLのアーカイブを参照してください。

▼図1　wheezy-backportsのmikutterパッケージを入れるための指定

```
apt lineにwheezy-backportsを追加して以下を実行
$ sudo apt-get update
$ sudo apt-get install mikutter
パッケージがすでにwheezyにあってwheezy-backportsの
バージョンを入れたい場合はその旨明示が必要
$ sudo apt-get install mikutter/wheezy-backports
あるいは
$ sudo apt-get install -t wheezy-backports mikutter
```

▼表1　stable-updatesとproposed-updatesの違い

名称	設定	収録対象パッケージ	注意点
stable-updates	デフォルトで設定される	ClamAVなど「最新のものでないと実用度が著しく落ちる」ものが収録される	proposed-updatesからも一部のパッケージが投入されることがある
proposed-updates	意図的に手動で追加しないといけない	次のポイントリリースに向けて「リリース候補用」のパッケージが収録される	問題を見つけてポイントリリース前にバグレポートするためのもの

技術誌に書くのはドキドキ!

篠田 奏子
SHINODA Kanako
レッドハット㈱
事業企画部

はじめに

はじめまして。レッドハット(以降、RH)で事業企画部に所属している篠田です。事業企画部は役割によりチームが2つに分かれています。1つは売上分析や目標値の設定などの営業支援業務、もう1つは受注業務です。筆者は後者の受注業務に携わっていますが、以前は営業職でした。RHでは社歴の長いほうで11年目になります。

『恵比寿通信』は今までエンジニア主体で書かれてきましたが、今回は営業から事業企画と職掌を変えながらOSSに携わってきた立場から、OSSを知らずにレッドハットへ来てしまった当初の驚きや学んだことを書きたいと思います。

うわっ……オープンソース、知らなすぎ……?

学生時代には情報処理を専攻しましたが、1998年に新卒入社した会社ではまだ各自にPCがあてがわれていませんでした。当時は必要に応じて特定の場所に固定されたPCを利用しにいく程度にしか使っておらず、その1〜2年後には各自にPCが与えられました。損益計算書やレポート作成のためにExcelの関数などは利用していましたが、自分が使っているソフトウェアがOSSなのかプロプライエタリ製品なのかということはまったく意識していませんでした。もしRHに転職しなければ、今でも多くの方がそうであるように、OSSに携わり理解することはなかったかもしれません。

「サブスクリプション」って何なの?

2003年の転職当時は「OSSは無料でしょ」という顧客もまだまだ多く、当社が販売するサポートなどを含むサブスクリプションを理解していただくのは苦労の連続でした。ソフトウェアといえばライセンス販売が大勢を占める時代なので仕方がないのですが、「どうすれば無料で入手でき、利用できるのか」「GPLにひっかからずにほかに提供する方法は」といった、今考えると若干失礼な質問も多くいただいた気がします。こちらもあまりわかってなかったんですけどね(笑)。

ソフトウェアのライセンス契約には「更新」という概念がなかったので、サブスクリプションの更新が必要なことを理解いただくのにずいぶんと時間を要した記憶があります。今ではサブスクリプションという言葉自体も広く認知され、あまり苦労することがなくなったのは喜ばしいことです。

ペンティアム? 64ビット?

最初はユーザからの質問がどうしても日本語には聞こえませんでした。何を言っているのかさっぱりわからず、その時期に質問をくださった方、ごめんなさい!

Intel社のPentiumやAMD社のOpteronといった部品単位での動作確認について問い合わせをいただきましたが、RHのハードウェア認証はサーバハードウェア構成ごとに行われます。つまり、たとえばあるサーバにIntelの

Pentiumが搭載されているからといって、ユーザが自作したPCに対する動作保証はありません。ですが、何が部品で何がサーバなのかや、この前提自体わからなかったので、問い合わせをもらうたびにエンジニアに確認したり自社ホームページを調べたりしていました。

エンジニアの方には簡単なことですが、このころはCPUの64ビット対応が始まった時期と重なり、OSやアプリケーションが32/64ビットのいずれで動作するかちんぷんかんぷんでした。CPUについてはこの時にいろいろと教えてもらいました。ずぶの素人にはItaniumとXeon、Opteronの「64ビット」の違いなんてわかるわけないですよねぇ……。

コンピュータとおしゃべり

エンジニアの方はターミナルを開いて、いろいろな作業をされていると思いますが、筆者は転職以前に自分でターミナルを開いたことがありませんでした。というより、ターミナルを開く必要がなかったので存在を知らなかったというほうが正しいです。最初にターミナルにコマンドを入力しているのを見たときはちょっと衝撃でした。まるでコンピュータとおしゃべりをしているようにみえました。killって入力するとサーバ止まる！　殺された！　自分でターミナルを開いてコマンドを入力した時は大はしゃぎでした。恥ずかしながら、コマンドで操作すると「ちょっとできる人」気分でした。

レッドハットらぶ♡

レッドハットに転職して間もないころは、OSSの考え方を理解できておらず、『伽藍（がらん）とバザール』の話も何それ？って感じでした。まだ社員数が少なく部署や役割が違うメンバーがすぐ隣や後ろの席にいたので、いろんな話を聞く機会がありました。同僚から本を薦めてもらい理解するにつれて、どんどんOSSの魅力に惹かれていきました。

当時、エンジニアの1人に「どうしてRHを選んだの？」と聞いたら「自分はとても幸せだよ。自分の好きな仕事でご飯が食べられるから」との返事でした。また「自分がこの仕事でご飯が食べられるのはOSSに貢献してきた何万人ものエンジニアのおかげだ」とも言っていました。努力を評価されて好きな仕事ができるのは理想だけど、それは自分の努力の賜物（たまもの）なのになぜ、ほかのエンジニアの話が出てくるのか、OSSへの理解が浅い筆者には不思議に思われました。

最初に彼の話を聞いたときは謙遜をしているのかと思っていたのですが、OSSを理解するとまったく当然のことだと思いました。自分の力だけでは限界があるけれど、同じ考えや志の人が集まりUNIXの代わりになるようなOSを作ってきました。タイミング良くちょうど顧客からもOSSが受け入れられてサポートも必要になり、やっとOSSがビジネスとして成り立つようになったからです。自分のご飯を食べるための仕事をしながら、OSSに貢献してきた先人がいたからこそ、今自分は好きなことを仕事にできているのだと。

RHにはそんなエンジニアがいる、というより、そういうエンジニアを惹きつける会社であり続けてほしいです。

恵比寿のオイシイところ

最後に軽めの話題を。恵比寿のおすすめランチはビストロ・フレンチの「レスパス[注1]」です。「鴨のコンフィ」を食べられるお店を探していて見つけたお店なのですが、オススメはレモンケーキです。甘さと酸味の絶妙なバランスが最高です。もちろん「鴨のコンフィ」（数量限定の要予約）も美味しいですよ。「豚のコンフィ」（予約不要）もオススメです。機会があればぜひお試しください。SD

注1）http://respace-ebisu.jimdo.com/

第48回

Ubuntu Monthly Report

ownCloudを使用する

Ubuntu Japanese Team
あわしろいくや　AWASHIRO Ikuya
Mail ikuya@fruitsbasket.info

今回は、ownCloudでお手軽ファイル同期サービスを構築する方法を紹介します。

ownCloudとは

ownCloudはDropboxやUbuntu Oneといったファイル同期サービスを提供するサーバです。最大の特徴はサーバもオープンソース（AGPL）で公開されていることです。Dropboxはサーバ・クライアントともにプロプライエタリであり、Ubuntu Oneもクライアントはオープンソースですが、サーバはプロプライエタリです。ここで紹介するのはownCloudのファイル同期機能だけですが、それ以外にもスケジュールやアドレスの同期もできますし、アドオンで機能を追加することによってほかにもさまざまなものを同期できます。

前述のとおりクライアントもオープンソースで提供されており、Ubuntu（Linux）用はもちろんWindowsやOS X用、AndroidとiOSといったモバイルOS用もあります。とはいえAndroid/iOS用のクライアントは有料ですが、自分でビルドしてどうこうする手間を考えたら購入してしまうのがいいでしょう。Android用のクライアントだと99円（2014年2月中旬現在）と、極めて安価です。ほかにも、最近のGNOMEやKDEだとクライアントをインストールする必要すらもなく対応していたりします[注1]。

なぜownCloudか

これから紹介することは、筆者が実際に行ったことではあるのですが、なぜownCloudを使用することにしたのか、DropboxやUbuntu Oneがあればそれでいいじゃないかというのは疑問に思うところではないかと思います。

まずDropboxもUbuntu Oneも無料の領域をほぼ使い切り、継続して使用するためには有償でより大きな容量を契約する必要があるという切実な現状があります。もちろんどこかと契約してもいいのですが、どうせならもっと自由に使用できるものに乗り換えるというのも選択肢のうちの1つです。この「自由に使用できるもの」というのは筆者にとって重要です。Dropboxの使用をやめることができれば、常用している環境からプロプライエタリなソフトを一掃できます。もちろん調子が悪いときに自分でメンテナンスできるのもメリットです。プロプライエタリなサービスだと調子が悪いからといってSSHでログインしてApacheをちょちょいと再起動とかできません。

UbuntuとownCloud

ownCloudはオフィシャルで公開しているパッケージ（以下オフィシャルパッケージ）とUbuntuのリポジトリにあるパッケージ（以下Ubuntuパッケージ）があ

注1）とはいえクライアントをインストールしたほうが絶対に便利ですので、その方法だけを紹介しています。

ります。2014年2月現在のownCloudの安定版は6.0.xで、どちらのリポジトリにもこのバージョンのownCloudが収録されています。では両者はまったく同じなのかというとそのようなことはないのが頭の痛いところで、Ubuntuパッケージはオフィシャルパッケージから一部の機能を削除しています。すべての機能を使用したい場合はオフィシャルパッケージを、一部機能が削られるものの運用を楽にしたいのであればUbuntuパッケージをという選択になるのですが、ここではオフィシャルパッケージをインストールする方法を紹介します。ただ、まったく同じバージョンだとUbuntuパッケージのほうがバージョンが高くなります。たとえば6.0.1だとオフィシャルパッケージのバージョンは“6.0.1-1”であり、Ubuntuパッケージだと“6.0.1+dfsg-1ubuntu1”です。ですから、インストール前に必ず、

```
$ apt-cache show owncloud
```

でバージョンを確認してください。Ubuntuパッケージ版には“dfsg”という文字列が入るので、これがないのが上にきていることを確認するといいでしょう。任意のバージョンのものをインストールしたい場合は、パッケージ名とバージョンを“=”で結んでください。上記の例だと、

```
$ sudo apt-get install owncloud=6.0.1-1
```

とすればインストールできます。

インストール

前述のとおり、今回はオフィシャルパッケージを使用する場合の解説をします。原稿執筆段階で14.04のリポジトリが公開されていないため13.10で解説しますが、リポジトリができたら13.10の部分を14.04に置き換えれば同じ方法でインストールできるはずです。その場合は前項を参考にして、必ずバージョンチェックを行ってください。

では、**図1**のコマンドを実行してください。

ownCloudではバックエンドのデータベースをPostgreSQL/MySQL/SQLiteの中から選択できます。オフィシャルパッケージの場合はSQLiteがデフォルトであり、Ubuntuパッケージ版ではMySQLがデフォルトです。データベースに強いこだわりがある場合を除けば、SQLiteが扱いやすくていいでしょう。

SSL

LAN内で使用するぶんにはこれで問題ないのですが、ownCloudの特性上インターネット経由で同期を取るようにしたいところです。そうであれば、平文のHTTPではなく暗号化されたHTTPSを使用するべきです。とはいえSSLの証明書を取得するとお金がかかってしまうので、手軽に自己署名証明書を使用することにします。第三者にサービスを提供する場合はともかく、自分で使う分にはこれでとくに問題ないでしょう。もちろん証明書を取得するのが望ましいのですが。

言うまでもなくSSLの設定はownCloudに対して行うものではなく、Webサーバ、今回だとApacheに対して行うものです。

自己署名証明書を使用するのなら設定はとても簡単です。次のコマンドを入力してください。これでおしまいです。

```
$ sudo a2ensite default-ssl
$ sudo a2enmod ssl
$ sudo service apache2 restart
```

