Special Feature 01
ビッグデータの扱い方
Special Feature 02
ベンチマークで正しく評価
2013 July
07
Software Design
[ソフトウェア デザイン]
OSとネットワーク、
IT環境を支える
エンジニアの総合誌
2013年7月18日発行
毎月1回18日発行
通巻339号
（発刊273号）
1985年7月1日
第三種郵便物認可
ISSN 0916-6297
定価
1,280円

Special Feature 01

ここからはじめる データ分析学習

データの価値を見直しませんか？

—— Excel・R・Mahout・ビッグデータ ——

Machine Learning.

Big Data

Data Analysis

ボトルネックを探れ!

Special Feature 02

自分の定規（モノサシ）を持っていますか?
「ベンチマーク」活用テクニック
[PC編/サーバ編]

Extra Feature

マニュアル通りでホントに使える?
現場から学ぶJenkins活用

新連載

すずきひろのぶ
「セキュリティ実践の基本定石」
みんなでもう一度見つめなおそう

上西康太
分散データベース未来工房
RiakとNginxだけで作る
静的ファイルホスティング

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55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。

オープンクラウド（Ⅱ）

前号では、業界全体で普及が進んでいる「オープンクラウド」について、その動向や主要なIaaS基盤ソフトウェアについて解説しました。今回はその続編として、PaaSレイヤやネットワークレイヤにおける基盤ソフトウェアや基盤技術を紹介します。

Cloud Foundry

Cloud Foundryは、VMware社が中心となって開発を進めているオープンソースのPaaS基盤ソフトウェアです。Cloud Foundry自体が独立したソフトウェアとして利用できることに加えて、VMware社からはCloud Foundryを利用したパブリッククラウドサービスとしてCloudFoundry.comが提供されています。また、Cloud Foundryの環境一式をVMware仮想マシン内にパッケージングしたMicro Cloud Foundryと呼ばれるソフトウェアも公開されています。

Cloud Foundryのアーキテクチャの特徴は、機能別に明確にコンポーネントが分けられている点です。アプリケーションの構成管理やWebサービスAPIの提供などといった核となる機能はCloud Controllerと呼ばれるコンポーネントによって行われます。アプリケーションの実行環境はDEA（Droplet Execution Agent）と呼ばれるエージェントとして提供され、外部からのリクエストはルータとロードバランサによって適切なDEAに振り分けられるしくみになっています。アプリケーションやDEAに予期しないエラーが発生した場合に、自動でインスタンスを再起動させる自己復旧機能を備えている点も大きな特徴です。

Cloud Foundryのライセンスは Apache License 2.0で、プログラミング言語はRubyやJava、Node.js、Scala、Erlangなどを、データストレージはPostgreSQL、MySQL、MongoDB、Neo4j、Redisなどをサポートしています。

OpenShift

OpenShiftは、Red Hat社が提供しているPaaS基盤ソフトウェアおよびサービスの名称です。OpenShiftの提供形態としては、オープンソースソフトウェアのOpenShift Origin、Red Hat社がPaaSサービスとして提供しているOpenShift Online、そしてオンプレミスのOpenShift環境を商用サポート付きで提供する企業向けPaaSサービスのOpenShift Enterpriseがあります。

OpenShiftのアーキテクチャは、大きく分けるとアプリケーションのデプロイや各種データの保存、認証機能などを提供するBrokerと、アプリケーションやサービスをホストするNodeの2種類のコンポーネントから構成されます。アプリケーションの実行環境一式はNode上でギアと呼ばれる単位で管理されます。ギアは自由に追加や削除が行えるため、必要に応じて柔軟に環境を拡張できる点が大きな強みと言えます。

OpenShift Originは Apache License 2.0で提供されており、プログラミング言語はJava、Ruby、Node.js、Python、PHP、Perlなどが、データストレージはPostgreSQLやMySQL、MongoDBなどがサポートされています。

SDN/OpenFlow

SDN（Software Defined Networking）は、ネットワーク構成をソフトウェアによって制御する技術の総称で、クラウド基盤を構築するうえでは欠かせないものになりつつあります。その中でもOpen Networking Foundation（ONF）によって標準化が進められているOpenFlowはとくに高い注目を集めている技術です。

OpenFlowによるネットワークは、複数のOpenFlowスイッチとそれを集中管理するOpenFlowコントローラによって構成されます。OpenFlowコントローラではパケットの経路計算や受信パケットの扱い方などが定義され、OpenFlowスイッチがその定義に基づいてパケット転送を行います。これによってスイッチ1つ1つの設定を変更することなく、コントローラの設定だけでネットワーク構成を管理することができるようになっています。

SDNに関する取り組みとしては、OpenFlowのほかにThe Linux Foundationによる OpenDaylightと呼ばれるプロジェクトもスタートしています。これはSDNにおける新しいOSSプラットフォームの構築を目指すもので、ネットワークコントローラだけでなく、プログラミングインターフェースやレイヤ4～7の機能、ネットワーク仮想化技術なども包括的にカバーしていく構想になっています。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。

Contents

Software Design Jul. 2013

1

第1特集 Special Feature 01

データの価値を見直しませんか?

ここからはじめるデータ分析学習

――Excel・R・Mahout・ビッグデータ――

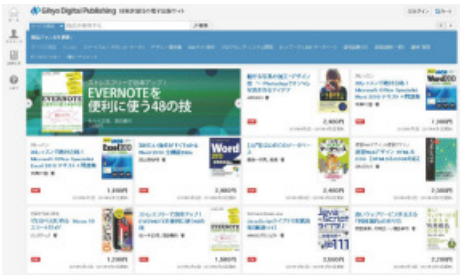
技術評論社の本が電子版で読める！

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

楽天 Rakuten KC　kobo

※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

Contents

Software Design Jul. 2013

2

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。

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。

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。

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 Jul. 2013

Column

Development

OS/Network

Inside View

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	Frank Greenaway/Getty Images
Illustration	イラスト	フクモトミホ、高野 涼香
Page Design	本文デザイン	岩井 栄子、近藤 しのぶ、SeaGrape、安達 恵美子 ［トップスタジオデザイン室］轟木 亜紀子、阿保 裕美、佐藤 みどり ［BUCH+］伊勢 歩、横山 慎昌、 森井 一三、Re:D、［マップス］石田 昌治

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

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

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

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

知識ゼロでもわかる統計学

基礎から学べる統計の本

松原望　著／A5 判／168 ページ
定価 2,079 円（1,980 円+税）
ISBN 978-4-7741-4520-4

加藤剛　著／A5 判／240 ページ
定価 2,079 円（1,980 円+税）
ISBN 978-4-7741-5065-9

加藤剛　著／A5 判／240 ページ
定価 2,079 円（1,980 円+税）
ISBN 978-4-7741-5630-9

前野昌弘　著／A5 判／160 ページ
定価 2,079 円（1,980 円+税）
ISBN 978-4-7741-4861-8

前野昌弘　著／A5 判／152 ページ
定価 2,079 円（1,980 円+税）
ISBN 978-4-7741-4962-2

前野昌弘　著／A5 判／160 ページ
定価 1,974 円（1,880 円+税）
ISBN 978-4-7741-4688-1

横内大介　著／A5 判／176 ページ
定価 2,079 円（1,980 円+税）
ISBN 978-4-7741-5315-5

実吉綾子　著／A5 判／192 ページ
定価 1,974 円（1,880 円+税）
ISBN 978-4-7741-5431-2

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

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！

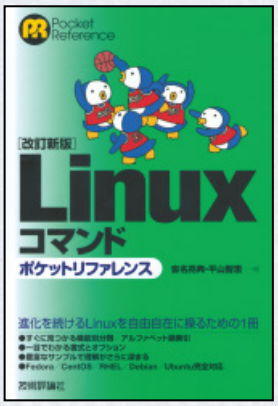
涌井良幸、涌井貞美　著
四六判／288 ページ
定価 1,764 円（1,680 円+税）
ISBN 978-4-7741-5513-5

高橋晶ほか　著
四六判／528 ページ
定価 3,024 円（2,880 円+税）
ISBN 978-4-7741-5715-3

岡本隆史、武田健太郎、相良幸範　著
四六判／272 ページ
定価 2,604 円（2,480 円＋税）
ISBN978-4-7741-5184-7

山森丈範　著
四六判／304 ページ
定価 2,499 円（2,380 円+税）
ISBN 978-4-7741-4396-5

石田つばさ　著
四六判／448 ページ
定価 2,289 円（2,180 円+税）
ISBN 978-4-7741-4836-6



沓名亮典，平山智恵　著
四六判／576 ページ
定価 2,394 円（2,280 円+税）
ISBN 978-4-7741-3816-9

牟田口大介　著
四六判／592 ページ
定価 2,919 円（2,780 円＋税）
ISBN 978-4-7741-5542-5

土井毅、髙江賢、飯島聡、髙尾哲朗 著　山田祥寛 監修
四六判／488 ページ
定価 2,709 円（2,580 円+税）
ISBN 978-4-7741-4948-6

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 3,129 円（2,980 円+税）
ISBN 978-4-7741-5600-2

藤本壱　著
四六判／336 ページ
定価 2,604 円（2,480 円+税）
ISBN 978-4-7741-5486-2

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

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3回

ラックの電源問題（その1）

イラスト：高野涼香

はじめに

今回は電源がテーマです。ブレーカーや契約電力などについて解説します。

契約電力って何ですか?

電力会社と契約している容量を超えるような使用をすると、ブレーカーが落ちてしまいます。これはアンペアブレーカーと呼び、電力会社と加入者の契約電力に基づき設置されます。契約アンペアを超える電流が流れた場合、電気の供給を遮断するものです。また、設備や建物の状況に応じて、契約できる最大容量が決まっていることが多く、大きくなればなるほど基本料金が高くなります。契約を変更すると、アンペアブレーカーも交換となります。契約容量を知るには、ブレーカーの色や記載された表示、もしくは電力会社からの請求書や領収書などの記載を確認します（写真1の左部分参照。60A契約にしています）。

契約電力については、どの程度機器で使用したいか、また生活でどの程度必要か見極めたうえで、容量を検討しましょう。これは同時に使用する電力が、契約電力の範囲に収まる必要があります。一人暮らしであれば、30A（アンペア）程度、世帯持ちの一戸建てや、2LDKクラスになると40A～60Aが一般的のようです。エアコン、ドライヤー、電子レンジなどを同時に使って、ブレーカーが落ちるのでしたら、契約電力が足りていないと考えられます（安全ブレーカーが落ちた場合は除く）。その場合は、使い方や契約の見直しが必要です。

契約可能な最大の契約電力については、電力会社に確認ください。ワンルームなどの場合、設備上の制限のため30～40A程度までしか契約できないこともあります。また、7KVA（70A相当）以上の契約は、アパートやマンションなどの場合難しいことや別途工事が必要になってくるので、注意が必要です。

安全ブレーカーって何ですか?

1つのコンセントにたくさん電化製品をつないだりとか、1つの部屋で壁の複数のコンセントからいろいろな電気製品をつないだときに、ブレーカーが落ちたという経験があると思います。たとえば、テレビを観ながらドライヤーと電気ポットを使ったときに「バチッ」と音がして消えるのですが、部屋の照明は点いているという状況です。これは安全ブレーカーが落ちたのが原因です。一般的に安全ブレーカーは15Aや20Aです。その容量は、ブレーカーからコンセントまで使用されるケーブル、コンセントの形状に依存します。

また、ケーブルに電流が流れると必ず抵抗が発生し、熱が発生します。規定以上の電流を流した場合、

▼写真1　60Aのブレーカー。となりの小さなスイッチは安全ブレーカー

発熱が火災原因となるため安全ブレーカーが設置されています(写真1の右側の部分参照。1つの回路あたり20Aです)。

建物の設計にもよりますが、キッチンは冷蔵庫や電子レンジ、炊飯ジャーといった消費電力の高い機器を使用する可能性があるので、独立した回路の場合が多いです。同様に洗面所も独立しているケースが多いのですが、これはドライヤーや洗濯機の使用を考慮しているためですね。そのほかでは、エアコン用のコンセントは15Aや20Aなど独立していることが多いです。

普通の居室コンセントの場合、負荷の高い機器や家電製品を置くことは、あまり想定されていないのか、または回路数の都合上なのか別の部屋のコンセントをまとめてしまうことがあります。そのため部屋単位で回路が独立していないことも多くあります。どのように接続されているか確認するには、照明器具をコンセントにつないで、ブレーカーのON/OFFをしてみてましょう。それでどこに共有されているか確認してみましょう。たまに予想外な組合せもあります(以前住んでいた部屋がそうでした)。とくにアパートやマンションなどの場合、回路の変更は難しいことが多いので、引越しを考えている場合は最初から要件に合う回路になっている物件を選びましょう。また、大きなリビングの場合は2回路ある場合もあります。

サーバラックへ引き込むコンセントをどうすべきか?