図1　インストール

```
$ wget -q http://download.opensuse.org/repositories/isv:ownCloud:community/xUbuntu_13.10/Release.key
-O- | sudo apt-key add -
$ sudo sh -c "echo 'deb http://download.opensuse.org/repositories/isv:/ownCloud:/community/
xUbuntu_13.10/ /' >> /etc/apt/sources.list.d/owncloud.list"
$ sudo apt-get update
$ sudo apt-get install owncloud
```

ログインと設定とアドオン

何はなくともひとまずログインします。Firefoxなどのwebブラウザを起動し、

```
https://(IPアドレスないしホスト名)/owncloud/
```

にアクセスしてください。ownCloudをインストールしたのがUbuntu(Desktop)であればFirefoxがインストールされているので、

```
https://localhost/owncloud/
```

でアクセスできます。VPSであればユーザ名がIPアドレスということもあるでしょうし、はたまた仮想マシンだったらブリッジ接続にしてLANのIPアドレスを指定すればいいでしょう。

ownCloudにアクセスすると、自己署名証明書の警告のあと初回のみデータベースの作成とユーザ名とパスワードを入力する画面が表示されます(**図2**)。

自己署名証明書でSSLを有効にしている場合、管理ページを開くと「WebDAVインターフェースが動作していないと考えられるため、あなたのWebサーバはまだファイルの同期を許可するように適切な設定がされていません。installation guidesをもう一度チェックするようにお願いいたします」というメッセージが表示されます。無視してもとくに問題ありませんが、メッセージを消したい場合は/var/www/owncloud/config/config.phpに、

```
'check_for_working_webdav' => false,
```

を追記し、Apacheを再起動してください。

その管理ページには重要な項目があります。SSLを使用する場合は必ず[セキュリティ]の[常にHTTPSを使用する]にチェックを入れてください。これでついうっかりHTTPで接続してしまったということがなくなります。

インターネット経由でのアクセス

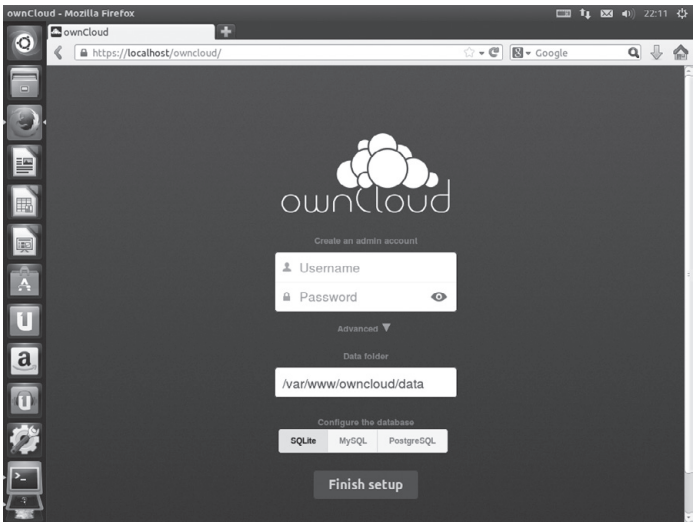
LAN内のみで接続する場合はこれでいいのですが、外部からアクセスするためには、SSLのポート443を開けておく必要があります。VPSの場合はufwでポート443を開ける必要があるでしょうし[注2]、ご自宅にある場合はルータの設定を変更してポート443をownCloudが動いているサーバに割り当てる必要があります。

同時にOpenSSH用のポートも開けておくといいでしょう。といいますかメンテナンスのためには必須です。OpenSSHはデフォルトでポート22ですが、そのまま使用すると外部からのアタックがひどいの

注2) そもそもこれをしていないと、ログイン画面にすら到達できないので大丈夫だと思いますが。

図2　初回のログイン画面

※[Advanced]をクリックすると使用するデータベースを確認できます。パスワードの確認欄がないので、誤りがないようにくれぐれも注意しましょう。



でまったく別のポートにするのが定石です[注3]。

開いているポートは少なければ少ないほどいいので、ownCloudしか動かしていないのであればこの2つ以外のポートを開ける必要はありません。

クライアント

■Ubuntu

クライアントに関してもUbuntuパッケージとオフィシャルパッケージがありますが、今回は前者を利用することにします。とはいえ14.04にしか最新版(1.5.0)はないため、これをリビルドしたものをPPAにアップロードしました。12.04用と13.10用です。**図3**のコマンドを入力してインストールしてください。

14.04にインストールする場合は3行目だけでいいですが、最新版をこのPPAに置くことがあるかもしれませんのでチェックしてみてください。初期設定はウィザードに従っていけばいいので、とくに問題となるようなところはないでしょう。

2週間ほどバリバリ使用してみた限りでは、おおむね問題なく動作してるといえます[注4]。今のところはDropboxのフォルダをそのままコピーして使用しているだけですが(フォルダの共有はしていません)、この分だとUbuntu Oneからも移行して問題なさそうです。ただ、ファイル名はDropboxよりも若干厳密のように見えます。

■Windows

Windowsクライアントはあまり使用していませんが、同じソースからビルドされるのでUbuntu(Linux)版もWindows版もほぼ同じ使い勝手です。そればかりか、ビルドもLinux(openSUSE)でクロスコンパイルを行っているので、自分でビルドするのも簡単にできそうです。

■Android

Android版のクライアントにはインスタントアップロード機能があります(**図4**)。これはスマートフォンやタブレットで撮影した写真をアップロードする機能で、DropboxやUbuntu Oneにもあってすごく便利です。手元で確認した限りでは正しくアップロードされていますが、ファイル名単位でアップロードされるので気を付けてください。すなわち、何らかの理由[注5]でファイル名がリセットされると上書きされてしまいます。これはUbuntu Oneでも同じ挙動ですが、Dropboxでは撮影時間で管理しているので、ファイル名が被っても上書きされることはありません。実際これに助けられたことがあるので、ここだけDropboxを残すのはありなのではないかと思っています[注6]。

Android 4.1を使用している場合は、"ownCloud Jelly Bean Workaround"も一緒にインストールしてください。これをインストールしないと、再起動するとアカウントが削除されてしまいます。このバージョンのAndroidをお使いでない場合は、とくにインストールする必要がありません。SD

注3) このようなことは本誌の読者であれば釈迦に説法ではないかと思います。

注4) 「なんか同期していないな」と思ったら、いつの間にかサインアウトしていたということはたまにあります。

注5) たとえばSDカードに写真を保存していて、そのSDカードが読み書きできなくなった場合とか、本体に保存していて、量が増えてきたので空き容量を稼ぐためにSDカードに移動した場合になどに起きます。もちろんどちらも実体験です。

注6) それと、ownCloudもUbuntu Oneもスクリーンショットはアップロードしませんが、Dropboxだとアップロードするのでこれも利点です。

図3 PPAからインストール

```
$ sudo add-apt-repository ppa:ikuya-fruitsbasket/owncloud
$ sudo apt-get update
$ sudo apt-get install owncloud-client
```

図4 Android版ownCloudクライアントの設定画面

※インスタントアップロード機能を有効にする場合は[自動アップロードを有効]にチェックをしてください。[WiFi経由でのみ写真をアップロード]の選択はできますが、Ubuntu OneのAndroid版クライアントにある充電時のみアップロードの選択はできません。

第25回 Linux 3.14の新機能
Btrfsとトレースのtrigger

Text：青田 直大　AOTA Naohiro

Linux 3.13が1月19日にリリースされ、3.14の開発が始まっています。今月はLinux 3.14の新機能から、Btrfsとトレースのtriggerについて見てみましょう。

Btrfsのsysfsサポート

Linux 3.14でもBtrfsはさまざまな変更が加えられていますが、一番目につく変化は/sys/fs/btrfs下でしょう。これまではioctl()を使って取得／設定していたさまざまな設定項目や情報が/sys/fs/btrfs下のファイルから読み書きできるようになりました。

では、/sys/fs/btrfsの下を見ていきましょう。/sys/fs/btrfs直下には“features”ディレクトリとmountしているBtrfsのUUIDの名前が付いたディレクトリが作成されています。

featuresディレクトリ

まずは、この直下の“features”ディレクトリとUUIDディレクトリの中の“features”ディレクトリについて見てみます(**図1**)。

Btrfsにはさまざまな機能があり、そのオン／オフをmountオプションなどで制御できます。“features”にはそういったすべてのフラグについてのファイルが、“<UUID>/features”には設定可能であるか、あるいはオンになっているフラグについてのファイルのみが入っています。sysfsから(mountしなおさずに)書き換えが可能なフラグについては、“<UUID>/features”内の該当ファイルに書き込みパーミッションがついていることに注目してください。

ここでのfeaturesは、それぞれ**表1**のような意味を持っています。これらのフラグは新しい機能が使われているファイルシステムを古いカーネルでmountしないように設定されるものです。

mixed_groupsは1つの「ブロックグループ」にデータとメタデータのブロックの混在を許すものです。Btrfsではディスクの固まった領域を切り出してデータ／メタデータ用に確保します。デフォルトではデータはデータ同士、メタデータはメタデータ同士で固めてディスク上に割り当てられますが、mixed_groupsがオンであれば1つの領域にデータもメタデータも書き込むようになります。

extended_irefは、inode参照の拡張を使うかどうかのフラグです。Btrfsではinodeの参照情報を“(inode番号 INODE_REF(定数) 親ディレクトリのinode番号)”の3つ組をキーとしたデータで管理しています。

これだけではわかりにくいので具体的に

INODE_REFについて見てみましょう。Btrfsではすべてのデータ／メタデータが3つ組をキーとした(複数の)B木で管理されています。この木のデータは“btrfs-debug-tree”コマンドを用いて実際に見てみることができます(**図2**)。

❶まず“/btrfs”に同じinodeのファイル“foo”と“bar”を作成し、そのinode番号を確認しておきます

❷btrfs-debug-tree -r を使って、fs treeのブロック番号を確認します。“-r”を指定することでBtrfsのそれぞれのツリーのルートとなるブロックが表示されます

❸“-b <ブロック番号>”を使ってツリーを探索していきます。inode番号が1412ですのでそれを頼りに探していきます

❹ブロック188759998464のitem 84のキーが“(1412 INODE_REF 256)”なので、これが探していた部分です。その下にこのキーに対応するデータとして“foo”と“bar”の名前が出ていることが確認できます

Btrfsではこのようにinodeの参照データを持っているので、1つのディレクトリの中に同じinodeのファイルをいくつも作ると、やがてブロックのサイズ制限にひっかかってしまいます。こ

▼表1　featuresの意味

big_metadata	ページサイズ(4KB)より大きなメタデータブロックを使用
compress_lzo	LZOによる圧縮を使用
default_subvol	デフォルトでmountするサブボリュームを指定している
raid56	RAID5/RAID6を使用
skinny_metadata	小さなサイズのメタデータを使用
no_holes	ファイル内のホールをエクステントとして記録しない

▼図1　featuresディレクトリ

```
$ sudo btrfs fi show /
Label: btrfs  uuid: a53121ee-679f-4241-bb44-ceb5a1a7beb7
        Total devices 3 FS bytes used 381.11GiBdevid    1
        size 1.36TiB used 397.02GiB path /dev/sdb2devid    2
        size 1.36TiB used 396.01GiB path /dev/sdc1devid    3
        size 931.32GiB used 1.01GiB path /dev/sde1

Btrfs v3.12-dirty
$ cd /sys/fs/btrfs
$ ls -l
total 0
drwxr-xr-x 5 root root 0 Feb 17 11:14 a53121ee-679f-4241-bb44-ceb5a1a7beb7
drwxr-xr-x 2 root root 0 Feb 17 11:14 features
$ ls -l features a53121ee-679f-4241-bb44-ceb5a1a7beb7/features
a53121ee-679f-4241-bb44-ceb5a1a7beb7/features:
total 0
-r--r--r-- 1 root root 4096 Feb 17 11:15 big_metadata
-r--r--r-- 1 root root 4096 Feb 17 11:15 compress_lzo
-rw-r--r-- 1 root root 4096 Feb 17 11:15 extended_iref
-r--r--r-- 1 root root 4096 Feb 17 11:15 mixed_backref
-r--r--r-- 1 root root 4096 Feb 17 11:15 skinny_metadata

features:
total 0
-r--r--r-- 1 root root 4096 Feb 17 11:15 big_metadata
-r--r--r-- 1 root root 4096 Feb 17 11:15 compress_lzo
-r--r--r-- 1 root root 4096 Feb 17 11:15 default_subvol
-r--r--r-- 1 root root 4096 Feb 17 11:15 extended_iref
-r--r--r-- 1 root root 4096 Feb 17 11:15 mixed_backref
-r--r--r-- 1 root root 4096 Feb 17 11:15 mixed_groups
-r--r--r-- 1 root root 4096 Feb 17 11:15 no_holes
-r--r--r-- 1 root root 4096 Feb 17 11:15 raid56
-r--r--r-- 1 root root 4096 Feb 17 11:15 skinny_metadata
```