安全ブレーカーから1回路まるっと使ってラックに引き込むのが「吉」です。理由としては、誤って他の家電をつないでブレーカーを落としてしまったり、電圧降下したりして、機器の動作不良が発生するなどの外的要因を減らすことができるからです。

部屋単位で独立した安全ブレーカーだと、さらに都合が良いですね。1回路まるっと使えるならば15Aは専有できます。また、建物や部屋によっては、エアコン専用の回路も別に敷設されているケースもあります。この場合は15Aか20A使えることが多いですが、200V電源の可能性もあるので注意が必要です。

次に、ラックへ機器を搭載するにあたり、どの程度の消費電力が必要なのか計算してみましょう。24ポートや48ポートのL2スイッチは100W未満なモノが多いですし、家庭向けのネットワーク機器は10W以下が多いですが、ラックに搭載するとなると、業務用の高性能なルータやL3スイッチなど、それ以上電気の消費する機器なども出てきます。ネットワーク機器については、ほぼカタログスペックの電力を常時消費することが多いです。最近は消費電力を抑えるため、未使用ポートなどへの電力を遮断するエコなモノもありますが、サーバマシンほど大きな変動はありませんので、電力計算はしやすいと思います。

また、最近のサーバマシンは電力管理がよくできており、CPUやI/Oの負荷に応じて大きく電力が変動します。アイドルタイム中にワットチェッカーで計測してみると100W[注1]も消費していないのに、負荷を掛けた瞬間に200W近く消費してしまうケースも少なくありません。たくさんのサーバを設置して負荷を掛けると、ブレーカーが落ちてしまうケースも十分に考えられます。十分に余裕を持って搭載量を計算しましょう。

運がよいと工事なしで、一部屋で30A程度の電力は確保できます。あとは、ちゃんとアースを取りましょう。家庭向けのコンセントでコンセントにアース端子があるところは少ないですが、トイレや洗面所、キッチンなら必ずありますし、エアコン用のコンセントにもアース端子がついていることが多いのでアース線を購入して、必要に応じて引き回しして接続しておきましょう。

まとめ

ネットワーク機器やサーバなどは、常時電力を消費しますので、一般的に生活する電力を20～30Aと計算して、機器で使いたい電力分を上乗せした分を契約電力とするとよいでしょう。一般的な場合、ハーフラック1本～2本の機器をフル稼働させると、ほとんどの場合、先に電力が足りなくなります。それ以上のラックを立てる場合、未使用の機器を搭載するためのラックとなりそうですね(笑)。

最後になりましたが、ブレーカーや回路など電気工事に関わる部分については、第二種電気工事士の資格が必要になりますので、くれぐれも注意してください。SD

注1) 電力(W)＝電圧(V)×電流(A)

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 175 >>>>>

Google I/O 2013で出会ったデジタルガジェットたち

基調講演から感じられた堅実な未来像

米国時間2013年5月15日から17日までの3日間、Google I/O 2013がサンフランシスコで開催されました。Google I/OはGoogleの各種技術に関して、多くのセッションが開かれる開発者向けのカンファレンスです。今年も会場のキャパシティいっぱいの6,000人を集め、さらにネット経由でセッションを視聴する参加者は100万人以上にも及びました。昨年より多い189のセッション、170の展示というこの巨大なカンファレンスには、シリコンバレー近辺の開発者はもとより、ヨーロッパ、アジア圏からの参加者も多くみられました。

今年の印象を一言で言うと「派手さ」よりも「堅実さ」を強く感じました。派手な未来の技術よりも、堅実で確実な、今便利になる改善や新しい機能が数多く知らされたのが初日の基調講演でした。

おそらく多くの参加者が期待していたAndroid関連の話題が全体に占める割合は少なく(後述)、昨年のGoogle I/Oで注目を浴びた眼鏡型ヘッドマウントディスプレイ兼カメラの「Google Glass」は今年はまた違った盛り上がりがありました。

会場の中にはGoogle社員以外にもすでに何人かGoogle Glassを装着している人がいましたが、昨年のGoogle I/Oで購入予約し、いち早く受け取ることができた方々でした。ただし、基調講演ではGlassの話題に触れられることなく、Google音声検索やGoogle Nowの躍進がGoogle Glass浸透への布石の1つのようでした。まだ広く一般に浸透するには準備ができておらず、開発者にいろいろサービスを作ってもらいたいという雰囲気でした。

基調講演の最後には、声帯の病気で療養していたGoogle CEOのLarry Page氏が登場し、皆を安心させました。少し枯れた声でゆっくりと「Googleのことを他社との比較で語らないでほしい。それはつまらないこと。成果を出すにはネガティブではいけない。いままで世の中に存在しなかったものを作り、テクノロジーを使って世の中をもっと良くしたい」という気持ちを会場に伝えていました。

そうは言いつつも今回の基調講演で発表された数々の事柄は、新しいというよりも既存の延長線上にあるものでした。それは会社として確実にビジネスになるところに資金や人的リソースが投下されているということなのでしょう。とくにGoogle Mapsを始めとするGEO関連(地図や位置情報関連)の進化と広がりが数多くアピールされていました。

センサーを使った会場実験

さらに、今回のGoogle I/O初の試みの1つに、会場中に張り巡らされたセンサーネットワークがありました。Google Cloudチームが、O'Reilly Data Sensing Labと共同で、今回のために制作したシステムです。オープンソースハードウェアのマイコン

会場となったMosconeセンター。昨年とは違うMapsのピン

会場入り口に設置された巨大なGoogle I/Oロゴ

Google Street Viewの撮影機材(左:山岳向け、右:海底向け)

ボード「Arduino」を元にした環境センサー525台を、会場のいたるところに設置するという試みでした。各環境センサーのデータ通信には、XBeeベースのメッシュネットワークが構築されました。基板むき出しのセンサーは、Google Cloud Platformに構築したソフトウェアと連係し、会場中の温度、湿度、騒音、フロアの振動、空気の汚れ、明るさなどを集計し視覚化するという壮大な実験でした。

たとえば会場の中でひと休みするときや電話（通話）をしたいときに、会場の静かな場所を探すなどの用途に活用できます。Googleのミッションである「世界中の情報を整理し、世界中の人々がアクセスできて使えるようにすること」を明確に具現化した試みでした。

Androidをとりまく状況

基調講演では、Javaの開発環境で知られるIntelliJベースの新しいAndroidアプリ開発環境「Android Studio」の発表がありました。Androidのエコシステムは巨大に増殖しつつありますが、勝者と敗者が明確に分かれつつあります。そうは言っても、今の勝者も安泰ではなく、いつ入れ替わってもおかしくはありません。そのうえ、Android端末のハードウェア、OS、ソフトウェア、フレームワーク、ユーザインターフェースなどが複雑に絡み合い、単にプログラムで何かが作れるといっただけでは不十分になってきました。

また、今回明確に言葉にはなっていませんが、メッセージとして受けとれたのは、今はAndroid三昧だが、今後（ネイティブの）Androidアプリは緩やかになくなっていくと思われることでしょう。Android部門のトップがこれまでのAndy Rubin氏から、Chrome担当のSundar Pichai氏に統轄されることになったのも今後の方向性がわかる出来事です。

今すぐではありませんが、近い将来すべてはWebアプリ、ブラウザを中心とするChrome OSに統合されるのかもしれません。今回のGoogle I/OでもChrome Packaged AppsやWebコンポーネントに期待する声が大きく、ブラウザは動作環境となり、Chrome Runtimeのような位置づけになるのかもしれません。Javaの遥か昔から、さまざまなテクノロジーがどこでも同じものを動かそうとする夢想を追っていましたが、WebアプリとChromeがまた、どこでも動く環境を求めはじめていることが感じられました。

巨大な会場と、興味深いセッションの数々

189に及ぶセッションは、Googleの各種テクノロジーの詳細をGoogleのプログラマ／エンジニア、デザイナに紹介してもらえるセッションで、今年はとくにChrome/Web関連のセッションが数多く開催されました。WebやAndroidスマートフォンのユーザインターフェースの作り方、考え方だけでなく、それらの背景にある認知心理学の応用の仕方や車載アプリへの展開など、広く深く内容の充実したセッションが数々用意されていました。

Google I/O後、すぐに各セッションの映像がYouTubeで公開されています（英語字幕を一緒に見るとわかりやすく、機械翻訳で日本語字幕を見ることもできます）。

Google I/Oセッションビデオ一覧
https://developers.google.com/live/io/

1階から3階の巨大な会場は想像以上に広く、展示ばかりではなく気楽に休憩する場所としてのラウンジやGoogleの開発者に直接質問ができるオフィスアワーなど、開発者同士の交流の場も設けられていました。

Googleのパワー

Google I/OではGoogleで働く人々（Googlerと呼ぶ）から現場の話を聞くことができるとともに、オフィスアワーのコーナーでは直接質問したり、意見を伝えたりすることができます。

そこでわかったのは、Googleといえども、ごくごく普通のことを、大量の有識者が専任で、良い環境で、余裕を持って、かつものすごいスピードでやっているにすぎないことでした。何か特別な魔法があるわけではないのです。ただし、その知見や人材の広さと深さは素晴らしく、さまざまな専門家が集まることで新しいことができる環境が生まれ、また課題や問題を次々と解決していく環境を創り出し、また維持しているのだろうということがわかりました。

活躍の場は広がっている

さらに今回強く感じたのは、すでに数多くのサービスや先端的な技術がありますが、開拓できていないミッシングポイントや解決しなければいけない課題は、じつはまだまだたくさんあると

Mt.Viewにある、Googleストアの出張販売


展示会場風景。ここで見えている8倍くらいの面積がある

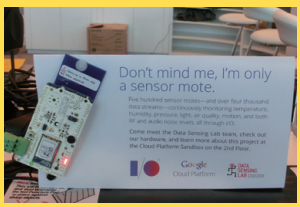
空中に浮かぶGoogle飛行船。搭載カメラからの映像が送信されている


Chromebook Pixelが大量に設置されたディスプレイ

いうことです。日本の技術者やデザイナが勝負できる深みや繊細さもまだまだたくさんあります。ただし、世界が持つ広さと厚みと、その情報のスピードは圧倒的に負けています。

エンジニアだから、デザイナだから、という区分けはもう無意味であり、エンジニアもデザインの勉強をするし、デザイナだけれどもコードも書きます。最初はどちらもたいしたレベルに到達できないでしょうが、それでも続けていくことで、あるとき超えられない壁が超えられるようになるのではないでしょうか。エンジニアはもちろん一流のビジュアルデザインはできないかもしれませんが、デザイナの気持ちやこだわりがわかるようになります。デザイナは一流のコードは書けないかもしれませんが、エンジニアの気持ちやロジックがわかるようになります。その理解とコミュニケーションの効果は絶大であり、今後は専門分野を持ちながらも広くスキルを持った人材が活躍していくことでしょう。

Google I/Oを始めとして、開発者向けカンファレンスもオンラインでさまざまな情報が瞬時に公開されるようになってきました。とはいえ、スマートフォンとタブレットが人々の生活にごく普通に浸透しつつあり、とくに米国都市部ではGoogle Maps、Foursquare、Facebookはもう生活のインフラとして浸透している印象を感じるなど、やはり現地に行かないとわからない温度感や人と人との出会いなどがあります。今年も大変充実したGoogle I/Oでありました。SD



会場内525ヵ所に設置された基板むき出しの環境センサー

gadget 1

NVIDIA SHIELD

http://shield.nvidia.com

ゲーム専用Android端末

グラフィックスチップメーカーとして知られるNVIDIAから、最新のTegra 4搭載、Android 4.2.1搭載のゲーム専用機SHIELDが展示されていました。HDMI出力、5インチ、1280×720マルチタッチ液晶を備え、350ドルで5月20日より先行予約開始、6月には出荷されるそう。重さは579gと、見た目に違わず少しごつい印象。Android OSの面影はあまりなく、ゲーム専用機の印象が強いものでした。

gadget 2

Google Chromebook Pixel

http://www.google.com/intl/ja/chrome/devices/chromebook-pixel/

超高精細ブラウザ専用ノートパソコン

2560×1700の解像度、400nit※の明るさ、178度の広視野角な液晶を搭載したGoogle Chromebook Pixel。最新の薄型ノートパソコンに比べると1.52kgと少々重たいですが、約5時間分のバッテリーを搭載し、Googleアカウントでログインすればすぐに使い始められる便利さがあります。北米仕様のマシンはLTE SIM搭載可、SSD搭載、1TBのGoogle Driveの利用権も付随します。キー配置の変更ができ、Google日本語入力もインストールされています。価格は1299ドル。

gadget 3

Rethink Robotics BAXTER

http://www.rethinkrobotics.com/

ものまねロボット

Rethink Robotics BAXTERは汎用の産業用ロボットです。人が動きを教え込んで、その動作をくりかえすロボットです。顔の部分に設置されたディスプレイに愛嬌のある目や顔が表示されます。お掃除ロボットRoombaの産みの親であるロドニー・ブルックス氏の会社です。特徴は、普通の人でもロボットに対して手取り足取り教えることで、難解な設定ができてしまうことです。価格は22,000ドルから。

gadget 4

OculusVR

http://www.oculusvr.com/

ゲーム用VR ヘッドマウントディスプレイ

OculusVRは両目で立体視、頭の動きをトラッキングする機能のついたヘッドマウントディスプレイ（HMD）です。従来のHMDとの大きな違いは、左右90度という視野の広さです。解像度は640×800ピクセルの左右あわせて2画面。圧倒的な没入感と手軽さと、超低遅延6軸ヘッドトラッキングセンサーによる追従性の良さにより、新しいタイプのゲームの登場が望まれます。入力はDVI/HDMIおよびUSB接続。価格は300ドル。会場ではほかにもRecon社のHMDも人気でした。

※nit…1nitは1平方メートルあたり1カンデラ（1cd）の明るさ。

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

2

結城浩の再発見の発想法

Threshold

Threshold—スレッショルド

スレッショルドとは

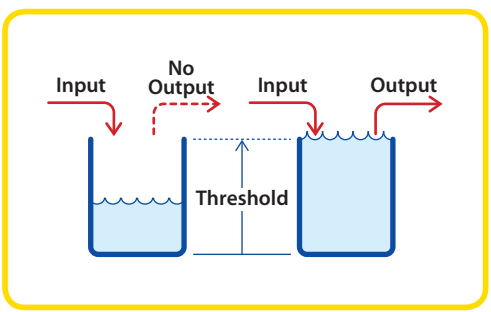
水道からコップに水を注いでいくと、コップに水がたまります。コップの縁を越えて水が溢(あふ)れるまでは周りは濡れませんが、溢れたとたん周りが濡れますね(**図1**)。今回は、そのような現象のお話です。

スレッショルド(threshold)とは、日本語では**閾値**と呼ばれています。「いきち」や「しきいち」と読みます。英語読みでスレッショルドあるいはスレショルドということもあります。

スレッショルドとは「その値を越えるまでは何も起きないが、その値を越えると変化や反応を起こす値」のことです。**図1**のコップの例なら「周りを濡らさずにコップにためることができる水の量」がスレッショルドになります。

スレッショルドを持つものの入力(Input)と出力(Output)の関係は、**図2**で表せます。入力がスレッショルドを越えるまで出力は何もありません。

▼図1　コップを用いたスレッショルドのイメージ



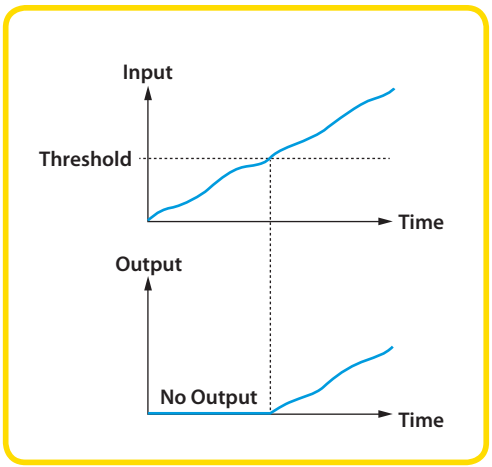
デジタル化とスレッショルド

コンピュータは「オン(On)とオフ(Off)という2つの状態を使って計算を行うデジタル装置」ですが、そのオンとオフはコンピュータ内部では電圧などの度合いで定まります。アナログな電圧を、デジタルなオンとオフとして扱うためには、スレッショルドが必要になります。すなわち「この値以上はオンとみなし、この値未満はオフとみなす」という値です。

2つのスレッショルド

スレッショルドに達するまでは、入力の変化

▼図2　スレッショルドを持つものの入出力の関係



は出力にまったく影響しません。ただし、入力がスレッショルドの近辺に達すると、入力のゆらぎは出力に影響を与えます。

たとえば、オフとオンを識別するスレッショルドが0.5Vだったとします。すると、電圧が0.5V近辺をふらふらと上下したら、それに合わせてデジタルの値もオンとオフをふらふらしてしまいます。この様子を**図3**に示します。

このような出力のふらつきを防ぐために、「オフ→オン」と「オン→オフ」で異なるスレッショルドを設定することがあります。こうすると、オンとオフは安定したままふらつきません。この様子を**図4**に示します。入力がふらついても、出力のふらつきは少なくなっていますね。

感情発露のスレッショルド

人が感情を外に見せる度合いにもスレッショルドがありそうです。いわば**感情発露のスレッショルド**です。その人に対するストレスや仕事の量などを入力とし、その人が外に見せる感情の度合いを出力と考えるということです。

我慢強い人(感情発露のスレッショルドが高い人)は、高いストレスがかかっても感情的な反応を外に見せませんが、我慢強くない人(感情発露のスレッショルドが低い人)は少しのストレスで感情的な反応をすぐに見せます。

もちろん、感情発露のスレッショルドが高いほうが良いか、低いほうが良いかは一概に判断できません。我慢強いほうが物事がスムーズに進むように見えますが、外からその人の状態を知ることが難しくなりますね。

ブレークのスレッショルド

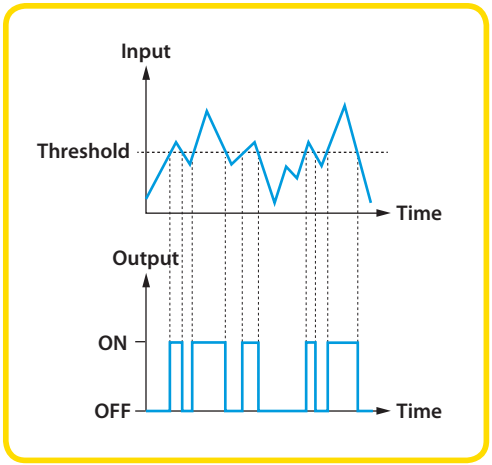
商品の売り上げに関して「ブレークする」という表現が使われることがあります。宣伝のためのコスト(入力)をかけても、なかなか売り上げ(出力)が上がらない。ところが、ある時点から急激に売れ始めるという状況です。この「ブレーク」も何かのスレッショルドを越した瞬間なのかもしれません。

スレッショルドがあると、入力の変化に応じた変化が出力に現れません。変化を早めに察知したいなら、スレッショルドを下げるか、中間的な出力を観察する必要があるでしょう。たとえば、売り上げという出力の前に、WebページのPVを見たり、ツイッターでの言及数を調べたりするのはそのような中間的な出力を観察していると言えるでしょう。

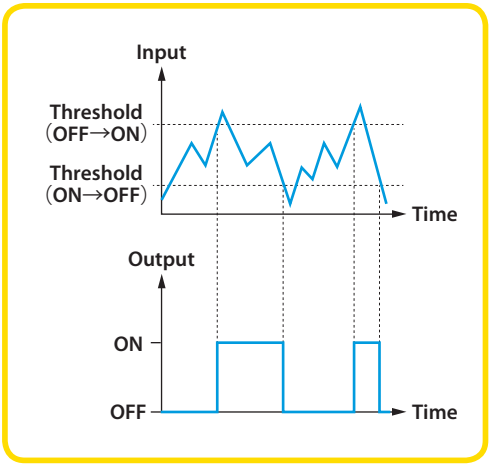
◆ ◆ ◆

あなたの周りでスレッショルドと呼べるものはありますか。そこでの入力と出力は何でしょう。ぜひ考えてみてください。SD

▼図3　入力のゆらぎが出力に影響する



▼図4　2つのスレッショルドで出力のゆらぎを防ぐ



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3回 秋葉原にNASAをつくる

田中諒が後にenchant.js(エンチャント・ジェイエス)と呼ばれるものを作り始めたころ、僕は近藤誠、伏見遼平(先に現地入り)、鎌田淳二、辻秀美らとともにサンフランシスコに向かっていました。ゲーム開発者会議、通称GDCと呼ばれるイベントに出席するためです。

コード・オン・ザ・パシフィック

朝、京成上野の駅前で待ち合わせてスカイライナーに乗り込むと、近藤は冗談まじりに言いました。

「清水さん、今回は飛行機のなかで何をやりましょうか」

近藤は学生アルバイトだった時代に、やはりサンフランシスコで開催されるMacWorldへ向かう飛行機の中で、僕にけしかけられてアプリを作ったことがあります。このアプリは毛筆の書き味を再現するもので、飛行機の中で作られたとは思えないほど良くできていました。到着後すぐに『i書道』という名前でリリースしたアプリは、お正月という時期もありApp Storeで1位を獲得。その後、AppleのCMにも採用され、世界中のApple Storeにi書道のアイコン(僕が飛行機の中で開発中のアプリで書いた「書」の文字)が飾られるなど注目を集めました。

そういうことがあったので、今回も何か変わったことをやらないか、という近藤なりの冗談だったのですが、僕は当時HTML5の可能性に注目していたので、ちょっと変わった提案をしてみました。

「今回の出張で集まったのはUEIのなかでもトップレベルのプログラマたちだ。今回は8時間の飛行時間でゲームを10本作ってみよう[注1]」

「これ、清水さんはやらないんですか？」

確かに普段、僕は偉そうなことばかり言ってるのですから、ここは逃げるべきときではないのかもしれません。無様に負けてしまうリスクはあるものの、僕にもプライドがあります。

「よし、受けて立とうじゃないか。今からスタートだ。終了は飛行機が着陸するまで」

号令一下、一斉にプログラミングを始めました。僕も上司としてのプライドがかかっているから真剣です。が、6本くらい開発したところで残念ながらMacの電池が切れてしまいます。これは悔しい。結局、誰も10本作ることはできませんでした。

そこで2回戦として、12時間で10本作ることにしました。プログラムのスプリントレースです(のちにこの突発イベントは『環太平洋ミニゲーム開発オリンピック』と名づけ、動画も公開しています[注2])。

注1) 連載第1回で「8時間で8本」と書きましたが、正しくは10本でした。お詫びします。
注2) http://www.youtube.com/watch?v=Z6B5bgWtPew

プログラミングほど素晴らしい遊びはない

今度は13本作ることができました。それで次からは僕は抜けて、現地で合流した伏見も参加することになりました。本当は企画が本業で来た辻にも「僕が教えるから君も作りなさい」と無茶なことを言ってみました。するとなんと辻は、翌日には立派にゲームを作れるようになっていました。