のときにextended_irefがオンであれば“(inode番号 INODE_EXTREF ハッシュ)”をキーとして参照を保存するようになります。ここで“ハッシュ”は親ディレクトリのinode番号とファイル名から計算されます。

では、実際にINODE_EXTREFのデータを作ってもらいましょう(**図3**)。

▼図2　btrfs-debug-treeコマンドでB木データを表示

```
# cd /btrfs                                          ❶
# touch foo
# ln foo bar
# ls -lid foo bar .
 256 drwxr-xr-x 1 root root 52 Feb 20 12:08 .
1412 -rw-r--r-- 2 root root  0 Feb 20 12:08 bar
1412 -rw-r--r-- 2 root root  0 Feb 20 12:08 foo
# btrfs-debug-tree /dev/sdb2 -r                      ❷
root tree: 235389108224 level 1
chunk tree: 20971520 level 1
extent tree key (EXTENT_TREE ROOT_ITEM 0) 235380916224 level 2
device tree key (DEV_TREE ROOT_ITEM 0) 30867456 level 1
fs tree key (FS_TREE ROOT_ITEM 0) 188759982080 level 1
checksum tree key (CSUM_TREE ROOT_ITEM 0) 235379621888 level 2
uuid tree key (UUID_TREE ROOT_ITEM 0) 188595240960 level 0
file tree key (256 ROOT_ITEM 0) 235379720192 level 2
file tree key (259 ROOT_ITEM 0) 235379572736 level 2
file tree key (815 ROOT_ITEM 18140) 188090941440 level 2
file tree key (816 ROOT_ITEM 18146) 188768731136 level 2
file tree key (837 ROOT_ITEM 19363) 235247665152 level 2
file tree key (838 ROOT_ITEM 19369) 235345559552 level 2
file tree key (839 ROOT_ITEM 19475) 44846546944 level 2
file tree key (840 ROOT_ITEM 19481) 188632334336 level 2
data reloc tree key (DATA_RELOC_TREE ROOT_ITEM 0) 29442048 level 0
log tree key (TREE_LOG ROOT_ITEM 256) 235393662976 level 0
log tree key (TREE_LOG ROOT_ITEM 259) 235400052736 level 1
total bytes 4000334217728
bytes used 406081994752
uuid a53121ee-679f-4241-bb44-ceb5a1a7beb7
Btrfs v3.12-dirty
# btrfs-debug-tree /dev/sdb2 -b 188759982080       ❸
node 188759982080 level 1 items 68 free 425 generation 19486 owner 5
fs uuid a53121ee-679f-4241-bb44-ceb5a1a7beb7
chunk uuid 8de2549d-903d-4086-a2ea-12b1b5a2f327
        key (256 INODE_ITEM 0) block 188760064000 (11521000) gen 19486
        key (261 EXTENT_DATA 0) block 235257643008 (14358987) gen 16831
        key (264 DIR_ITEM 1317728885) block 28931768320 (1765855) gen 45
…
        key (1353 EXTENT_DATA 0) block 235270242304 (14359756) gen 16831
        key (1370 EXTENT_DATA 0) block 44989792256 (2745959) gen 19475
        key (1403 INODE_ITEM 0) block 188759998464 (11520996) gen 19486
# btrfs-debug-tree /dev/sdb2 -b 188759998464       ❹
leaf 188759998464 items 88 free space 8637 generation 19486 owner 5
fs uuid a53121ee-679f-4241-bb44-ceb5a1a7beb7
chunk uuid 8de2549d-903d-4086-a2ea-12b1b5a2f327
        item 0 key (1403 INODE_ITEM 0) itemoff 16123 itemsize 160
                inode generation 6228 transid 19481 size 84 block group 0 mode 40755 links 1
        item 1 key (1403 INODE_REF 257) itemoff 16107 itemsize 16
                inode ref index 3 namelen 6 name: rootfs
…
        item 83 key (1412 INODE_ITEM 0) itemoff 11117 itemsize 160
                inode generation 19486 transid 19486 size 0 block group 0 mode 100644 links 2
        item 84 key (1412 INODE_REF 256) itemoff 11091 itemsize 26
                inode ref index 6 namelen 3 name: foo
                inode ref index 7 namelen 3 name: bar
…
        item 87 key (FREE_SPACE UNTYPED 0) itemoff 10837 itemsize 41
                location key (FREE_INO INODE_ITEM 0)
                cache generation 163 entries 1 bitmaps 0
```

❶先ほどのファイルへのリンクを長い名前で大量に作成します
❷"(1412 INODE_EXTREF …)"のキーができていることがわかります
❸ブロック235791695872は"items 1"と"free space 86"とあるように、"(1412 INODE_REF 256)"のデータだけでいっぱいになっていることが確認できます
❹ブロック235792334848には"(1412 INODE_EXTREF ハッシュ)"のキーが保存されていることが確認できます

allocationディレクトリ

UUIDディレクトリ下に"allocation"というディレクトリもあります。このディレクトリ下からはディスクの割り当てに関する情報を取得できます(**図4**)。allocationの直下には"data"、"metadata"、"system"(設定によっては"mixed"も)のディレクトリと"global_rsv_reserved"と"global_rsv_size"の2つのファイルとがあります。dataなどのディレクトリはそれぞれデータ、メタデータ、システム用に割り当てられているブロックグループについての情報が入ったディレクトリを意味しています。

"global_rsv_*"は、ディスク容量が少なくなっていてもメタデータを更新できるように、予約されているメタデータ専用の領域についてのデータを得ることができます。"size"のほうがこの領域のサイズで最大512MBとなっています。

▼図3　INODE_EXTREFのデータを作る

```
# perl -e 'BEGIN{$x="aaa";} for($i=0;$i<256;$i++) {print "a" x 125,$x," n"; $x++}' | xargs
-n 1 ln foo  ❶
# btrfs-debug-tree /dev/sdb2 -b 235786747904       ❷
…
        key (1408 EXTENT_DATA 4194304) block 235786764288 (14391282) gen 19565
        key (1412 INODE_REF 256) block 235791695872 (14391583) gen 19565
        key (1412 INODE_EXTREF 5472337) block 235792334848 (14391622) gen 19565
        key (1412 INODE_EXTREF 2396007050) block 235792728064 (14391646) gen 19565
# btrfs-debug-tree /dev/sdb2 -b 235791695872       ❸
leaf 235791695872 items 1 free space 86 generation 19565 owner 5
fs uuid a53121ee-679f-4241-bb44-ceb5a1a7beb7
chunk uuid 8de2549d-903d-4086-a2ea-12b1b5a2f327
        item 0 key (1412 INODE_REF 256) itemoff 111 itemsize 16172
                inode ref index 6 namelen 3 name: foo
                inode ref index 7 namelen 3 name: bar
…
# btrfs-debug-tree /dev/sdb2 -b 235792334848       ❹
leaf 235792334848 items 74 free space 3629 generation 19565 owner 5
fs uuid a53121ee-679f-4241-bb44-ceb5a1a7beb7
chunk uuid 8de2549d-903d-4086-a2ea-12b1b5a2f327
        item 0 key (1412 INODE_EXTREF 5472337) itemoff 16137 itemsize 146
                inode extref index 257 parent 256 namelen 128 name: (略)
        item 1 key (1412 INODE_EXTREF 21314772) itemoff 15991 itemsize 146
                inode extref index 236 parent 256 namelen 128 name: (略)
…
```

▼図4　allocationディレクトリの情報

```
# ls -l allocation
total 0
drwxr-xr-x 4 root root    0 Feb 17 11:14 data
-r--r--r-- 1 root root 4096 Feb 17 11:14 global_rsv_reserved
-r--r--r-- 1 root root 4096 Feb 17 11:14 global_rsv_size
drwxr-xr-x 4 root root    0 Feb 17 11:14 metadata
drwxr-xr-x 4 root root    0 Feb 17 11:14 system
# grep . allocation/global_rsv_*
allocation/global_rsv_reserved:536870912
allocation/global_rsv_size:536870912
```

“reserved”のほうは実際に予約されている領域のサイズで、書きこみを行いながら見ていると、この値が減っているときがあるのに気づくことができるかと思います。

“data”や“metadata”のディレクトリの下はどれも**図5**のようなファイル構造でディスク使用量などの情報を提供しています。予約領域関連の値として、bytes_may_useはinodeなどのアロケーションに使われるかもしれない予約領域のサイズ、bytes_pinnedはトランザクション用に確保されている領域のサイズ、bytes_reservedがそれらをふくめた予約済み領域の合計サイズとなっています。

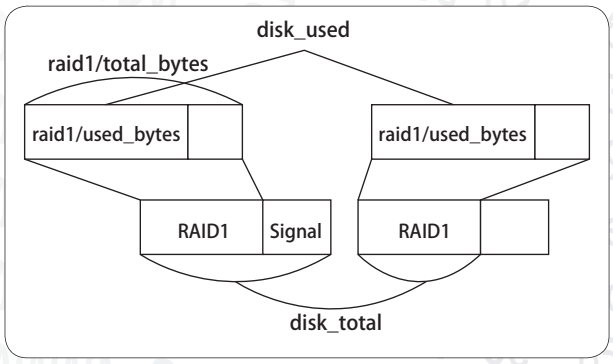
bytes_usedはファイルシステム上での使用容量を、disk_usedはディスク上での使用容量を現しています。たとえばこのシステムではBtrfsのRAID1を使用しているので、disk_usedはbytes_usedの2倍となっています。disk_totalはデータが書き込まれてはいないもののディスクからは確保されている領域となります。この値もraid1のtotal_bytesの2倍にsingleのtotal_bytesとなっていることが確認できます(**図6**)。

メタデータであれば4、システム用であれば2となっています。

▼図5　data、metadataのファイル構造

```
# grep . data/* data/*/* 2>/dev/null
data/bytes_may_use:950272
data/bytes_pinned:0
data/bytes_reserved:1171456
data/bytes_used:402235744256
data/disk_total:841821978624
data/disk_used:804471488512
data/flags:1
data/total_bytes:420915183616
data/total_bytes_pinned:2752512
data/raid1/total_bytes:420906795008
data/raid1/used_bytes:402235744256
data/single/total_bytes:8388608
data/single/used_bytes:0
```

▼図6　disk_usedとdisk_totalの概念



その他のファイル

UUIDディレクトリ下には“features”と“allocation”ディレクトリのほかにも“label”というファイルと“devices”というディレクトリがあります(**図7**)。“label”はBtrfsに設定されたファイルシステムラベルが、“devices”ディレクトリにはファイルシステムを構成するデバイスへのリンクが張られています。

trigger

次はトレースのtriggerについて紹介します。カーネル内にはさまざまなイベントが定義されており、そのイベントを有効にし、トレースを有効にすることでカーネル内でそのイベントがいつどのように起きているのかを見ることができます。このトレース機能にLinux 3.14では新しくtriggerというものが実装され、何かのイベントが起きたときにほかのイベントを有効／無効にしたり、スタックトレースを表示するといったことができるようになります。

まずはsyncシステムコールが呼ばれるとkmallocをトレースするようにして、syncが終わればkmallocのトレースを終えるようにしてみましょう(**図8**)。triggerを追加するには、イベントのディレクトリ下の“trigger”ファイルに実行したいことを指定する文字列を書き込みます。すなわち、今回　は“events/syscalls/sys_enter_sync/trigger”に“enable_event:kmem:kmalloc:1”を書き、“sys_exit_sync/trigger”に“disable_event:kmem: kmalloc:1”を書きます。ここで“:1”は回数指定になっており、1度だけenable/disableする、ということに

なります。こうしてcattrace_pipeしつつ、syncコマンドを実行すればkmallocのトレースが始まり、止まるのを見ることができます。

また、“stacktrace”とtriggerに書けばスタックトレースを得ることができます。**図9**では“btrfs/btrfs_sync_fs”にstacktraceトリガーを追加しています。