何度もこういうスプリントレースを繰り返すと、アイデアがどんどん湧いて来ることに驚きました。とくに鎌田君は凄まじく苦し紛れのアイデアでみんなを爆笑の渦にたたき落としたし、伏見は独特なノリで高度なゲームを作り上げました。

なんて楽しいんだろう――

久しぶりにコードを書く快感を感じた僕は素直にそう思いました。アイデアを考え、コードにして、遊んで、それを誰かに見せて、笑ってもらう、楽しんでもらう。

プログラマをヒーローにする

ニコニコ動画のように、自分たちが作ったゲームを自由に発表できる遊び場。投稿したゲームを自分や友達に遊んでもらって、感想をやりとりできる場所。HTML5ならそれが可能です。とりわけスマートフォンなら簡単です。

よし、帰ったらこういうイベントをやろう。そう思いました。プログラミングの楽しさをもっと多くの人に伝え、プログラマがそこでヒーローになって、大きな飛躍を達成するような。

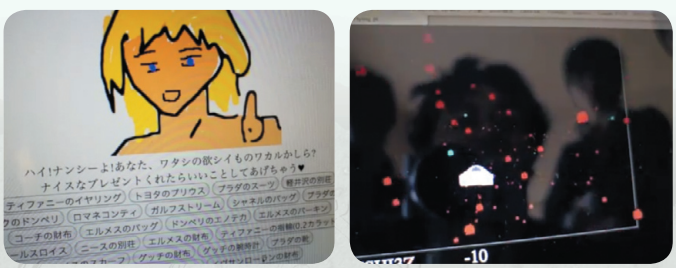
それで“leap”という単語が頭に浮かんできました。“飛躍”という意味の英単語です。leapだけでは商標的に問題があるので、数字をくっつけて「9leap」としました。ナイン・リープです。9は、あと1で桁が変わる、そういう数字です。まだ何者でもない人たちをちょっとした後押しで大きく飛躍させる。それで9leapという名前を考えました。一応、頭字語にするためにLeaders Education and Achievement Programと名付けました。リーダーの教育と表彰を目的としたプログラム、です。のちにNTTドコモと電通のジョイントベンチャーであるD2Cさんが賛同してくださり、当面の資金提供を約束してくれました。

テキサス州オースチン

話を米国出張に戻しましょう。GDCのあと僕と辻は他のメンバーと別れて、テキサス州オースチンに向かいました。そこでまた別の社員と合流です。サウス・バイ・サウスウエスト(SXSW)という全米で最も大きな、音楽と映画とインターネットの祭典に参加するためです。

これは頓智・(当時)の井口尊仁さんに誘われたイベントでした。TwitterやFoursquareがデビューした凄いイベントがある、UEIもこういうところで勝負すべしと。それで僕ものこのこやってきたわけです。展示するのは「Zeptopad FOLIO」という製品でした。これはNECが開発中のLT-Wという2画面Androidタブレットに、高度なメモ機能を載せたもので、まだプロトタイプでした。辻はこのZeptopadの企画と仕様設計を入社以来ずっと担当してきていました。次世代の文房具を作りたい、という気持ちを僕はとても強く持っていました。

SXSWが始まるまで少し時間に余裕があり、



8時間で10本という縛りで作られたハイ&ローゲーム(左)と3Dゲーム(右)

それまで僕はボートに乗ったり、セグウェイで町を巡ったりして時間を過ごしていました。セグウェイでオースチンの町を走っているときに、この計画の全容を思いついたというのは連載の第1回でご紹介したとおりです。

僕は新しい思いつきに胸をワクワクさせました。ところがまさにこのとき、愛する祖国、日本ではとんでもない事態が起きていたのです。

悲劇

夕方、ホテルの部屋でテレビを見ているとブレイキング・ニュースが入りました。よほどの大事です。何事かと思って画面を凝視すると、映っていたのは文字どおり信じられない光景でした。日本で大地震が起き、津波が東北地方を壊滅させていると。

僕はすぐ現地にいるスタッフに電話をして、家族の安否を確認するよう言いました。しかし当の僕自身も、家族と連絡をとることはできません。会社とはGmailのチャット機能で連絡ができたので、帰宅命令を出しました。しかし、絶望的な気分になったのはその後です。「原発が危機的状況にある」――福島の原発に万が一のことがあれば、東京も危ないはずです。僕は祖国から遠く離れ、なす術もありませんでした。愛する人たちを守ることができず、自分の無力さを悔いました。もちろんあのとき、あれをどうにかできた人などいないでしょう。しかし本当にあれは絶望的な状況そのものでした。

翌朝、現地に集まった日本人を井口さんが招集しました。その場で募金支援することが決定し、SXSWの事務局の協力も得て、最もいい場所に募金ブースを設置できることになりました。募金はうまくいき、集まった募金は3日間で2,000万円にも達しました。そして現地の友人に頼んで赤十字に募金することができました。

家族とは翌日連絡がとれました。妻の実家はなくなってしまったけれども、幸い、全員の無事が確認できました。社員にもけが人はなかったのですが、今度は放射能の恐怖に怯える日々が始まりました。原発は完全に予断を許さない状況にあり、いつ爆発してもおかしくない状況でした。僕は現地スタッフを日本に帰すのが正しい判断なのかどうか、本気で迷いました。

震災後の東京

そして僕はようやく東京に戻ってきました。あの悲劇が起きてから、1週間後のことでした。

こんな状況でゲームみたいなことを仕事にするのは不謹慎な気がして、しばらくは仕事をする気が起きませんでした。ところが実際に震災で被害に遭った義弟と話しをして、感想はまったく逆のものになりました。

「会社もなくなっちゃったし、仕事もないし、原発ぁ仕方ないけど、ほんと、やることがないんだよ。毎日毎日なんにもすることがなくてさあ」

彼らはもはやこの悲惨な日常をなんとか忘れたいのです。なんとか日常を忘れ、次に自分たちがなにをすればいいのか、示してほしいのです。生きる希望を欲しているのだ、と僕は思いました。僕にできることがあるとすれば、それはゲームを作ってただ与えるのではなく、“ゲーム作りという遊び”そのものの素晴らしさを伝え、それで生活していけるスキルを獲得するきっかけを与えるということです。

このとき、9leapのアイデアがにわかに現実味を帯びてきました。コンピュータさえあればどこでも誰でもゲームが作れるHTML5という環境。それをどんな場所でも遊べるスマートフォンという環境。みんなでゲームを作って、作り方を覚えて、いつしかそれでプロになれるような世界――そういうものを創りだせるのは、今は僕しかいないのだと。そして僕は決めました。この事業に本気で取り組むということを。

ARC誕生

被災地から東京に戻り、臨時の取締役会を開

急遽チャリティTシャツをSXSWで販売。Tシャツの購入と同時に募金してくれる人がほとんどだった

きました。お馴染みの面々とも久しぶりに顔をあわせた日でした。僕はもったいつけた所作で椅子に座り、こう言いました。

「秋葉原にNASAを作ろうと思う」

「宇宙事業やるっての？」

「違う。世界最高の知性の象徴としてのNASAだ。知っているか？　アポロ計画の航空管制官、ジーン・クランツは当時35歳だったってことを。飛行計画についてはアメリカ大統領よりも強い権限を持つその人物が、わずか35歳。これがどういうことか解るか？　つまり、合衆国の最高の最高の最高の知性を集めた組織の現場リーダーは、35歳なんだ。新しいことを成し遂げるには、優れた人間を集め、教育し、信任する。年齢なんか関係ない。単純に能力だけが問題だ。それがNASAの凄さであり、アメリカ合衆国の凄さだ」

「そこで何をするつもりで？」

「月に人類を送り込む……それは冗談だが、ソフトウェアという惑星に囚われた我々が、“次の段階”へ進むための偉大な計画を進めるのだ」

「その組織の名は？」

「ARCだ」

「アーク？」

「秋葉原リサーチセンター、通称ARC。我が社の最高知性を結集し、世界で最も進んだ組織を作るんだ」

「ソニーのCSLみたいな？」

「ああいうのとは違うよ。ARCには当面、プロの研究者は置かない」

ARCのロゴマーク。古代ギリシャ語で「幾何学を知らぬ者入るべからず」と書いてある。プラトンの興したアカデメイアに由来する。ARCが単なる事業部門ではなく教育部門であることを示す

頭の中にちらりとある計画が過(よぎ)りました。が、その考えをいったんは追い出して、ひとまずARCはプロの研究者なしの組織としてスタートすることにしていました。

「やるのは構いませんが」

一番重い売上げ責任を持っているCTOの水野拓宏が言いました。事実上、UEIの主要な事業は彼が握っています。水野が反対するならなんとしても説得しなければなりません。

「やるのは構いませんし、やってほしいと思っています。けど、1年以内に何か成果を見せると約束してください」

水野の立場からしてみれば、現場から不満が出るのを心配しているのでしょう。なぜその組織があるのか、誰の目にも明らかな成果を見せなければ、そんな組織を運用維持するために自分たちはダシにされてると思ってしまうかもしれません。

「わかった。約束しよう」

そして2011年4月1日付けで新組織、秋葉原リサーチセンター、通称ARCがスタートしました。所長は僕が兼務することにして、生え抜きのプロパー社員を中心に据え、その下に伏見や高橋、近藤、辻といった若い人間を入れました。ARCの大番頭を努めるのは、元電通でベンチャーキャピタルに勤務した経験もある柿添。

enchant.jsも9leapも、すべてこのARCの管轄となります。そして2011年4月17日、enchant.jsがリリースされ、5月1日には9leapがスタートします。スタート後、わずか10日で100本もの作品が集まりました。9leapは、まるでロケットのような勢いで始まったのです。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3回

エルゴノミクスキーボードの雄

KINESISコンタード & Maltron dual hand 3D

写真1 KINESISコンタードキーボード（Advantage pro）

はじめに

今回はキーボード配列を自由自在に変更できるプログラマブルキーボードの1例として、KINESISコンタードキーボードを紹介します。また、プログラマブルキーボードではありませんが、KINESISコンタードキーボードと似た形状を持つMaltron dual hand 3Dキーボードも取り上げます。

KINESISコンタードキーボード

アメリカのKINESIS社の有名なエルゴノミクスキーボードです。現在は、接続がUSBのAdvantageシリーズが販売されており、通常のバージョンのほかにProfessionalバージョン(**写真1**)、Dvorak配列の印字をキートップにしたバージョン、採用しているキースイッチが異なるlinear feelバージョン(**写真2**)などの種類が存在します。

Professionalバージョンには、いくつかの付加機能のほかにボタンが1つの専用フットスイッチが付属します(**写真3**)。ボタンが3つのフットスイッチも別途販売されています(**写真4**)。

特徴

KINESISコンタードキーボードは大まかに次のような特徴があります。

- **お碗型かつ親指を有効に使える形状**
- **格子状のキー配列**
- **プログラマブル機能**
- **メカニカルスイッチを採用**

お碗型の形状と格子配列により腕や手首をほとんど動かさずに打鍵ができます。また、親指で打鍵できるキーが一般的なキーボードよりも多いため、比較的丈夫な親指を有効活用できます。

プログラマブル機能により、あらゆるキーの位置をキーボードの設定で自由自在に変更できます。OS側のキーボード設定は英語配列になっているだけでよく、それ以外はまったく不要です。キー配列の変更自体も特別なソフトウェアを必要としません。PCにキーボードを接続した状態で、programキー＋F12と押せばリマップモードに入ります。コピー元のキー、コピー先のキーと順番に入力し、もう1度programキー＋F12を押せばリマップは完了します。さらに、複数のキー入力

写真2 KINESISコンタードキーボード（Advantage）

写真3 コンタード専用フットペダル（シングル）

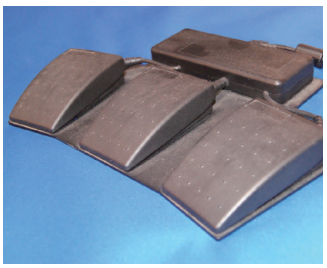


写真4 コンタード専用フットペダル（トリプル）

を1キーで入力するマクロの定義もできます。この設定にも特別なソフトウェアは必要ありません。Professionalバージョンであれば、リマップした配列を第三者が変更できないようにロックすることもできます。

キースイッチには、メカニカルスイッチのCherry茶軸を採用しています。Cherryスイッチは耐久性に優れ、心地良い打鍵感があります。茶軸はキーを押したと認識する前にわずかにひっかかりがあることが特徴です。linear feelバージョンはCherryの赤軸となります。赤軸は茶軸と異なり、認識直前にひっかかりはなく深く押せば押すほど線形に反発が強くなることが特徴です。

入手方法

KINESIS製品には、日本代理店が存在します[注1]。KINESIS社から直輸入するよりも安く、年に何度かセールを行っており、そのときは数%ほど値引きした価格で買えます。Amazonからも購入できます。値段はどちらの場合でもおよそ3万円ほどです。

なお、コンタードキーボードのlinear feelバージョンは、代理店にメールで問い合わせをすると取り寄せてもらえます[注2]。

Maltron dual hand 3Dキーボード

Maltron社[注3]が販売する歴史のあるエルゴノミクスキーボードです(**写真5**)。レイアウトがDvorak配列のタイプや、トラックボールが備わっているタイプ、備わっていないタイプなど、さまざまな種類があります。

写真5　Maltron dual hand 3Dキーボード

特徴

MaltronはKINESISと形状が非常によく似た英字キーボードですが、次のような違いがあります(**写真6**)。

- **ファンクションキーがゴム製ではなく普通のキーである**
- **テンキーが中央に独立して存在する**
- **キーボード配列がわずかに異なる**
- **トラックボールが備わっている**
- **プログラマブル機能がない**
- **親指のキーが窪んだ位置にある**

トラックボールは小さめですのでホームポジションからは遠く感じます。親指のキーのある場所が窪(くぼ)んでいるため、これもKINESISに慣れていると遠く感じます。全体的に手が大きい方向けという印象です。

そのほかに、MaltronもCherryのメカニカルスイッチを採用していますが、こちらは黒軸です。黒軸は基本的に赤軸と同じですが、赤軸よりも反発が強いことが特徴です。

入手方法

日本代理店が存在しますが、値段がかなり高く12万円以上します。本家のWebショップでは435ユーロで販売しているようですが、日本にも発送してもらえるかどうかは不明です。また、稀(まれ)にebayに出品されていることがあります。筆者はebay経由で入手しました。落札価格はおよそ500ドルでした。

◆ ◆ ◆

今回は、プログラマブルキーボード2種を紹介しました。Maltronは多少手が大きい人向けではありますが、両者ともに実績のある有名なエルゴノミクスキーボードで、腱鞘炎に悩んでいるのであれば、導入を勧めることのできるキーボードです。今回は大きく取り上げませんでしたが、Cherryのメカニカルスイッチを採用しているという特徴もあります。Cherryのスイッチについては今後紹介する予定です。SD

写真6　MaltronとKINESISの比較

注1) http://edikun.ocnk.net/
注2) http://edikun.ocnk.net/page/17
注3) http://www.maltron.com/

はんだづけカフェなう

ベイエリアのMaker Faireに出展してきた

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/

Maker Faire Bay Area 2013

先月の深圳（しんせん）でのMaker Faireに続き、今回は世界最大級の規模であるMaker Faire Bay Areaに参加してきました。その名のとおり、アメリカのベイエリア、我々の間ではシリコンバレーとして知られるエリアにあるSan Mateo（サンマテオ）という都市で開催されています。サンマテオはサンフランシスコ国際空港からタクシーでも$30も要さない、比較的近い距離にあります。筆者は、スイッチサイエンスのブースで出展していたので、あまり見て歩くことができなかったのですが、それでもさっと1周はしてきましたので、おもしろかったものを紹介したいと思います。

日本のMaker Faireと共通することなのですが、来場者は家族連れが多く非常にファミリー向けのイベントでした。Maker Faireは英語表現の“Fair”ではなく、“Faire”となっています。これはSan Hose Fantasy Faireという中世を模した祭りの綴りからきているそうなのですが、まさにお祭りが行われていました。出店もいろいろと出ていますが、この出店も日本からすると相当にスケールの大きなものです。たとえば、筆者がほかの出展者の勧めで食べた七面鳥の足をスモークする機械も、**写真1**のような規模です。

もちろん、出展されている物の大きさも日本では考えられないようなサイズのものが数多く見受けられます。Lifesize Mousetrap（**写真2**）という名前の巨大なピタゴラスイッチのような、どうやって会場に運んできたのだろうかと不思議になるようなサイズのものまであります。また小型の潜水艦（**写真3**）も出展されていました。

▼写真1　出店のくんせい機

▼写真2　Lifesize Mousetrap

▼写真3　潜水艦

子供のMake:

最近のアメリカでのMake:シーンは子供向け

の企画が充実してきており、たとえばYoung Makersといったプログラム[注1]が実施されています。中でも空気圧でマシュマロを飛ばすマシュマロ・キャノンが有名で、オバマ大統領がホワイトハウスに作者の少年を招いて、マシュマロを飛ばしたニュースがあるくらいです。興味を持った方は、“Extreme Marshmallow Cannon”で検索をしてみてください。

もちろん、Young Makersの保護者がエンジニアだったりして、彼らの手助けがあるのでしょうが、なかなかに大人からしてもおもしろそうなものが出展されています。ブルーマンの楽器を彷彿とさせる塩ビ管を使った楽器（**写真4**）や、マキタの電動工具を動力にしているホバークラフト（**写真5**）、巨大パチンコ（**写真6**）といった具合です。これもまた、日本の都心の住宅事情では作るのがかなわないような力作揃いでした。アメリカでMake:がもてはやされている理由の1つが製造業への回帰なのですが、日本の製造業が弱くなったのにはこういった工作や遊びをする機会が失われてしまったことにもあるように思えました。

ロボット

深圳でも感じたことなのですが、海外では日本よりもロボットを作る人が数多く見受けられます。やはりアメリカのgeekの間ではスターウォーズが人気のようで、この作品に出てくるロボットが大量に居て、皆喜んで写真を撮っていました（**写真7**）。展示もたいへん上手で、映画に出てくる宇宙船の巨大なパネルの前で写真が撮れるように工夫されています。東京のMaker Faireよりも皆、見せるのがうまいなぁと感じました。また、それっぽい作品にBEER2D2があります（**写真8**）。BEER2D2はビールの樽を使って作られたようなのですが、会場をところ狭しと走り回っていました。

日本でもロボットの競技があるのですが、アメリカの競技ロボットを作っている人達も出展をしていました（**写真9**）。すごいなと筆者の目を引いたのはフリスビーを飛ばすロボットで

注1） http://youngmakers.org/

▼写真4　塩ビ管楽器

▼写真5　ホバークラフト

▼写真6　巨大パチンコ

▼写真7　R2 BUILDERS

▼写真8　BEER2D2

▼写真9　競技ロボット

す。フリスビーを飛ばしてゴールにうまく入れるのですが、さらに落ちてきたフリスビーを回収して補充する機能もついています。自律で動いていたのに非常に感心しました。

コスプレ

会場にはMake:文化をよく知らないアメリカ人でも「変な人がいっぱいいる」というくらい、変わった人で溢れています。コスプレをしている人も多く見受けられました(**写真10**)。かなり本格的なもので、自分で型を作って樹脂部品を作るくらいの力の入れようです(**写真11**)。

ほかにも、ゲームに出てきそうなトゲトゲの着ぐるみの人もいました。ちなみに、トゲは体の動きに合わせて光るようです(**写真12**)。ほかにも、悪魔の格好をした人もいました(**写真13**)。このコスチュームにはボイスチェンジャーが付いていて、声に合わせて口部分の点灯パターンも変化する力の入りようです。

コスプレとは異なるのですが、深圳で会った方も出展していました。Katia Vegaさん[注2]という方で、ウェアラブルコンピューティングなどの研究をしている方です(**写真14**)。彼女のアイラインは導電性の素材でできており、目の開閉でコントロールをしたり、ネイルにRFIDを付けて触れた指を認識できるようにしていたりしました。

hackerspace

筆者があちこち訪ね歩いているhackerspaceですが、彼らもMaker Faireには出展をして自分たちのスペースの紹介をしています。今回はベイエリアにあり、バイオ系という珍しい畑のhackerspaceであるbio curiousに興味があったのでブースを見てきました。少し前のMake: Japanのブログで紹介されていたのですが、彼らはDIY BioPrinter(**写真15**)というのものを作りました。最近、3Dプリンタが話題ですが、細胞を3Dプリントして移植用の組織を作るバイオプリントという技術が最先端では研究されています。このプリンタはインクジェットプリンタのカートリッジを流用して、大腸菌をシャーレの寒天の上に印刷するというものです。もちろん、最先端の技術には到底及ばないものですが、この分野のものをDIYで手軽に楽しめるというのがとても革新的です。このプリンタの作り方は、instructablesという手順共有サイトで公開[注3]されています。このinstructablesでは、おもしろいプロジェクトが公開されています。

注2) http://katiavega.com/

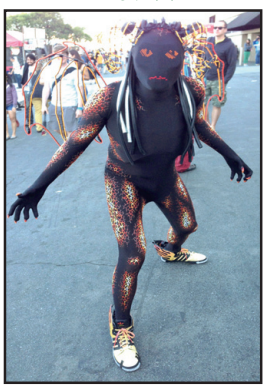
注3) http://www.instructables.com/id/DIY-BioPrinter/

▼写真10 本格的なコスプレ

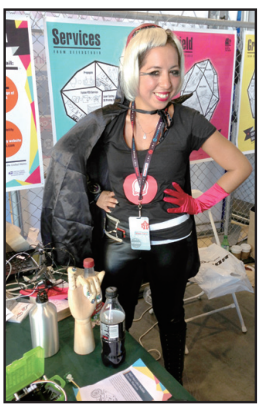
▼写真11 武器の型

▼写真12 トゲトゲの人

▼写真13 悪魔の格好をした人



▼写真14 Katiaさん



日本人出展者

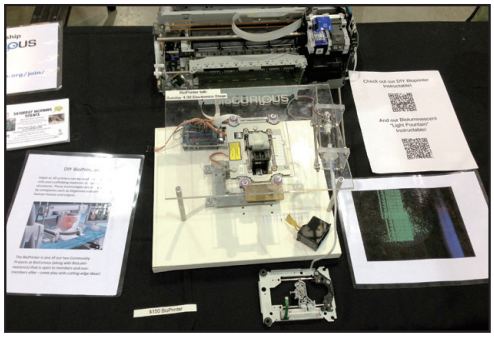
今回のMaker Faire Bay Areaには複数の日本人出展者が参加していましたので紹介します。

ベイエリアに長年居住している、かづひ氏はGlueMotor（**写真16**）という作品を展示していました。かづひ氏は昨年のMaker Faire Bay Areaにも出展しており、去年のEditor's Choiceに選ばれ、Make:本誌の記事も書いたうえ、今年はEducation Awardを受賞して二冠を達成した方です。とても気さくでおもしろいおっちゃんで、筆者もベイエリアのジャンク屋さんに連れていってもらったりしました。

GlueMotorはほとんど回路部品を使わずにiPhoneなどのスマートフォンのイヤホン端子からサーボをコントロールするというものです。とても簡単に作ることができるうえ、スマートフォンのアプリケーションを書くのが作業のほとんどですので、本誌読者には馴染みやすいものだと言えるでしょう。こちら[注4]に同氏が情報を書いていますので、興味を持った方はアクセスしてみてください。

日本から来ている出展者ももちろんいました。伊藤氏（@itog）は“FourBeat”（**写真17**）という、Androidに接続してさまざまなアプリケーションから利用できる押しボタンスイッチを出展していました。これは、日本版Kick Starterとも言えるCAMPFIREでもファンドが成立した作品で、多くの子供がブースには集まってボタンの連打に熱狂していました。