追加されたtriggerは“!”を先頭に付けたものを対応するtriggerファイルに書き込むことで削除できます(**図10**)。

まとめ

今回はBtrfsのsysfsサポートとperfのtriggerについて紹介しました。来月もひき続きLinux 3.14に入る機能を見ていきたいと思います。SD

▼図7　devicesディレクトリ

```
# ls -l devices
total 0
lrwxrwxrwx 1 root root 0 Feb 17 11:16 sdb2 -> ../../../../devices/pci0000:00/0000:00:1f.2/
ata2/host1/target1:0:0/1:0:0:0/block/sdb/sdb2
lrwxrwxrwx 1 root root 0 Feb 17 11:16 sdc1 -> ../../../../devices/pci0000:00/0000:00:1f.2/
ata2/host1/target1:0:1/1:0:1:0/block/sdc/sdc1
lrwxrwxrwx 1 root root 0 Feb 17 11:16 sde1 -> ../../../../devices/pci0000:00/0000:00:1f.5/
ata4/host3/target3:0:0/3:0:0:0/block/sde/sde1
```

▼図8　kmallocをトレースする

```
# cd /sys/kernel/debug/tracing
# echo 'enable_event:kmem:kmalloc:1' > events/syscalls/sys_enter_sync/trigger
# echo 'disable_event:kmem:kmalloc:1' > events/syscalls/sys_exit_sync/trigger
# cat trace_pipe     (syncコマンドをたたく)
…
          chrome-19990 [000] ...1 627671.249682: kmalloc: call_site=ffffffffa01b43b5
ptr=ffff8800b87e4c00 bytes_req=336 bytes_alloc=512 gfp_flags=GFP_TEMPORARY|GFP_NOWARN|GFP_
NORETRY
          chrome-19990 [000] ...1 627671.249687: kmalloc: call_site=ffffffffa01b340d
ptr=ffff880166f7f840 bytes_req=88 bytes_alloc=96 gfp_flags=GFP_TEMPORARY|GFP_ZERO
          chrome-19990 [000] ...1 627671.249693: kmalloc: call_site=ffffffffa01fe953
ptr=ffff880166f7f240 bytes_req=88 bytes_alloc=96 gfp_flags=GFP_KERNEL
```

▼図9　スタックトレース

```
# echo 'stacktrace' > events/btrfs/btrfs_sync_fs/trigger
# cat trace_pipe     (syncコマンドをたたく)
…
            sync-24037 [002] ...1 627882.363268: <stack trace>
 => btrfs_sync_fs
 => sync_fs_one_sb
 => iterate_supers
 => sys_sync
 => tracesys
            sync-24037 [002] ...1 627882.363326: <stack trace>
 => btrfs_sync_fs
 => sync_fs_one_sb
 => iterate_supers
 => sys_sync
 => tracesys
```

▼図10　トリガーの削除

```
# cat events/syscalls/sys_enter_sync/trigger
enable_event:kmem:kmalloc:count=0
# echo '!enable_event:kmem:kmalloc' > events/syscalls/sys_enter_sync/trigger
# cat events/syscalls/sys_enter_sync/trigger
# Available triggers:
# traceon traceoff snapshot stacktrace enable_event disable_event
```

April 2014

NO.30

Monthly News from jus

Japan UNIX Society

日本UNIXユーザ会　http://www.jus.or.jp/
法林 浩之　HOURIN Hiroyuki　hourin@suplex.gr.jp
波田野 裕一　HATANO Hirokazu　tcsh@tcsh.csh.sh
高野 光弘　TAKANO Mitsuhiro　takano32@jus.or.jp

荒ぶるインターネットを乗りこなす

今回は、2013年11月に行われたInternet Week 2013について報告します。

Internet Week 2013

■Internet Week 2013
～荒ぶるインターネットを乗りこなす～
【日程】2013年11月26日（火）～29日（金）
【会場】富士ソフトアキバプラザ

このイベントはJPNICが主催するもので、今回で17回目の開催です。プログラムはカンファレンス14本、チュートリアル11本、ハンズオン3本、ランチセミナー4本、同時開催イベント3本、BoF6本、懇親会の合計42本にもおよび、4日間で約2,650名が参加しました。jusは後援団体としてプログラムの企画や告知などの協力を行いました。ここではプログラムの中から何本かを選んで報告します。

■エンジニアも知っておくべき管理会計

Internet Week 2012で「エンジニアも知らなきゃならない財務会計」が好評だったことから、前年に続き「エンジニアも知っておくべき財務会計」のチュートリアルが行われ、さらに同会場の次のコマでは「エンジニアも知っておくべき管理会計」のチュートリアルが行われました。

まず管理会計の全体像について解説が行われました。管理会計の目的は「企業価値を向上させるための戦略を遂行するうえでの意思決定を支援する」ことで、「①原因を繰り返し分解していくこと」「②PDCAで計画、実行、評価を繰り返すこと」が特徴である、とのことでした。そして、「管理会計」の学問領域は範囲が広く、経営全般、企業会計、マーケティング、投資評価、さらにマネジメントシステムにまでおよぶといいます。その中で今回は、企業の生産性、損益分岐点分析、プロダクトポートフォリオマネジメント、責任会計、内部振替、予算管理、投資、原価計算、活動基準原価計算（ABC）についてそれぞれの概要に関する解説がありました。

続いて、「ITシェアードサービスにおける管理会計の課題と解決アプローチ」について解説されました。シェアードサービスとは、グループ各社、各事業部門に分散している間接業務や関連設備／情報システムを集約してグループ内に提供するサービスのこと。社内情報システム部門やITインフラ部門などは比較的この形態で行われているケースが多いといいます。

これらの形態においては、実際にかかるコストを提供サービスにどう適切に配賦するか、その貢献度をどう評価するかが課題となっており、その解決のアプローチとして、コスト配賦については活動基準原価計算、貢献度評価についてはバランススコアシート（BSC）、それぞれに解説が行われました。

管理会計の特質上論点も多様で非常に濃密なセッションでした。参加者も熱心で、現場での日頃の苦労に対する解決手法としての管理会計に対する期待を強く感じるセッションとなりました。

■SDN関連プログラム

Internet Week 2013ではSDNを主題として開かれたセッションが数多く行われました。

- **SDN時代を生き抜く為のグラフ理論とネットワークのアルゴリズム入門**
  SDNそのものというより、SDNをうまく扱うために将来必要となりそうな概念について深く触れるというタイプのセッションでした。具体的にはSDNを用いたネットワーク設計で必要となる基礎的なグラフ理論やネットワークのアルゴリズムについての話でした。アルゴリズムによって必要な計算量や記憶量などの特徴が大きく異なる事例を交え、それらを実感的に捕らえられるように宇宙を用いた例えが参考になりました。ほかにアルゴリズムの一部を変更するのみで大きく特質が変わる事例について、コードを交えた説明がなされました。参加者の理解を補助するために、グラフの塗り分けなどを可視化するアプリケーションも例示されました

- **SDN再入門～便利なSDN？　難しいSDN？　よくわからないSDN？～**
  SDNが幅広く使われだした現状から、その言葉が持つ意味を考え、概要を把握するとともに、NFV（Network Function Virtualization）についても説明がなされました。SDNの分類では、大きく「データセンタとWAN」と「物理と仮想」という２軸を切り口に、さまざまなベンダが得意としている分野について分類した説明がわかりやすかったです。また、NFVはその成り立ちがサービスプロバイダ（ETSIワーキンググループ）を起源としていること、ネットワーク機器から汎用的なサーバへの移行を目的としていることが解説されました。OpenFlowがパケットの制御に焦点をあてているのに対して、NFVは物理的な移行を目的に策定が進められていることが興味深かったです

- **SDNからNFVへ～ネットワーク仮想化パズルの完成を目指す～**
  SDNとNFVを組み合わせて実現するネットワークがどのようなものになるかという可能性について説明されました。実際にベンダが提供しようとしているネットワークの構想にはさまざまなものがありました。現段階でNFVを利用しているサービスプロバイダが、実際にNFV対応のルータを作成するデモもあり、NFVの時代が近いことを感じました

■IP Meeting 2013
～荒ぶるインターネットを乗りこなす～

毎年最終日に行われる伝統のプログラムです。午前の部は「Internet Today!」と題して、現在のインターネットの概況を伝えるセッションが組まれました。運用動向、標準化動向、インターネットガバナンスの動向といったお馴染みの内容に加えて、今回はオープンデータの現状報告がありました。オープンデータは、政府や自治体が持つデータを一般に公開し、民間が自由に利用できるようにするものです。諸外国での事例とともに日本での取り組みとしてCODE for JAPANも紹介されましたが、米国では頻繁にハッカソンなどが行われているのに比べて日本ではまだこれからの部分が多く、民間側からもっとデータを出すようにいわないと税金のムダ使いになってしまうとのことです。

後半は、今回のInternet Weekのサブタイトルに採用された「荒ぶるインターネットを乗りこなす」をテーマとするセッションを行いました。まずセキュリティ分野における最近のトピックを解説したあと、今回のInternet Weekで実施したカンファレンスの内容を、担当したプログラム委員がライトニングトーク形式で発表しました。最後のセッションはグローバル化をテーマとするパネルディスカッションでした。システムのクラウド化が進んだおかげで国際間のトラフィックが増え、また流れが一方向ではなく複雑になっていること、世界各国でインターネットについての規制が厳しくなりつつあり、それに対してインターネットのあるべき姿を維持するために政府と闘っていく必要があることなどの話がありました。そして日本人もインターネットガバナンスにもっと参加し、世界に貢献してほしいという意見が出されました。2015年にはIETFやAPRICOT-APANが日本で開催されるので、世界を身近に感じる機会になりそうです。SD

より速く、より莫大に、より高みへ！

サーバマシンの測り方

ベンチマークを極める実践テクニック

第5回　HTTPベンチマークからネットワークを測る

本誌2013年7月号ではネットワークのベンチマークとしてnetperfを用いてスループットを計測しました。ネットワークのベンチマークとなると帯域の測定をはじめに行うかもしれません。しかし、本稿では実際のシステムに近づけた形でネットワークを計測します。今回はHTTPのベンチマーク方法とそのためのチューニングについて紹介します。

Writer ㈱IDCフロンティア　ソリューションアーキテクト　藤城 拓哉（ふじしろ たくや）／Twitter@tafujish

ネットワークを計測するポイントとは

ネットワークを計測するとき、その対象はいくつか考えられるので、シンプルなネットワーク構成（**図1**）をもとに見てみると主に3つの対象があります。

(1)サーバ間通信

サーバはそれぞれスイッチに接続されており、その間の性能を計測します。このとき、サーバ自体のネットワーク性能やスイッチの性能が結果に影響してきます。

(2)ネットワーク機器

ファイアウォールやロードバランサといったネットワーク機器はボトルネックになりやすい個所なので、ネットワーク機器を経由する通信からその性能を計測することは重要です。

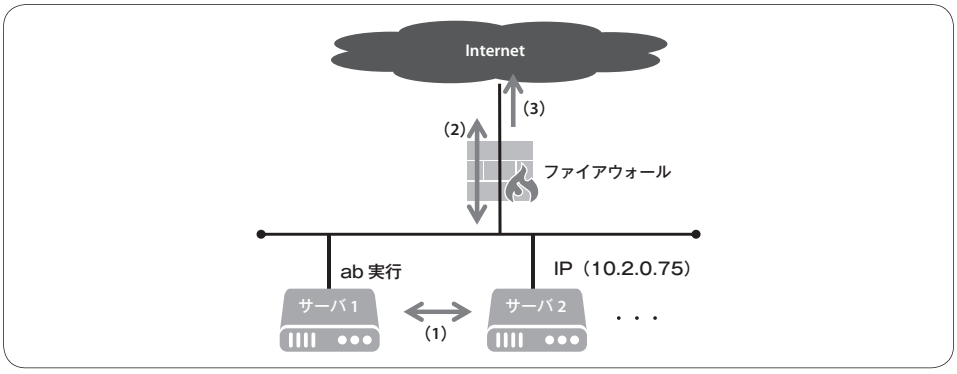
(3)上位回線

インターネットなどへの外部接続のスループットも計測ポイントの1つです。インターネット経由でのベンチマークとなるわけですが、クラウドサービスを利用すれば容易に環境を準備できるので便利です。