▼写真15　DIY BioPrinter



同じく日本から出展していた方に新里氏（@hirotakaster）は心拍に連動して光るスライム（**写真18**）を展示していました。こちらもたくさんの子供が集まってスライムを弄り倒して遊んでいました。

まとめ

先ほど少し触れたKick Starter[注5]というWebサービスが最近話題になっています。今回のMaker Faireでは、多くの出展者がKick Starterでファンディングをしているという掲示を出して自分たちのプロジェクトをより進める資金の調達に挑戦していました。Intelやトヨタ、シボレー、フォードといった大企業もMaker Faireのスポンサーをするなど、Makerムーブメントが世間一般に与える影響も徐々に大きくなってきているのを感じました。SD

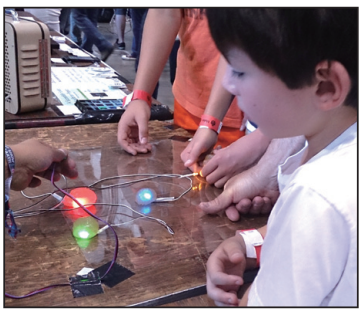
注4）　http://www.gluemotor.com/

注5）　http://www.kickstarter.com/

▼写真16　GlueMotor

▼写真17　FourBeat

▼写真18　新里氏のスライム



PRESENT 読者プレゼントのお知らせ

『Software Design』をご愛読いただきありがとうございます。本誌サイト http://sd.gihyo.jp/ の「読者アンケートと資料請求」からアクセスし、アンケートにご協力ください。ご希望のプレゼント番号をご入力いただいた方には抽選でプレゼントを差し上げます。締め切りは2013年7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 メカニカルキーボード OWL-KB109BM (B)IIR

1名

ドイツのZF Electronics社製のCherryMXメカニカルキースイッチ「赤軸」を採用した日本語109フルキーボード。より軽く滑らかに押し下げ圧が変化します。入力の手ごたえがありつつも、指先への抵抗が少ないため軽快な入力が可能です。

提供元 オウルテック
URL http://www.owltech.co.jp

02 コリャ英和！一発翻訳 2014 for Win

1名

531万語収録の高精度翻訳エンジンを搭載した英日／日英翻訳ソフト。Adobe Acrobat、MS Office（32/64bit）などに翻訳機能をアドインして利用することも可能。対応OSはWindows 8/7/Vista/XP SP2、SP3。

提供元 ロゴヴィスタ **URL** http://www.logovista.co.jp

03 USB メモリ Touch T825

1名

回転式のキャップでコネクタ部分を保護でき、しかも書類やノートを挟んだり留めたりできるクリップ機能も備えた利便性に優れたUSBメモリ。インターフェースはUSB2.0（USB1.1互換）、容量は8GB。

提供元 シリコンパワー **URL** http://www.silicon-power.com

04 ディケイド 小物入れ（S）

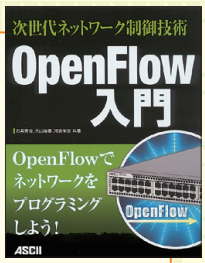
3名

「長く付きあえ愛着がわくプレミアムなデザイン」をコンセプトに作られた小物入れ。机上にあふれる小物をスッキリ収納。使い勝手の良い仕切りつき。サイズはW110 × L97 × H74mm。

提供元 カール事務器 **URL** http://www.carl.co.jp

05 OpenFlow入門

3名



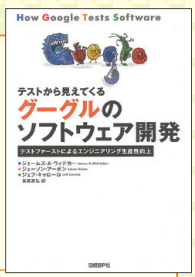
石井 秀治、大山 裕泰、河合 栄治 著／
B5変型判、192ページ／
ISBN = 978-4-04-886953-9

ソフトウェアを用いて柔軟にネットワークを制御できる新しい技術「OpenFlow」。その仕様からプログラミングまでを詳細に解説します。

提供元 アスキー・メディアワークス **URL** http://asciimw.jp

06 テストから見えてくるグーグルのソフトウェア開発

3名



ジェームズ・A・ウィテカー、ジェーソン・アーボン、
ジェフ・キャローロ 著、長尾 高弘 訳／
A5判、432ページ／ISBN = 978-4-8222-8512-8

ほぼ毎日リリースされるというグーグルのソフトウェアはどのようにテストされているのか？ 担当者の証言をベースに、テストと一体化した開発の実像、未来像を詳細に解説します。

提供元 日経BP社 **URL** http://www.nikkeibp.co.jp

07 Software Design 総集編［1990～2000］

3名

Software Design 編集部 編／
B5判、108ページ／
ISBN = 978-4-7741-5714-6

SDの創刊号～2000年の過去記事を収録したDVDと書き下ろし記事が一緒になった総集編。インターネット黎明期の技術情報が満載の本書は、若手エンジニアが技術の変遷を追体験するのに最適です。

提供元 技術評論社 **URL** http://gihyo.jp

第1特集　データの価値を見直しませんか？

ここからはじめるデータ分析学習

― Excel・R・Mahout・ビッグデータ ―

“ビッグデータ”という言葉の印象が強く、ともすると自分には関係ないと思いがちなデータ分析の分野ですが、激しい競争を生き抜くために、新しい機会の創出やニーズの発掘などが必要不可欠な時代です。そこに、大小にかかわりなく、データから有意なものを見つけ出すデータ分析の需要の高まりは間違いなく存在します。

しかし、データを分析し、正しい予想を立てるにはどうしても数学・統計学的な知識が必要になってきます。あきらめるか、学びはじめるか。ここがエンジニアとして強力な武器を持てるかどうかの分岐点です。

本特集で、ソフトウェアエンジニアにとって今後ますます重要なスキルとなる、データ分析の分野を俯瞰しましょう。そして、必要な知識の学び方や書籍の紹介などを学習の助けとし、Excel・R・Mahoutというツールを使ってみることで、分析手法のイメージをつかんでください。

CHAPTER 1

ソフトウェアエンジニアに贈る

データサイエンス入門&習得ガイド

「機械学習やデータ分析に興味はあるけれど、専門的な情報ばかりで何から学んでよいかわからない」という人は多いのではないでしょうか。本章では、ソフトウェアエンジニア向けにデータサイエンスの興隆と概要、そしてデータサイエンスに必要な知識と技術の紹介をし、その習得に必要なガイドを示します。

柏野 雄太（かしの ゆうた）　バクフー株式会社 代表取締役

背景——ビッグデータ狂騒から

最近は落ちついてきましたが、少し前までは「ビッグデータ」の狂騒状態でした。世界中で事情を知らないマーケッターや経済新聞までがビッグデータという大文字を喧伝していました。日本でもHadoopやMapReduceという文字が日本経済新聞を飾った日もあります。しかし、その後2年ほど経っても、ビジネス上に大きな変化が現れないとなると、気の早い書き手が「ビッグデータは死んだ」[注1]と大げさに嘆き、「ぶっちゃけほとんどのデータは大きくないのに、そのフリをして金をドブに捨てている」[注2]と煽りたてる記事を書き始めるようになってきました。

結論からすると、2年経ったからといってビッグデータがなくなったわけではありませんし、これからもデータは大きくなっていきますから、ビッグデータを扱う技術や知識はますます必要になっていきます。既存の組織のデータに加えて、ネットワーク上で顧客と直接取引することによるデータの増加、オープン化する公的データの増加、ソーシャルWebデータの急増とそれを容易に取得する手段の増加、そしてネットワーク化するセンサデータの増加など、データは拡大する方向に向かっています。

そういえば、5年前も「クラウド」のことを「バズワードで実態がないハイプである」と騒いだ煽り屋たちが多くいましたが、今となってはクラウドは完全にサーバサイドコンピューティングを支配しているのはあきらかです。

ビッグデータはクラウドと同様にコンピューティングのあり方を変えます。あまりに大きな流れですので、クラウドと同様、誰もがそこに移行せざるを得ません。移行するのが数年後か10年後かというタイムスケールの問題だけです。好むと好まざるとにかかわらず、大きなデータを取り扱う知識や技術なしではコンピューティングを取り扱うことなど不可能な時代が来ています。そのため、エンジニアとしては大きなデータへの対応ができるようにならなければ、将来は生きづらいエンジニア人生が待っていると思います。

エンジニアが大きなデータを扱えるようになるうえでの一番の問題は、必要な知識や手段がかなり大きな体系であるため、新しいWebフレームワークを覚える程度の手軽さでは身につけることができない、ということです。

データサイエンス

データサイエンスという言葉があります。大きなデータを扱う知識と技術の体系のことで

注1）"Big Data" is dead' http://venturebeat.com/2013/02/22/big-data-is-dead-whats-next/

注2）'Most data isn't "big," and businesses are wasting money pretending it is' http://qz.com/81661/most-data-isnt-big-and-businesses-are-wasting-money-pretending-it-is/

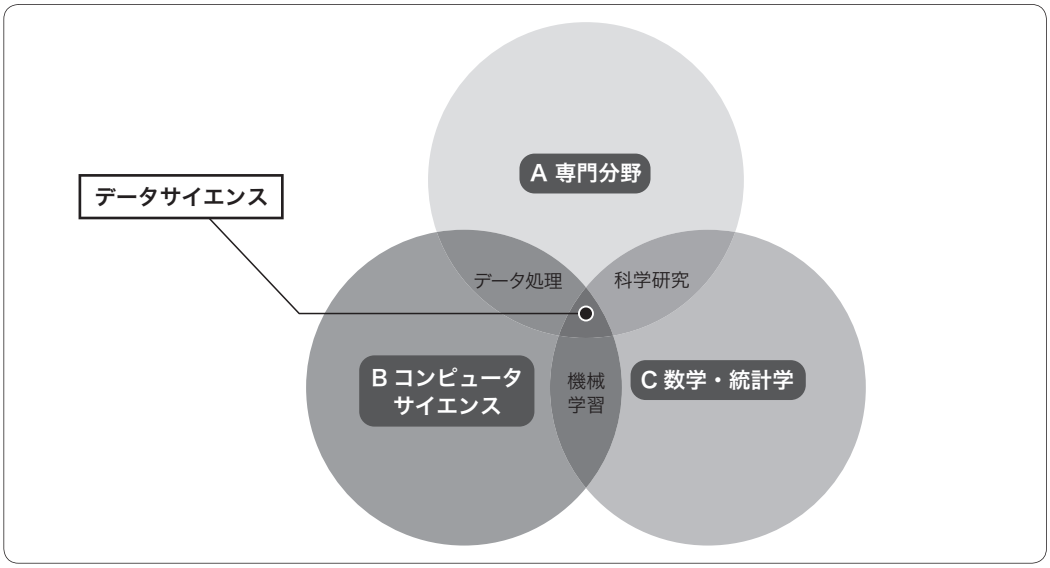
す[注3]。データサイエンスの具体的な実態は、統計学／データ解析、プログラミングスキル／コンピュータサイエンスの知識、実ビジネスや研究分野などの専門知識が密に融合したものです。データサイエンスをやる人はデータサイエンティストと呼ばれています。データサイエンスは専門知識＋数学知識＋コンピュータサイエンスの知識の総合格闘技と呼んでもよく、このことは**図1**[注4]で表すことができます。

まず基本的な3つの領域があります。ビジネスや研究上の専門知識／スキル（領域A）、プログラミングスキルやコンピュータサイエンスの基本知識（領域B）、数学や統計学の知識（領域C）の3つです。この3つの領域が重なっている部分、たとえばAとBの重なりがビジネスや研究で行われるデータ処理の領域です。BとCの重なりが最近興隆してきた機械学習の領域になります。そして、AとCの重なりが昔からある科学研究の領域と言っていいでしょう。データサイエンスはこのA、B、Cの3つが重なる中心にあるものです。このデータサイエンスこそ将来を見据えたエンジニアが取り組むべきものになると筆者は確信しています。