また、本番環境であればサービスリリース前に、(1)～(3)をシステム全体として負荷試験することもあるでしょう。一方で、ネットワークのベンチマークとしては、netperf[注1]や

注1）URL http://www.netperf.org/netperf/

▼図1　ネットワーク構成例



iperf[注2]を用いてスループットの計測をよく行います。ネットワークの帯域を計測するだけであれば、これらのツールは有効ですが、Web系のサイトで流れるような実際のトラフィックとは違います。また、ファイアウォールなどのネットワーク機器では、帯域(bps)だけでなくパケット数(pps)や新規のコネクション数(cps)の方がボトルネックになるケースもあります。

そこで今回は、Webサーバを立て、そこに対して定番ツールであるApache Bench[注3](以下、ab)を用いてHTTPのベンチマークを行うことで、ネットワークを計測する方法を紹介します。

Apache Benchを使い倒す

abはHTTPへのベンチマークツールなので、HTTPサーバが必要になります。ネットワークの構成としては、**図1**でのサーバ1をabを実行するマシン、サーバ2をHTTPサーバ(例ではIPアドレスを10.2.0.75に設定しています)とします。CentOS6.4 x86_64の環境を例に紹介していきます。

HTTPサーバを立てる

abの実行対象先としてのHTTPサーバは何でもかまいません。今回はサーバ1台でより大量にリクエストをさばくためにNginx[注4]を用いました。**図2**のとおり公式リポジトリからインストールしました。

注2) URL https://code.google.com/p/iperf/
注3) URL http://httpd.apache.org/docs/2.4/programs/ab.html
注4) URL http://nginx.org/
注5) コンフィグファイル(/etc/nginx/nginx.conf)、デフォルトのドキュメントルート(/usr/share/nginx/html/)

▼図2　Nginxインストール[注5]

```
$ sudo  yum -y localinstall http://nginx.org/packages/centos/6/noarch/RPMS/nginx-release-centos-6-0.el6.ngx.noarch.rpm
$ sudo yum -y install nginx
$ sudo service nginx start
$ sudo chkconfig nginx on
```

▼図3　HTTPコンテンツ作成

```
$ cat /dev/urandom | tr -dc '[:alnum:]' | head -c 22k > test22k.html
```

▼図4　abインストールと実行

```
$ sudo yum -y install httpd-tools
$ ab -c 10 -n 100 http://10.2.0.75/test22k.html
……省略……
Concurrency Level:      10
Time taken for tests:   0.024 seconds
Complete requests:      100 ←成功したリクエスト数
Failed requests:        0
Write errors:           0
Total transferred:      2299063 bytes
HTML transferred:       2275328 bytes
Requests per second:    4090.31 [#/sec] (mean) ←秒間リクエスト数
Time per request:       2.445 [ms] (mean) ←リクエストあたりにかかった時間
Time per request:       0.244 [ms] (mean, across all concurrent requests)
Transfer rate:          91834.86 [Kbytes/sec] received ←スループット
……省略……
```

HTTPコンテンツを設置

abの実行対象先のHTTPコンテンツを何にするか。システムの負荷試験の際には、本番のWebアプリケーションやそれに近いものを利用すると思います。今回はネットワークに負荷をかけたいので、HTTPサーバのCPUに負荷をかけないために、PHPなどで書かれた動的なアプリケーションではなく、静的なHTMLファイルで十分です。また、abは対象のファイルをダウンロードし容量とHTTPステータスコードくらいまでしか利用しないため、適当なテキストファイルで十分です(HTMLの中身は解釈していません)。

今回は**図3**のように、22KBのランダムなテキストファイルを用意しました。これをドキュメントルートに設置します。

Apache Benchの よくある使い方

abのインストールと実行は**図4**のとおりです。yumでabを導入しURLを指定して実行するだけです。

よく使うオプションとして、**-c**は同時接続数です。ここで指定した数のソケット(それぞれ異なるソースポート)から接続しますが、それぞれの接続がプロセスやスレッドで分かれるわけではありません。**-n**はリクエスト数です。ここで指定した数までHTTPアクセスを繰り返します。1つの接続に対して1つのリクエストとなります。abのリクエストは、指定したファイルを取ってくるだけで、たとえばそのHTMLファイルの中で画像のリンクがあったとしてもその画像にはアクセスしません。この**-c**と**-n**の値を大きくしていくことで負荷を大きくしていきます。

次にabの実行結果として主に見る値です。「Complete requests」や「Failed requests」のリクエストが成功した数や失敗した数です。リクエストが失敗している場合、どこかに問題があります。性能結果の目安としては、「Requests per second」の秒間リクエスト数(値が大きい方が高速)や「Time per request」リクエストあたりにかかった時間(値が小さいほうが高速)を用います。スループットは「Transfer rate」で見られます。

では、環境が整ったのでこの後は実際に実行してみてハマる点とそのチューニングをしていきます。

ネットワーク帯域

abを実行し大きく負荷をかけるとTransfer rateが125,000[KB/秒]前後で頭打ちになるかもしれません。1Gbpsのネットワークだと帯域がボトルネックになります。このような場合は、アクセスするHTMLファイルのサイズを小さくします。極端な例ですが、1文字書いて1〜2バイトのテキストファイルを作成するだけです。

実際の環境でも、ショートパケットが多い場合は帯域の前にパケットの処理数でボトルネックになることがしばしばあります。このような理由から、数キロ〜数十キロバイトのファイル

▼図5　ab実行時のエラー例(Too many open files)

```
$ ab -c 1030 -n 10000 http://10.2.0.75/test2b.html
This is ApacheBench, Version 2.3 <$Revision: 655654 $>
Copyright 1996 Adam Twiss, Zeus Technology Ltd, http://www.zeustech.net/
Licensed to The Apache Software Foundation, http://www.apache.org/

Benchmarking 10.2.0.75 (be patient)
socket: Too many open files (24)
$ ulimit -n
1024
```

へのアクセスと併せて数バイト～数十バイトのファイルの2パターンを計測するようにしています。

チューニング1「ファイルディスクリプタ」

abを実行するとき-cを図5のように大きな値にすると「Too many open files」エラーで実行できないかもしれません。各ユーザがオープンできるファイル数は制限されています。その値はulimit -nで確認できます。そして、-cで指定した数のソケットが開かれるのでファイルディスクリプタがこの制限に抵触してしまいます。

手っ取り早くこの上限を最大まで引き上げるには、/etc/security/limits.confに

```
* soft nofile 65536
* hard nofile 65536
```

と追記し、再ログインすることで反映されます。

チューニング2「マルチキューネットワーキング」

abの実行中に、その実行しているマシンやHTTPサーバのCPU使用率が、図6のようにマルチコアであっても1つのコアに偏りかつsi（ソフトウェア割り込み）の値が大きい場合があります。CentOS6のデフォルトでは、ネットワークインターフェースの処理に1つのコアしか利用できないためです。これは、図7のように受信パケット・ステアリング（RPS）、受信フロー・ステアリング（RFS）、送信パケット・ステアリング（XPS）により受信キューや送信キューを複数のコアで処理できるようになります。

RPSはIPアドレスとポートによりどのコアを使うかが決まるため、今回のように1対1の通信だと効果が薄い場合があります。

図7の例では、eth0に対してrps_cpusとxps_cpusの値をfとしています。これは1、2、3、4番のコアを使うという設定です。

▼図6　software interruptの値が大きい例

```
top - 18:10:01 up 4 days, 25 min,  3 users,  load average: 0.30, 0.11, 0.03
Tasks: 114 total,   2 running, 112 sleeping,   0 stopped,   0 zombie
Cpu0  : 16.0%us, 46.0%sy,  0.0%ni,  0.0%id,  0.0%wa,  1.0%hi, 37.0%si,  0.0%st
Cpu1  :  0.0%us,  0.0%sy,  0.0%ni,100.0%id,  0.0%wa,  0.0%hi,  0.0%si,  0.0%st
Cpu2  :  0.0%us,  0.0%sy,  0.0%ni,100.0%id,  0.0%wa,  0.0%hi,  0.0%si,  0.0%st
Cpu3  :  0.0%us,  0.0%sy,  0.0%ni,100.0%id,  0.0%wa,  0.0%hi,  0.0%si,  0.0%st
Mem:   3924448k total,   417092k used,  3507356k free,    75652k buffers
Swap:        0k total,        0k used,        0k free,   111724k cached

  PID USER      PR  NI  VIRT  RES  SHR S %CPU %MEM    TIME+  COMMAND
 4133 root      20   0  162m  78m 2196 R 99.7  2.1   0:10.88 ab
```

▼図7　マルチキューネットワーキングの設定例

```
$ sudo echo f > /sys/class/net/eth0/queues/rx-0/rps_cpus
$ sudo echo 4096 > /sys/class/net/eth0/queues/rx-0/rps_flow_cnt
$ sudo echo 32768 > /proc/sys/net/core/rps_sock_flow_entries
$ sudo echo f > /sys/class/net/eth0/queues/tx-0/xps_cpus
```

▼図8　ab実行時のエラー例（Connection reset by peer）

```
$ ab -c 1000 -n 10000 http://10.2.0.168/test22k.html
……省略……
Completed 8000 requests
Completed 9000 requests
apr_socket_recv: Connection reset by peer (104)
Total of 9851 requests completed
```

▼図9　ab実行時のエラー（Range 0..20000）

```
$ ab -c 30000 -n 200000 http://10.2.0.75/test2b.html
ab: Invalid Concurrency [Range 0..20000]
```

たとえばデュアルコアで1、2番両方のコアを使う場合は3を設定、8コアで全部使う場合はffとなります。

また、図7の設定はOS再起動で消えてしまうので/etc/rc.localなどのOS起動時に実行されるスクリプトに仕込むと良いでしょう。

HTTPサーバもチューニング

abを実行していると図8のように「Connection reset by peer」と言われ計測が途中で止まってしまうことがあるかもしれません。このエラーが出たときは、たいていHTTPサーバ側が負荷に耐えきれなかった状態です。計測中のHTTPサーバのCPU使用率を確認し、まだ余裕があるのであればHTTPサーバ側も計測用にチューニングしましょう。

今回変更したNginxのコンフィグを紹介します。

・「worker_processes」

動作させるプロセス数を増やしマルチコアのCPUを活用します。今回は4コアのマシンだったため4と設定しました。

・「worker_connections」

接続数を増やします。静的なHTMLファイルを返すだけであればメモリはガラ空きなので50000と設定しました。このとき、先述のファイルディスクリプタの設定をしておかないとサービスのリスタート時に「worker_connections exceed open file resource limit: 1024」と警告が出ます。

・「access_log」

ネットワークに負荷をかけるだけなのでアクセスログは不要です。余計なI/Oを出さないためにoffと設定しました。

コンフィグの変更個所は以上です。

このほかに、先述のマルチキューネットワーキングもとても有効なので設定しておきます。

また、大量のアクセスを行うのでSYN flood攻撃とみなされる場合があります。/var/log/messagesに「kernel: possible SYN flooding on port 80. Sending cookies.」とログが出ているようであれば、/etc/sysctl.confのnet.ipv4.tcp_syncookiesの値を0にして無効化します。

このsyncookiesの無効や先述のaccess_log無効は本番のシステムにはセキュリティ上適さないため注意してください。

Apache Benchの限界

abを実行するときの-cの値に20000を越えた値を設定すると図9のようなエラーが出ます。abで扱える同時接続数は20000まででこれ以上の負荷は作れません。

また、abのプロセスによってコア1つのCPU使用率が100%になっていることでしょう。abではマルチコアに対応していないためabを実行するCPUのコア1つまでの性能しか出せません。

次回予告

今回はabを用いてネットワークに負荷をかけてみました。abから十分に負荷をかけるにはチューニングが必要で、チューニングをすると今度はabの限界が見えてきました。

最終回の次回はabに代わるHTTPベンチマークツールを使って、ファイアウォールやロードバランサがあるネットワークへベンチマークをしていきます。SD

Software Design plus　技術評論社

GPU —CUDA・OpenGLの導入と活用 並列図形処理入門

コンピュータグラフィックスの基礎を
豊富なサンプルコードで学習（VisualStudio 対応）

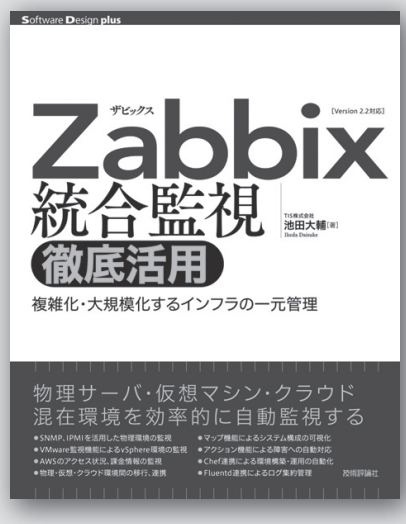
2Dや3Dなどのコンピュータ画像処理を担うGPU（Graphic Processing Unit）は性能向上が著しく、その処理能力を活かすためのソフトウェア開発が求められています。GPUはCPUと異なり並列処理機能に秀でており、複雑な図形計算を高速処理できるからです。
本書はGPUによる並列処理機能を軸に、nVIDIA社のCUDA（Compute Unified Device Architecture）の利用方法とOpenGLのプログラミング方法を基礎の基礎から解説します。

乾正知 著
B5変形判／352ページ
定価（本体3,200円+税）
ISBN 978-4-7741-6304-8

大好評発売中！

こんな方におすすめ
・GPUの並列処理計算機能
・CUDA・OpenGLに興味がある技術者

Software Design plus　技術評論社



ザビックス Zabbix [Version 2.2対応] 統合監視 徹底活用

クラウド環境を利用することがシステム構築時の一般的な選択肢の一つとなりつつありますが、用途によっては物理環境、仮想環境が適切なケースも多く、さまざまな動作環境が混在しているのが現状です。
本書はこのような物理・仮想・クラウドが混在した環境をオープンソースソフトウェア（OSS）の統合監視ツールである「Zabbix」を用いて、効率的に・一括して運用管理する手法を解説します。また、著者らによりOSSとして開発・公開中のカスタマイズ版Zabbix「HyClops」についても紹介します。

TIS㈱ 池田大輔 著
B5変形判／384ページ
定価（本体3,500円+税）
ISBN 978-4-7741-6288-1

大好評発売中！

こんな方におすすめ
・商用の監視・管理ツールからOSSに乗り換えを検討している方
・既存の運用管理者でクラウド化への対応が迫られている方

温故知新
IT むかしばなし

第32回

グラフィック アクセラレータボード

北山 貴広　KITAYAMA Takahiro　kitayamat@gmail.com　Twitter：@kitayama_t

はじめに

今回は、1990年代前半にWindows 3.0/3.1やWindows 95とともに開発され普及した、パソコン(AT互換機)用のグラフィックアクセラレータボード(ビデオボード、グラフィックボード)を思い出しながら当時のことをお話したいと思います。

筆者のパソコンの使い方は、いろいろなOSを入れて遊んだりパソコン通信やインターネットに接続して電子掲示板やメールのやりとりをしたり、エディタやオフィスソフトを使ったり、DVDやYouTubeで動画を見るなどで、高速な3Dグラフィックス性能を要求するゲームはしませんでした。

必要とするグラフィックス機能は、使用しているディスプレイの最高解像度でフルカラーで表示できて、できればストレスなく操作ができて動画がスムーズに観られれば良い程度でした。そのため、チップセット内蔵のグラフィック機能がそこそこ使えるようになってからはビデオボードはあまり買わなくなりました。

Tseng Lab ET4000/W32

筆者が一番最初に買ったビデオボードは、Tseng LabのET4000/W32で、CPUはインテル80486でVLバスでした。VLバスはVESA Localバスの略で、当時乱立していたローカルバス規格をVideo規格の標準団体のVESAが制定しました。VLバスは8ビットのISA(AT)コネクタの先に486のメモリバス信号線をそのまま載せたMCAコネクタを付け加えた豪快な仕様でした。

当時はすでにPCIバスの第一世代のPentium(P5)マシンも発売されていましたが、PCIバスがまだ出たてで拡張ボードもあまりそろっておらず、価格対性能比でもあまり魅力がなかったので、コストパフォーマンスの面からも主流だった80486マシンを買いました。

当時はマザーボード上に周辺機器用のパラレルポートもシリアルポートもオンボードで載っていなかったため、プリンタをつないだりモデムやシリアルマウスをつなぐためにマルチファンクションI/Oボードも買ってVLバスに挿しました。

20年ほどたった現在では、シリアルポートやパラレルポートはレガシーインターフェースとして廃止され、使い勝手の良いUSBポートに統一されてオンボード実装されているので、拡張ボードを買うことはなくなりました。

Number Nine 9FX Motion 531

次に筆者が購入したビデオボードは、Number Nine 9FX Motion 531の2MBで、ビデオチップのS3 Vision 868はビデオチップトップメーカーのS3社のメモリアクセスを64ビットにして動画再生支援機能を付けたチップで当時主流でした。

Motion 531は1MBと2MBのモデル、最上位のMotion 771は2MBと4MBのモデルがありました。

当時使用していたMAG MX-17Sのモニタでは、筆者が購入した2MB DRAMでは解像度1024×768(1280×1024)でフルカラーと呼ばれた16M色(24ビット)は表示不可で64K色(16ビット)表示でした(もちろん解像度を800×600に落とせ

ば16M色出せましたが……)。

16M色表示可能な4MBのMotion 771を買えばよかったのですが、Motion 531との価格の差はそこそこ大きかったこともあり、Motion 531を買いました。表示色数の違いの実感は買ってからしかわからなかったので、MB単位のメモリの違いでの表示色数の違いは身をもって理解できました。

ATI Mach 64

会社でGateway 2000(FDIVを有するPentiumプロセッサのころ)を買ったときのビデオボードがATIのMach 64でした。当時別途購入していたSCO(Santa Cruz Operation)のUnixをGateway 2000で動作させようと思い調べたところ、SCO Unixのビデオドライバが ATI Mach 32までの対応でATI Mach 64は未対応でした。そこで、当時出回っていたXFree86ではATI Mach 64も対応しているのを見つけて、自分が所有しているハードウェアが使えるドライバが存在するかどうかは非常に重要であることと、商用ソフトだからドライバがハードウェアにすぐに対応するわけでないと思いました。

このときにあらためて認識しなおしましたが、とくに当時ハードウェアを買うときはできる限り広い範囲でサポートされている定番を買うようにしていました。あまり最新のものを買うとドライバがない場合が多く買っても使えないと無駄ですので、少し前に発売されて普及しているものを買うなど気を付けていました[注1]。

Matrox Millenium /MGA

当時所属していた会社はMatroxのビデオカードの正規代理店をしていました。Matrox社のビデオカードはWindowsで非常に性能が高かったのですが、情報が公開されてなかったからか、当初XFree86ではサポートされてなくてAccelerated-Xという商用製品ではサポートされていました。

豊橋技術科学大学(当時)の真鍋敬士氏がUNIX USERの連載「プレイ・パーソナル Linux」の第7回のAccelerated-Xのときに(「第7回ビデオ・カードの性能を最大限に生かすハイ・パフォーマンス商品Xサーバ『Accelerated-X』[注2]で、「Accelerated-Xの評価ということでマトロックス製のカードでもと思ったが、これが残念ながら入手できなかった」と書いていたのを読み、社内のMatroxを扱っている部署に貸し出しを依頼して、真鍋氏にMGA Impression Plusを評価していただいた結果を第11回で掲載[注3]してもらいました。

余談ですが、筆者が単身赴任で東京に住んでいる近所のもんじゃ焼き屋さんで、現在JPCERT/CCにいらっしゃる真鍋氏と偶然お会いすることができました。真鍋氏が中心となってまとめていたLinuxの日本語化パッケージJE(Japanese Extension)が登場したのが1993年5月と、もう20年以上経つんですね。

終わりに

今のパソコンのアーキテクチャの起源となるIBM PC/XT、ATは、本体にグラフィック機能は持たず、ISAバスなどの拡張スロットにビデオカードを挿してグラフィックス機能を追加していました。

そのため、CADやグラフィックなどの高解像度を要求するものと、文字が出れば良い低解像度のものなど、多様なニーズに対応でき、多くのベンダがこの市場に参入して半導体の進歩やアーキテクチャの改善競争により、結果的により良いものが使える正のフィードバックのプラットフォームとなりました。

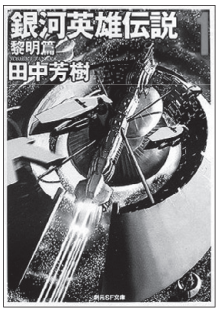
現在、チップセットやCPUに統合されたグラフィック機能は、通常使うには十分な性能です。独立した拡張ボードという形では、非常にハイエンドなグラフィックス処理のほかに、GPGPUという汎用目的でのハイパフォーマンスコンピューティング(HPC)分野への活用があります。グラフィックスという分野での発展した技術がコンピューティング(計算)に戻った流れは今後も楽しみです。SD

注1) SCSIカードならばAdaptec SCSI 1542CF、ネットワークカードならばNE2000など……思いだすのはISAバスと古いですね。

注2) http://kitano.plala.jp/~kitano/PlayPCLinux/07/index.html

注3) http://kitano.plala.jp/~kitano/PlayPCLinux/11/index.html

第4回



『銀河英雄伝説』
（田中 芳樹／東京創元社）

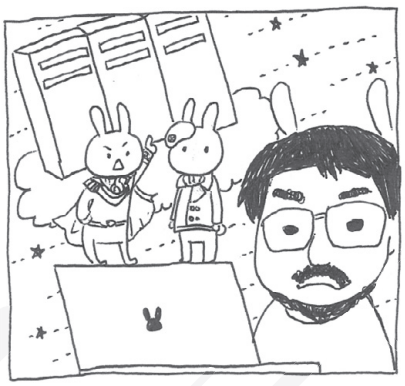
「日本で最も売れたSF」といえば、「銀英伝」こと『銀河英雄伝説』（田中芳樹）ということになるでしょうか。100刷、1,500万部。もちろん異論はありえます。1,500万部といっても本編10巻、外伝4巻（新書判）と分かれているので、上下巻で385万部の「日本沈没」（小松左京）のほうが「全巻を一作として数えれば」多いという見方もできますし、マンガを含めると『ドラえもん』（藤子・F・不二雄）が圧勝のような気もします。しかし「ドラえもん」は短編一話完結なのでむしろ比べるとしたら、合計3,000万部を超えている星新一のショートショートということになりそうですし、SF長編として認められている小説では最も売れた、いや今なお売れている、作品だと言い切ってよさそうです。

とはいえ、本作はSFとしては「ぎりぎりSF」で、「『スターウォーズ』はSFじゃない!」という人はSFには分類しないかもしれません。宇宙戦艦は出てきても宇宙人は登場しない本作はむしろ架空歴史小説で、むしろそれがまた読者を増やした要因かもしれません。宇宙はスカスカなのに「地形をいかした戦闘」や「要塞戦」が出てくるのは、SFのSを重視したい人には「ありえなく」とも、しかし我々が歴史で学ぶ地上の戦いからその様子がかえってリアルに想起できるのですから。で、本作の世界では、誰が誰と戦っているのか。

銀河帝国のラインハルト・フォン・ローエングラムは、姉を皇帝の妾として奪われたことをきっかけに、皇帝の外戚という立場すら活用して帝国簒奪をイケイケと進めていきます。自由惑星同盟のヤン・ウェンリーは、学費無料で歴史が学べるとして士官学校に入学したばかりに、イヤイヤ戦場に送られてはイヤイヤ戦果をあげ、イヤイヤ出世していきます。イケイケとイヤイヤの対決やいかに!?