データサイエンスのメッカとなっている米国では、7年も前からGoogleにいる著名なミクロ経済学者Hal Varian氏の“statistics is the next sexy job”[注5]という言葉をきっかけに広義の「統計学」[注6]であるデータサイエンスが注目されてきました。

その一方で、データサイエンスという言葉を独立して使うことに批判的な意見もあります[注7]。たとえば、CERN（欧州原子核研究機構）の

▼図1　データサイエンスの知識体系イメージ



注3） データサイエンスの「宣言」的な最初の論説は、O'ReillyのMike Loukides氏の“What is data science?”だと思います（http://radar.oreilly.com/2010/06/what-is-data-science.html）。この論説にデータサイエンスのエッセンスが詰まっています。

注4） 図1の元ネタは、最近データサイエンスコミュニティに投稿されて話題になった、IAベンチャーズの著名データサイエンティストであるDrew Conway氏のデータサイエンスの定義です（http://drewconway.com/zia/2013/3/26/the-data-science-venn-diagram）。ただ筆者は、Hacking SkillsやDanger Zoneなどはあまり適切なカテゴライズではないと思い、より適切な解釈をしたものが図1なのです。

注5） http://www.nytimes.com/2009/08/06/technology/06stats.html

注6） ここで統計学という言葉が出てきますが、Varian氏がGoogleで主導しているのは、伝統的な主流派の統計学である頻度主義的なネイマン・ピアソン流統計学ではなく、むしろ機械学習で主流となっているベイズ統計学です。

注7） Moshe Y. Vardi “Science has only two legs” http://delivery.acm.org/10.1145/1820000/1810892/p5-vardi.html?ip=110.5.47.50&acc=OPEN&key=1B55DF923F77674F55057ED4F3766CA0&CFID=217973290&CFTOKEN=96019330&__acm__=1368853946_dc072110a74bb3bb1259ba9527e3d119

Higgs粒子を発見するための実験に象徴されるように[注8]、科学は理論と実験に加え、大規模なコンピューティングを行っていますが、それは、昔は人手でやっていたことを今はコンピュータを道具にしてやっているだけで、新しい分野が増えたわけではない、という立場です。

しかし、その理論や実験を行うのに必要なデータの量が巨大化し、科学者が兼業で取り扱えないくらい大変になれば、新しい専門家が必要になります。その専門家こそデータサイエンティストと呼ばれるべき人間です。

事実、2010年ごろに初めてデータサイエンスという言葉が出てきたときは、多くの批判が発せられましたが、2013年の現在では、専門家集団がゆるく形成され、1ヵ月に1回は各地で数百人規模を超えるカンファレンスが開かれるまでになりました。そして、実際に労働市場としてデータサイエンティストの市場が形成された結果、最初のころに出たような批判は鳴りを潜めてきました。

データサイエンスを形作る知識の構成要素

図1にある3つの領域のうちAの専門知識を除いた領域は、統計学／データ解析とコンピュータサイエンスになりますが、初学者にとって問題なのはどちらの分野も敷居が高く簡単には身につかない知識であるところです。

データサイエンティストに必要とされる統計学の知識は、記述統計、推定、検定、回帰などの統計学基礎と、データサイエンスの主流統計学とも言えるベイズ統計学です。コンピュータサイエンスで必要とされる知識は、データ構造を理解すること、複数の分散コンピューティング環境を使いこなせること[注9]、複数のプログラム言語を習得できること[注10]、データ解析のフレームワークを組み合わせられること[注11]です。

統計学とコンピュータサイエンスの嫡男とでもいうべき機械学習は、データサイエンスを「データ（を扱う）サイエンス」たらしめている中心の知識です。機械学習というのは文字どおり、あるタスクをこなすために機械（ソフトウェア／プログラムやハードウェア）に過去の経験を学習させ、そのタスクのパフォーマンスを上げる手法の集合を意味しています。オンライン講義サイト「Coursera」のAndrew Ng氏の有名な講義に出てくるTom Mitchell氏の言葉に機械学習の定義が明確に表れています。「今、プログラム（機械）が、あるタスクTを成績Pでこなすことができた経験Eがあるとします。そのEがあればタスクTの成績をさらに改善できるときに、プログラム（機械）が学んだと言ってもいいでしょう」。

この機械学習の習得が初学者には特に大変なのです。日進月歩の領域であることに加えて、機械学習は、物理学などの伝統科学のような第一原理から演繹的に導かれる知識体系ではなく、いわば巨大なヒューリスティクスの集合だからです。

以上のようにデータサイエンスを習得するのはなかなか骨が折れますが、習得のためのコツがまったく存在しないわけではありません。

学習のコツと学習リソースの紹介

何度も繰り返しているように、データサイエンスは必要な最低限度の知識量がかなり多く、それが学習を困難にしています。このような大量の知識を習得するのに一番いいのは、やはり大学や大学院で講義、演習、輪講、自主ゼミな

注8） CERNのLarge Hadron Collider（大型ハドロン衝突型加速器）は実験が始まると40テラバイト／秒のデータを生成します。そのデータを取得し、フィルターし、ストアし、解析するためには、新しい「科学」と「科学者」が必要でした。

注9） 具体的にはHadoop、Spark、Storm、0mq、Cassandraなどです。

注10） 具体的にはJava、C/C++、Python、R、Matlab、Octave、Scalaなどのデータ解析でよく使われる言語です。

注11） 具体的にはscikit-learn、Mahoutなどのメジャーなフレームワークのことです。フレームワークの網羅的な列挙はmloss.orgのソフトウェアカテゴリ（http://mloss.org/software/）にあります。

どの機会を利用して、時間をかけてじっくり教科書を潰し、演習をし、プログラムを書くことです。

しかし、普段は仕事があるという場合には、なかなか学生のように学習をすることができません。そこで一番お勧めするのは、最近進展の著しいオンライン教育リソース(Massive Open Online Course：MOOCと言います)[注12]を積極的に利用することです。これならば最小の労力で、日本にいながら米国のトップスクールで行われている講義を聴講することができます。もちろん、講義は英語で行われるという問題がありますが、それほど難しい英語は使っていないですし、エンジニアとしてはこれから英語に向き合わなければ生きていけるはずもありませんので、学部の理系講義くらいの英語は聞き取れる、読み取れるということを前提に以下では話を進めていきます[注13]。

また、今回は書籍についての紹介も割愛します。紹介したい良い本がたくさんあるのは事実ですが、本だけで独習するのかなりの労力が必要で、たくさんの知識を短期間で身につけるのは難しいからです。

さらにデータサイエンティストとして必要なコンピュータサイエンス周りの知識、メジャーな言語、データ解析／機械学習フレームワーク、そして最適化や探索アルゴリズムなどのコンピューターサイエンスの知識の紹介も割愛させていただきます。本章では、統計学や機械学習のMOOCのリソースを紹介することを中心に記述します。

早速ですが、ズバリMOOCを利用する場合のコツは次のものになります。

・ビデオはいつでも視聴できるようにダウンロードする

CourseraならPythonベースのcoursera-downloader[注14]がお勧めです。YouTubeならFirefoxなどのオープンソース系のブラウザを使うとダウンロードエクステンションが充実しています。もしもダウンロードできない場合でも、スクリーンレコーディングソフトを利用すれば講義をいつでも利用できるようになります[注15]。Courseraの講義はだいたい最大15分程度になっています。そのくらいしか集中が続かないということで作られています。15分ならスキマ時間で視聴できます

・ビデオはどこでも視聴できるようにタブレット端末に入れる

筆者はNexus 7にいれて、客先に訪問するときでも、お風呂に入っているときも視聴しています。こま切れで見られるように、動画再生プレーヤは再生位置を覚えておいてくれるものがいいです。Android上で動作するMX動画プレーヤはその意味でとても重宝しています

・演習やハンズオンで理解を深める

講義は飛ばし見をしてもかまいませんが、自分が重要だと感じた項目の演習やハンズオンはできる限りやりましょう。手や頭を利用して内容を定着させる意味もありますが、演習をやればその解答を得るために何度か講義を再視聴することもあり、内容の理解が深まることが多いのです

・完璧主義はやめる

研究者の訓練を受けるのではなく、あくまでエンジニアとして基礎知識を手っ取り早く手に入れるのが目的ですので、完璧な知識は必要ありません

それでは、次からは具体的なMOOCのコースや、日本語でも学習できるWebサイトの紹介をしていきます。

記述統計学／推定／検定／回帰

これは統計学のイロハと言いますか、この知識がないと言葉が喋れないのと同じ、というく

注12) http://en.wikipedia.org/wiki/Massive_open_online_course

注13) 残念ながら、2013年5月時点ではデータサイエンスに関する日本語のオンラインビデオのコース講義は皆無です。

注14) https://github.com/siddharthasahu/coursera-downloader

注15) 検索エンジンで「screen recording software open source」などの検索ワードで探せば、オープンソースのスクリーンレコーディングソフトを見つけられます。ただし、サイトによってはストリーミング視聴のみを許可しているところもありますので、サイトの許諾説明を確認してください。

らいに重要なものなのですが、非常に退屈です。そこでサクッと飛ばしながらMOOCで学んでしまうのがいいと思います。edxのStat2.1x、Stat2.2x、Stat2.3xのコースが簡潔かつ十分な内容で、確率変数、期待値、分散、ベイズルール、推定などの基本概念を1週間ほどで身につけてしまうといいです。

https://www.edx.org/course/uc-berkeley/stat2-1x/introduction-statistics/594
https://www.edx.org/course/uc-berkeley/stat2-2x/introduction-statistics/685
https://www.edx.org/course/uc-berkeley/stat2-3x/introduction-statistics/825

MOOCではないですが、R言語を用いて日本語で統計学のイロハを正確に学べる最適なリソースが、群馬大学社会情報学部の青木繁伸氏のサイトです。中でも「統計学自習ノート」は統計学の基礎とRを学ぶためには出色の出来だと思います。

http://aoki2.si.gunma-u.ac.jp/lecture/index.html

またこれに関連しまして、R言語関連の日本語情報サイトの定番を紹介しておきます。筑波大学医学医療研究系の岡田昌史氏が運営するRjpWikiです。

http://www.okada.jp.org/RWiki/

ベイズ統計の初歩

データサイエンスはベイズ統計を使わないで済ますことはできません。この初歩を学ぶのに一番いい講義はPyCon USで行われた“Bayesian statistics made simple”のビデオレクチャーです。かなり独特な実装のベイズ統計のライブラリには正直戸惑いますが、簡単なPythonコードを書きながらハンズオンで実装するうちに、ベイズ統計の感覚を2時間程度で学ぶことができます。

http://pyvideo.org/video/1724/bayesian-statistics-made-simple

データ解析初歩

Rを用いたデータ解析の初歩はCourseraの“Computing for Data Analysis”が、手早く入門するのに適しています。

https://www.coursera.org/course/compdata

データサイエンス入門

現在進行中のCourseraに、そのものずばりのデータサイエンスの入門講義“Introduction to Data Science”があります。実践的な内容を目指しているらしく、リレーショナルデータベース、HadoopなどのMapReduce、データクレンジングの手法まで野心的に紹介しています。しかし、データサイエンスはあまりに領域が広いので、入門コースとしては百科事典的に雑駁（ざっぱく）な講義となってしまっているのが残念です。講義と課題の内容の乖離も大きく、時間のないビジネスパーソンはポイントだけを絞って見るだけにすべきかもしれません。

https://class.coursera.org/datasci-001/class/index

機械学習入門

MOOCであるCourseraの共同創業者であり、飛翔ロボットの世界的な研究者でもあるAndrew Ng氏のMachine Learningコースは鉄板のクオリティです。明快な説明、入門として過不足ない構成、Octaveを使ったハンズオン実習など、このコースを視聴すれば、機械学習の基本が身につきます。初めて機械学習を学ぶ人は、何を差し置いてもまずはこれから始めるべきです。