それは本作で確認していただくとして、本作のテーマである、最高の帝国主義vs最低の民主主義という構図は、ITの世界ではやけにリアルです。Linus Torvaldsも寛大（Benevolent）とはいえ独裁者（Dictator）ですし、かつては自分で立てていたサーバもクラウドに集約され、寡占状態はますます進んでいます。強制の結果ではありません。その方が安全で便利だから。しかし雲（クラウド）の上に昇ってからはしごを外されたら？

本作は「伝説が終わり、歴史がはじまる」という言葉で完結していますが、我々はまだ伝説のさなかにいるのでしょうか。再アニメ化をきっかけに、再確認してみることにしましょうか……。SD



題字／イラスト　by aico

ひみつのLinux通信

作)くつなりょうすけ
@ryosuke927

第4回 ハリセンポリマーひしがた先輩

to be Continued

僕も歳を喰ったんだな、と改めて思い知らされました。同僚のコマンドエイリアス設定に愕然としましたよ。もうslコマンドを知らない若者がいるんだ、と。かつてはコマンド打ち間違えて「タイプ矯正」を潔く受けようと「sl="sl -la"」までしたのにね！　ちなみに、sex.6はDebian系ではemacs-commonパッケージに含まれています。

2月の豪雪にもめげずに徒歩出勤！
豪快出勤も辞さない、くつな先生に愛のドーピング剤を！
（エクストリーム　おたより）

Software

ネオジャパン、「desknet's NEO」の新バージョンを提供開始

㈱ネオジャパンは2月19日、グループウェア「desknet's NEO」のメジャーバージョンアップ版の提供開始を発表した。誰でも使えるやさしさとわかりやすさをコンセプトに、中小企業から大企業まで同じ機能の製品を、規模に見合った価格で提供しているのが特徴。新バージョンでは次の5つの新機能が追加された。

①グローバル対応

英語インターフェースが標準搭載される（3月下旬に正式提供予定）。設定画面から言語とタイムゾーンを変更するだけで、画面の表示を英語に切り替えられる。

②ワークフローの強化

申請処理を効率よく、ペーパレスにまわすワークフロー機能が強化された。申請書の書式をドラッグ＆ドロップで自由に作れるようになった。また、申請の階層を変更したり、申請書の内容によって承認先を分岐させたり、申請経路を柔軟に変更できるようになった。一方で、「ワークフローの自由度の高さゆえに、厳格な内部統制がとりづらい」というユーザの声にも注目した。つまり、申請先を各社員が自由に変えられるということは不正が入る余地があるとも言える。そこで同製品では、申請先を個人で入力させるのではなく、システムで自動入力させることも可能にしている。

そのほか、申請書の回付機能なども加わった。

③ソーシャル機能の充実

同製品にはすでにネオツイと呼ばれるチャット感覚で使えるSNS機能が備わっている。V2.0ではこのネオツイのダイレクトメッセージ機能で、絵文字やスタンプを使用できるようになった。

④社内向け発信ポータル

ポータルサイトの作成機能が強化された。HTMLなどの知識がなくてもイラストや写真が入ったコンテンツを作成できるコンテンツエディタを搭載。総務ポータル、社内報ポータルなどの目的別のポータルサイトも作成できる。たとえば、総務ポータルには出張申請や仮払申請などの申請画面へのリンクを作っておけば、新入社員のように同製品のワークフロー機能に慣れていないユーザを、申請画面に誘導することも可能。

⑤Webクリップと短縮URL

気になったWebページや画像を1クリックで取り込めるWebクリップ機能／画像クリップ機能が搭載された。また、desknet's NEO上の情報に直接アクセスするためのURLを短くする短縮URL機能も加わった。

導入方法／動作環境

同製品はサーバにインストールするだけで、個々のクライアント端末にはインストールは不要。PC以外にもスマートフォンやタブレットにも対応している。詳細な動作環境やライセンス体系については同製品Webサイト（http://www.desknets.com/）を参照のこと。

すでに同製品を利用しているユーザは無料でV2.0にアップグレードできる。

▲ネオツイに追加された絵文字、スタンプ機能

CONTACT ㈱ネオジャパン
URL http://www.neo.co.jp/

Software

アドバンスソフトウェア、Excelで帳票デザインできる帳票開発ツール「VB-Report 8」を発売

アドバンスソフトウェア㈱は2月27日に、Excelで帳票をデザインできる帳票生成支援コンポーネントの最新版「VB-Report 8」を発売した。

同製品は帳票ツール独自のデザイナ機能を新たに覚える必要はなく、ユーザが使い慣れたMicrosoft Excelで帳票をデザインできる帳票生成支援コンポーネント。データの差し込みや出力などのプログラミングは開発者が行い、帳票レイアウトはエンドユーザが作成するというハイブリッドな帳票開発を実現した。エンドユーザがあとから帳票レイアウトを変更することもできる。xlsx形式、xls形式の両形式に対応し、Excelで作成した帳票レイアウトに対してVB.NETやC#のプログラム上からデータを流し込んで帳票を作成する。

新機能となるセクションレポート機能では、帳票のヘッダー部、明細部、フッター部を各セクション単位で管理できるため、複雑な連続帳票も簡単に作成できる。また、付属のセルデザイナを使用することで、データを設定するセル位置の指定やデータベースの各フィールドとのバインドを直感的なUI操作で行え、よりシンプルなコードで帳票出力が可能となる。

また、新たに追加されたWeb用ビューアコントロールは、ASP.NET対応のWebコンポーネントで作成した帳票をHTMLに展開できる。そのため、デスクトップPCだけでなく、タブレット端末やスマートフォン上での出力も可能。

価格は1開発ライセンス85,000円（税別）、サーバライセンスは200,000円（税別）で販売する。

▼動作環境

OS	Windows Vista（SP2以上）/7（SP1以上）/8/8.1 Windows Server 2008（SP2以上）/2008 R2（SP1以上） Windows Server 2012/2012 R2
開発ツール	Visual Studio 2008/2010/2012/2013
対応言語	Visual Basic.NET、Visual C# （Windows/ASP.NET/WPF/Windows ストア）
フレームワーク	.NET Framework 3.5（SP1以上）
Excelバージョン	Excel 2003/2007/2010/2013
ハードディスク／メモリ	100MB以上の空き領域 /1GB以上

CONTACT アドバンスソフトウェア㈱
URL http://www.adv.co.jp/

Software

ノベル、OpenStack Havanaをベースとした「SUSE Cloud 3」を発表

ノベル㈱は2月25日、SUSEのエンタープライズ向けプライベートクラウドソリューションの最新版「SUSE Cloud 3」の提供を開始した。

SUSE Cloudは、IaaSプライベートクラウドを構築するための企業向けOpenStackディストリビューション。その最新版のSUSE Cloud 3は、OpenStackのHavanaをベースとしている。おもなアップデートは次のとおり。

- **インストール自動化、アップグレードの容易化**
  Crowbarプロジェクトをベースとしたインストールフレームワークを搭載し、クラウドインフラの管理および展開を自動化した。また、OpenStack Grizzlyベースの前バージョンからのアップグレードにも対応できる
- **マルチハイパーバイザの強化**
  KVM、Xen、Hyper-Vに加え、本バージョンよりVMware vSphereをフルサポートし、柔軟性が大きく向上した
- **オーケストレーション機能の強化**
  Havanaのオーケストレーション機能（Heat）のサポートにより、事前に設定したテンプレートに基づいて複数の仮想マシンの制御と調整を自動化できる
- **テレメトリ機能の搭載**
  インフラリソースやユーザによる利用状況を計測／収集するテレメトリ機能を搭載。収集したデータは課金システムに送ることができ、エンドユーザへのサービスを向上させるとともに、パフォーマンスを確保し、運用の管理負荷を低減する。オーケストレーション機能の自動スケーリングサービスと連動し、必要に応じてリソースを増加／減少できる

SUSE Cloud 3では、OpenStack Block Storage（Cinder）、OpenStack Networking（Neutron）、そしてこれらのプロジェクトのプラグインモデルをサポートしているほか、すべてのOpenStack APIをサポートしている。

CONTACT ノベル㈱
URL http://www.novell.com/ja-jp/

Software

レッドハット、「Red Hat Enterprise Linux 7」の強化ポイントについて説明

レッドハット㈱は2月6日、Red Hat Enterprise Linux（以下、RHEL）の次期リリースとなる「Red Hat Enterprise Linux 7」の強化ポイントについて、メディア向けに説明会を実施した。説明は米Red Hat社のプラットフォーム事業部門シニアディレクターのMark Coggin氏から行われた。

同OSの現在の最新バージョンは6で、次期バージョンの7は2013年12月にβ版として公開されている。一般の市場向けには2014年の下旬に発表するという。RHEL 7の主要な強化点として次の6つが挙げられた。

- インストール時に行われるチューニングによるパフォーマンス最適化機能に加え、「PERFORMANCE CO-PILOT」や「THERMOSTAT」といったパフォーマンスを計測／チューニングする機能が提供される
- デフォルトのファイルシステムがXFSになり、サポートされる最大容量は500TBまでとなる。RHEL 6のデフォルトファイルシステムであるext4や、Btrfsもオプションとして存在するため、選択することは可能。ただし、サポートされる最大容量は50TBまでとなる

▲米Red Hat社 プラットフォーム事業部門シニアディレクター Mark Coggin氏

- Linux Containers、cgroup、SELinux、Namespaceなどの技術によりコンテナ型仮想化を実現する。これによりアプリケーションレベルで環境を分離でき、軽量で安全な仮想環境を利用できる。同機能については、コンテナ型仮想化のOSS「Docker」を開発するDocker社とも連携し開発を進めている
- Windowsとの相互運用が強化される。SSSDという機能によりActive Directoryでの認証をもとにRHELにアクセスできるようになる
- インストーラが新しくなり、わかりやすいUIで簡単にRHELを導入できる。より迅速なデプロイメントができるよう、あらかじめ必要なパッケージが選択され事前構成されたテンプレートも利用できる。RHEL 6.xを利用しているユーザは7へインプレースアップグレード（再インストールせずに7へアップグレードすること）が可能。systemdにより優先度の高いサービスを先に開始するなど、システムの起動が早くなった
- 新しいシステム管理インターフェース「OPENLMI」を導入。統合された管理ツールと幅広いシステムリソース管理により、管理業務を効率化できる

Mark Coggin氏は「RHELは、Red Hat Storageに組み込まれているOSであり、Red Hat Enterprise Linux OpenStack Platformのコアであったりと、同社のポートフォリオの中では基礎となる重要な要素と考えている。RHEL 7のローンチにより、Red Hatは今後も継続的に市場をリードし、ユーザのイノベーションに貢献していきたい」と述べた。

CONTACT レッドハット㈱
URL http://jp.redhat.com/

Service

日本マイクロソフト、Windows Azureの日本データセンターを開設

日本マイクロソフト㈱は、クラウドビジネス強化のために、同社が展開するクラウドサービス「Windows Azure」（以下、Azure）の日本データセンターを開設した。東日本と西日本の2拠点にリージョンを設置し、2月26日から稼働開始した。

同社は、日本の企業や公共機関が必要とする、コンプライアンス対策、災害対策、レイテンシの改善などの要件に応えるべく、日本データセンターを開設した。高度なセキュリティと高い信頼性を、よりグローバルなスケールで提供するという。国内に2つリージョンを設置することにより、地理的冗長性を持たせ、かつコンプライアンス対策にも対応した災害対策を日本国内のみで実現できる。

すでに36社の企業および同社パートナー企業が、日本データセンターの早期利用プログラムによる評価を完了し、本格活用することを表明しているとのこと。さらに、120社を超えるAzureのパートナー企業も日本データセンターを利用したソリューション提供を表明しているという。

CONTACT 日本マイクロソフト㈱
URL http://www.microsoft.com/ja-jp/

Hardware

ぷらっとホーム、Zabbix社、
Zabbix 2.2搭載アプライアンス製品を提供開始

Zabbix社は、ぷらっとホーム㈱製ハードウェアを採用し、最新版Zabbix 2.2を搭載したアプライアンス「Zabbix Enterprise Appliance ZS-5220/ZP-1220」の2製品の提供を開始した。

本製品はZabbix 2.2の搭載により、VMware仮想環境のハイパーバイザや仮想マシンを自動的に監視対象として登録し監視を行う機能を新たに搭載した。煩雑な設定作業を行うことなく物理、仮想環境の統合的な監視を可能にする。

また、Web監視機能やWindowsイベントログ監視機能の改善、Windows WMI監視機能や内部イベントの追加、Webインターフェースの改良、パフォーマンスの大幅な改善など、従来製品のZS-5200、ZP-1200と比較し100以上の改善と機能追加を行っている。

すでにZS-5200、ZP-1200を利用しているユーザはZabbix Enterpriseサイト（http://enterprise.zabbix.co.jp）にてソフトウェアをアップデートしてZS-5220、ZP-1220と同等の機能を利用できる。

ZS-5220は小型ながら、約200監視対象までのシステムを監視でき、中小規模のシステムに対して統合監視ソリューションを導入できる。本体価格は298,000円、保守費用は100,000円（いずれも税別価格）。

ZP-1220はZabbixサーバの子サーバとして動作し、約200監視対象までのシステムを監視できる。収集した監視データはZabbixサーバに送付され、監視データは一元管理される（本機のみで監視を行うことはできない）。本体価格は149,000円、保守費用は50,000円（いずれも税別価格）。

▲Zabbix Enterprise Appliance ZS-5220

▲Zabbix Enterprise Appliance ZP-1220

CONTACT
Zabbix Japan LLC
URL http://www.zabbix.com/jp/
ぷらっとホーム㈱
URL http://www.plathome.co.jp

Hardware

A10ネットワークス、
DDoS対策専用アプライアンス「A10 Thunder TPS」を発表

A10ネットワークス㈱は2月13日、DDoS対策専用セキュリティアプライアンス「A10 Thunder Threat Protection System（TPS）」シリーズを発表した。

Thunder TPSは、同社のアプリケーションデリバリーコントローラ（ADC）「Thunder ADC」シリーズのDDoS防御機能をもとに、スタンドアロンの製品ラインとして開発された。Thunder ADCシリーズと同様に、同社独自のOS「ACOS（Advanced Core Operating System）」プラットフォームを搭載しており、最大155Gbpsの大容量スループットを実現し、大規模化の進むDDoS攻撃への対応が可能となった。

ラインナップとして次の6モデルが存在する。

- Thunder 4435 TPS
- Thunder 4435S TPS
- Thunder 5435 TPS
- Thunder 5435S TPS
- Thunder 6435 TPS
- Thunder 6435S TPS

各モデルのおもな違いはアプリケーションスループットで、4435/4435Sは38Gbps、5435/5435Sは77Gbps、6435/6435Sは155Gbpsとなっている。また、4435S、5435S、6435SにはSSLアクセラレーションを行う専用のハードウェアを搭載している。

同製品のすべてのモデルにFPGAによる高性能FTA（Flexible Traffic Acceleration）テクノロジが採用されており、コアシステムの汎用CPU性能を損なうことなく、30種類以上の一般的な攻撃をハードウェアで高速に検知／防御できる。

いずれのモデルも3月15日に提供を開始している。価格はオープン価格。

▲Thunder 6435S TPS

CONTACT
A10ネットワークス㈱
URL http://www.a10networks.co.jp/

Letters from Readers

「第3の○○」というと……

最近、「クラウド」「ビッグデータ」「モビリティ」「ソーシャル」から成る情報基盤のことを「第3のプラットフォーム」と呼ぶそうです。ちなみに、第1はメインフレーム、第2はクライアント／サーバシステムだそうです。「第3のプラットフォーム」なんて取ってつけたような呼び方に感じますが、それよりも「第3の○○」と目にしてビール業界を連想してしまったのは、担当だけではありますまい。

2014年2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 関数型プログラミング再入門

CPUのマルチコア化が進んだことで、並列処理を実装しやすい関数型プログラミング言語に注目が集まっています。そこで本特集では、Lisp、OCaml、Haskell、Python、Erlang、Java SE 8、Rubyを取り上げ、それぞれの関数型プログラミング手法を解説しました。

関数型言語は流行だと知っていても、いつの間にかオブジェクト指向でコードを書いてしまうので、例があるととてもうれしい。どうしてもオブジェクト指向で書いてしまう人の失敗例をたくさん載せてもらえると、自分のコードについても見つめなおせるので助かります。

東京都／hidewonさん

Lispは括弧を多用することしか知らなかったので、この記事で少し興味を持った。

東京都／binaさん

関数型プログラミング再入門、楽しく、そして頭を必死に回転させながら読ませていただきました。かねてから関数型言語に興味を持っていましたので、とてもタイムリーでした。関数脳を作る良いきっかけになったと思います。

宮崎県／maehrmさん

ときどき関数型プログラミング特集を継続してやってほしい。

愛知県／入江さん

Lispは学生のとき以来触っていないので、ちょっと触ってみようかと思います。

大阪府／栗原さん

関数型プログラミングは、従来のプログラミングとはまったく違った考え方で行う必要があります。そのため「難しい」と言われます。それにもかかわらず多くの感想が集まったのを見て、たくさんの人の関心を集めていることを、あらためて感じました。とくにLispに関するコメントが多かったのが印象的でした。

第2特集 2014年IT業界はどうなるのか?

新たな年を迎えたということで、各分野で活躍するエンジニアの方々に、2014年がどのようになりそうか、どんな技術に注目すべきかを予測してもらいました。ご自身の今年の進むべき方向が見えたでしょうか?

直接的に関係しないが、さまざまな予測を読むのは楽しかった。

岐阜県／silverfoxさん

これからのIT業界がどう変わっていくのか、どこに注目していればいいのかを知ることができました。

千葉県／有ノ木さん

インフラ系の技術がソフトウェア中心になってくると恐いと思いました。セキュリティなどのことを考えると……。

兵庫県／桑村さん

IT業界は変化が速く、また技術の分野も年々、多種多様になっています。本特集を読んで、スキルアップするためにどこから手をつけていったら良いか、その方向性がつかめたなら幸いです。

一般記事　会社組織を活性化するスパイス「コンパ」

京セラコミュニケーションシステム㈱(KCCS)は、チームワークを築くため、社員の意識を合わせるためにコンパを実施しています。その取り組みを紹介しました。

会社風土の改善などにも興味があったので、おもしろかったです。

東京都／大塚さん

飲みニケーションは一長一短。経営者によって意見が違いますね。

宮崎県／Whiskeyさん

コンパでどのような取り組みをするからチームワークがアップするのかなどの例が欲しかったです。

東京都／juliaさん

賛否両論いろいろな意見がありましたが、人間関係を深めるためのノウハウをOSI参照モデルにたとえた説明に思わず納得したという読者もいらっしゃいました。やはり技術にたとえるほうが心に響くのですね。

連載

「digital gadget」がお気に入りです。ちょっと気合を入れて読む必要がある記事が多い中、気軽に読めて好奇心を刺激されています。

神奈川県／ewiad420さん

どうしても文字ばかり、コードばかりになりがちな本誌記事の中で、写真が豊富な数少ない記事の1つです。さまざまな製品を写真で見ると、刺激されることも多いですよね。

「セキュリティ実践の基本定石」はとても参考になります。2月号の111ページに「筆者はEmacsのメールリーダーを使っているのですが（以下略）」は、まさに私のメール環境と同一で、安心できました。

東京都／psiさん

セキュリティに関する報道は、専門的過ぎて敬遠しがちになりますが、本連載では、今起こっている事件／事故について、基本的な技術を説明したうえで、その原因、対策などをわかりやすく整理しています。本連載でセキュリティに関する基礎知識を養っていただきたいと思います。

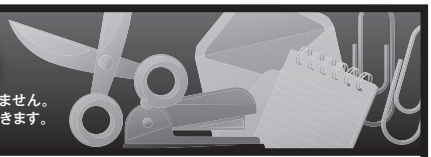
「チャーリー・ルートからの手紙」が良いと思います。普段見落としがちなコマンドの出力について、具体的な読み解き方に気づかせていただいて、たいへん参考になります。今後も期待しています。

東京都／HIさん

UNIXの内部のしくみを理解すれば、コマンドの出力結果も細かく読み解けるようになります。せっかくUNIXやコマンドを扱うからには、より深い使い方ができるようになりたいものですね。

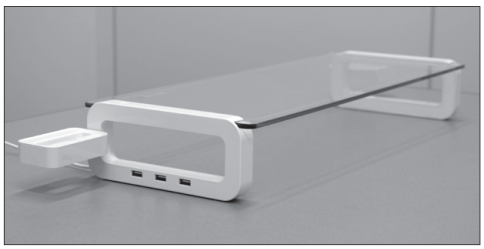
エンジニアの能率を高める一品

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。



U-BOARD SMART

5,600円（税込）／㈱テック　http://www.tecnosite.co.jp/



◀写真1
U-BOARD SMARTをディスプレイの前に置いて使う

U-BOARD SMARTはモニターボードですので、ボード上にディスプレイを置いて、その下にキーボードを置くのが普通の使い方なのでしょうが、担当は**写真1**のようにディスプレイの前に置いてみました。このほうが、ボード上にペンや書類などを置けるのでデスクが広くなります。さらに「ボードの下に手を突っ込めば、キーボードを手前に引き出さずに入力できるのでは?」と思い、さっそく試すことに。そのままでは高さが足りなかったので**写真2**のように足の下に書籍を置いてみました。これならボードの下でも快適にキーボード入力ができます。資料を見ながら入力するときも、資料はボード上に置けるので、キーボードの手前は常に広々していて快適です。いっそのことポインティングスティック付きのキーボードにすれば、マウス操作のためにボードの外に手を出す面倒もなくなります。足の側面にはUSBポートが3つあるのですが、手近なところでUSBメモリなどを抜き差しできるのもうれしいですね。

▲写真2 書籍を置いて高さを調整

2月号のプレゼント当選者は、次の皆さまです

①ノートPCタブレットデュアルアームDN-10313 ..... 京都府　阿部憲幸様
②Blue Raspberry Pi ..... 東京都　瀬長孝久様

★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
May 2014

2014年5月号
定価（本体1,220円+税）
176ページ

4月18日発売

［第1特集］ネットワーク技術超入門

「ポート」と「ソケット」がわかればTCP/IPネットワークがわかる

コンピュータネットワークにおける「ポート」と「ソケット」の役目をご存じですか？　当たり前のものとして使っているTCP/IPプロトコルですが、OSから見たときの背後のしくみと役割、そしてアプリケーションソフトウェアから見たときの通信とは何か、基礎をくまなく覚えて、ネットワーク技術の理解を深めましょう!

［第2特集］必須UNIXコマンド特訓講座

rmからcurlまで基礎技術を習得

どんなプロでも使用するときに注意が必要な「rm」コマンドをはじめとして、ネットワークエンジニアからWebプログラマまで広く使われている「curl」コマンドなど、エンジニアとして技術力を上げるコマンドの使い方を解説します。

［特別企画］「Google Glass」GDKで何が見えた?
［短期連載］Rettyのサービス拡大を支えた“たたき上げ”DevOps
［新 連 載］るびきち流「Emacs 超入門」

※特集・記事内容は、予告なく変更される場合があります。あらかじめご容赦ください。

SD Staff Room

●入稿中に風邪をひいた。ノドと鼻をやられ粘膜が真っ赤になっている（と医者はいう）。声が変だなと思っていたのだけど、日に日に悪化し最悪の時期にピーク。声が異常なときは気管支近くまで炎症が来ているので治りが遅いのだ。さらにこれから花粉症のピーク時期を迎えるので5月までヤバイなぁ。（本）

●スマホをXperia Z1に機種変した。本当はZ Ultraにしようと思ったのだが、大きさを見て思いとどまった。液晶が黄色っぽいのもパッチが出たし、カメラ性能もまずます。バッテリの持ちも良い。これにiPad mini retina with みおぼんで、やっと快適な生活が送れるようになった。（幕）

●小学生の合唱祭を聴いてきました。出場校の構成人数はさまざまでしたが、ひな壇を全部埋めるほどの大構成を擁する学校の生徒たちは、ボディーバーカッションやかけ声などもこなす完全なエンターテイナー。いったいどれくらい練習しているんだろうと感心。きれいな歌声にいやされた一日でした。（キ）

●私は毎朝、TBSラジオ「生島ヒロシのおはよう一直線」で目覚めます。6時にタイマーをかけています。6時半からの「森本毅郎・スタンバイ!」をしっかり聴きたいので、正直「おはよう一直線」は準備運動のつもりで聴き流しています。脱力感溢れる番組なので惰性で聴くのにピッタリです。（よし）

●記録的な大雪が降った日。会社は偶然休みだったがスノボツアーは決行。2時間遅れで出発したが、大渋滞で区内からも脱出できず結局中止に。翌週は残雪のため高速道路が規制されたままでまたまた渋滞。終電にも間に合わず。まさか2週続けて7時間越えのバスの旅になるとは……大雪恐るべし。（ま）

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
2014年4月号

発行日
2014年4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

●編集アシスタント
松本涼子

●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。