https://class.coursera.org/ml-2012-002/

カリフォルニア工科大学のYaser Abu-Mostafa氏の講義もすばらしい出来栄えです。コースワー

クとして視聴するより、何か1つのコースを手早く終えたあとに、トピック別で見るほうがふさわしいと思います。

http://work.caltech.edu/library/index.html

MOOCではないですが、機械学習関連の日本語情報サイトで一番充実しているのは産業技術総合研究所の神嶌敏弘氏が運営している、「朱鷺の社Wiki」です。

http://ibisforest.org/index.php?%E6%A9%9F%E6%A2%B0%E5%AD%A6%E7%BF%92

さらに、技術評論社のオンラインサイト(gihyo.jp)でも「機械学習はじめよう」という機械学習初心者向けの丁寧な入門講義が連載されています。ときどきPythonを使った実装も提示され理解を助けます。好連載です。

http://gihyo.jp/dev/serial/01/machine-learning

機械学習実践

機械学習の実践講義として、scikit-learnに特化するならば、PyCon USのビデオは必聴です。scikit-learn開発の中心にいるJake Vanderplas氏とOlivier Grisel氏のチュートリアルセッションはとてもわかりやすく、彼らの提供するノートブックにはいろいろなティップスが詰め込まれていて、とても勉強になります。お勧めです。

http://pyvideo.org/video/1655/an-introduction-to-scikit-learn-machine-learning

http://pyvideo.org/video/1719/advanced-machine-learning-with-scikit-learn

機械学習上級

機械学習の上級コースとして、Courseraには機械学習業界で流行中の深層学習のコース“Neural Networks for Machine Learning”があります。

https://class.coursera.org/neuralnets-2012-001/class/index

さらにvideolectures.netには、全世界で行われたコンピュータサイエンス系学会のチュートリアルセッションが大量にアーカイブされています。有名なChristopher Bishop氏の「パターン認識と機械学習」の部分講義もあります[注16]し、博士号最終審査(Ph.D.ディフェンスといいます)！さえあります。このリソースを活用しない手はありません。このvideolectures.comについては、紙幅の都合上この程度で終わらせますが、機会があったらもう少し詳しく紹介してみたいです。

http://videolectures.net/

最後に

以上、MOOCによるデータサイエンスの学習リソースについて、駆け足で説明してきました。これらのデータサイエンスの知識や技術の基礎の上に、データ解析ならソーシャルネットワーク解析、地理空間データ分析、マーケティングデータ分析などの応用がありますし、機械学習なら自然言語処理、画像認識、自動翻訳、発音認識、乗り物の自動運転などの応用があるのです。

紙幅の関係上でまったくもって網羅的とは言いがたく、個人的なバイアスがある、抜けが多い記述になりました[注17]が、このあたりで筆をおくことにします。お読みくださってありがとうございました。SD

注16) http://videolectures.net/mlss04_bishop_gmvm/

注17) データサイエンスの大きな領域として、非正規でないデータを使えるように整形し(データクレンジングとかデータクリーニングといいます)、そのデータを処理するための分散コンピューティング環境を整え、大きなデータを遅延なく保存し、必要なときに遅延なく検索できる、というものがあります。こちらについてはまた機会をあらためて書ければうれしいです。

CHAPTER 2 1

何ができるのか知らなければはじまらない

ツールを使ってデータ分析をやってみよう [Excel編]

データ分析を本業にしていない人にとっては、いったい何を、どう分析するのか想像できないと思います。まずはどんなふうに分析するのか、身近なExcelを使ったデータ分析の手法を実際にやってみましょう。

高木 基成（たかぎ もとしげ）　サントリーシステムテクノロジー（株）

はじめに

「データ分析」と聞いて具体的な作業を想像できる読者の方はどの程度いらっしゃるのでしょうか。経験や知識が備わっているのであれば、この章に時間を割く必要はありません。しかし、筆者のように学生時代だけでなく、社会人になってからもデータ分析にかかわることなく過ごしてきた読者の方には、少しだけおもしろい情報をお届けできるかもしれません。少々お時間をいただき、一緒にデータ分析を見ていきましょう。

データ分析を取り巻く現状

ビッグデータ、データサイエンス、機械学習、統計解析などのさまざまな単語がIT業界を飛び交っています。大量に生成されているデータから価値のある情報を引き出して役立てたいという関心が生まれ、データ分析に注目が集まっています。また、数学や統計学などの専門知識を有したデータサイエンティストと呼ばれるデータ分析のスペシャリストにも注目が集まり、人材が不足している現状や育成に議論が行われているところです。

一般的な企業においては、基礎的・基本的なデータ分析のニーズも数多く眠っており、どのような体制で取り組むべきか手探りの状況と言えるでしょう。データサイエンティスト、ビジネスの現場担当者、情報システム部などのスタッフ部門の担当者の誰が分析すべきなのか、議論は尽きません。このような状況の中、ITエンジニアはどのようにデータ分析と向き合うべきでしょうか。

データ分析とは？

そもそも分析とは何でしょうか。国語辞典には、「所与の対象・表象・概念などを、それを構成する部分・要素・条件などに分け入って解明すること（大辞林）」と書いてあります。データ分析とは、現実で起きていることを解明するため、データから意味のある情報を取り出すこと、という側面があるでしょう。

データ分析の分類

データ分析は大きく2つに分類することができます。**発見型**と**検証型**です。前者は、明確な目的を定めずにデータを多角的に分析し、これまで知られていない新しい知見を見つけることを目的とします。斬新な情報を得られることから発見型には大きな注目が集まります。しかし、実際のデータ分析は検証型で行われることがほとんどです。検証型は、予想した因果関係を証明するためにデータを使って裏付けを取ることを目的とします。

何を分析するのか？

データ分析をビジネスに適用する場合、企業の目標に対してメリットのある情報を提供する

ことが求められます。具体的な分析ニーズを少し紹介します。

・売上／需給予測
・売上要因分析（何をすれば売れるのか？）
・顧客分析（どの顧客が何を買っているのか？）
・業務効率化

これらの例からもわかるとおり、データ分析を通してわかることは、①過去の事象を解析して現状を把握すること、②過去の事象を踏まえて将来を予測することの2点となります。分析を活用しなければ、経験と勘に頼った業務の運営になります。「新商品の顧客は10代が多いから、若者向けのキャンペーンを展開しよう」、「売上アップにはテレビCMが最適だから長時間露出するようにしよう」などの施策が客観的な情報に基づく意思決定を経ずに実施されます。ビジネスの現場において、経験と勘は大きな武器になることは間違いありませんが、担当者個人への依存度が高くなります。データ分析を活用することで見えない知見が形式化されることになり、結果として業務を安定して進められるようになります。

データ分析の流れ

データ分析は大雑把に言えば**表1**の流れで進めることになります。将来を予測したい場合であれば、分析フェーズ④のモデル作成を実施することになります。現状を把握したいだけであれば、モデル作成は必要ありません。

さて、**表1**の実施手順を見て、皆さんはどのフェーズが難しいと思いますか？　データ分析やモデル作成は専門スキルが要求されるので難しそうに感じるでしょうか。分析内容やプロジェクトの状況もさまざまですから、一概にどのフェーズが難しいとは言えないかもしれません。しかし、データ分析の経験者にインタビューすると、たいていの場合は「データ収集整理は、分析作業と比べると何倍も労力や時間がかかる作業である」という答えが返ってきます。たとえば、他社の売上と比較分析をしたいと考えたときに、他社の売上情報を簡単に入手することはできるでしょうか。入手を通して加工された情報に間違いはないのでしょうか。少し考えるだけでも問題点がたくさんありそうだと容易に想像できるでしょう。

ITエンジニアはどのように関わるべきか?

ここまでの内容を通して、データ分析のイメージを少しは具体的に持っていただけたでしょうか。さまざまな情報がシステムの中に格納されるようになり、ビッグデータ時代と呼ばれて活用が望まれていますが、現実には一般的な分析ですら実施している企業は少ないでしょう。データの山に宝が埋もれていることを知りながら、放置されているというもったいない状況です。もっとデータ分析を活用していくべきではないでしょうか。

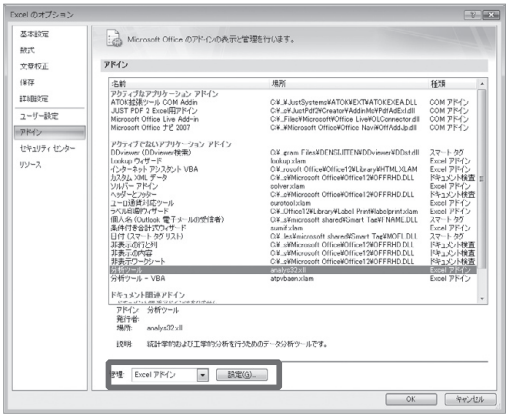
▼表1　データ分析の大まかな流れ

分析フェーズ	実施項目	実施内容
①	目標設定	・分析をして明らかにしたい内容を決める ・分析結果を踏まえて実施したいアクションを洗い出す
②	データ収集整理	・分析に必要となるデータを集める ・データ異常を取り除き、粒度をそろえる
③	データ分析	・分析手法を使いデータ分析を行う ・必要に応じて前のフェーズに戻る
④	モデル作成	・分析結果を踏まえた数理式を構築する ・モデルの精度など検証を行う
⑤	ビジネス適用	・分析結果や予測結果を踏まえて実施するアクションを決める ・ビジネスに起きた変化を測定し、さらなる分析につなげる

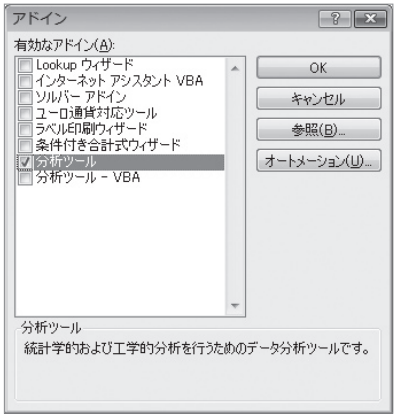
▼リンク1　分析ツールの追加手順サイト

Ver.	URL
2003	http://office.microsoft.com/ja-jp/excel-help/HP001127724.aspx
2007	http://office.microsoft.com/ja-jp/excel-help/HP010021569.aspx?CTT=1
2010	http://office.microsoft.com/ja-jp/excel-help/HP010342659.aspx?CTT=1
2013	http://office.microsoft.com/ja-jp/excel-help/HA102749007.aspx?CTT=1

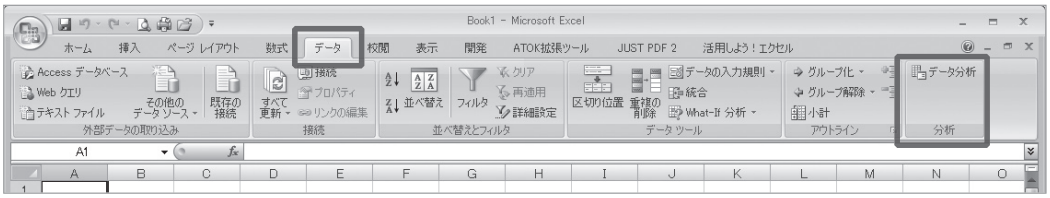
▼図1　Excelのオプション画面



▼図2　有効なアドインの選択画面



▼図3　リボンに追加された「データ分析」



では、冒頭でも書きましたが、ITエンジニアはどのようにデータ分析と向き合うべきでしょうか。筆者自身も明確な答えはありません。業務担当者やデータサイエンティストのような専門家に任せておけば良いという考え方もあるでしょう。しかし、積極的に取り組んでいくべきではないかと考えています。システムがデータを生成し、保管しているわけですから、システムの開発や運用をしているITエンジニアがデータ分析のスキルを身につけることで、データ活用が進むのではないでしょうか。

ここからは手を動かしていきましょう。はじめのステップとして、Microsoft Excelを活用していきます。ビッグデータに宝が埋もれている、と言ったものの、少しのデータを分析するだけでもさまざまな知見を得ることができます。ビッグデータ活用までの道のりは遠く、挑戦すべき課題はたくさんある……ということでしょう。戦場としては申し分ないですね。

今から作業するデータの整理とは、**表1**の「データ分析の大まかな流れ」に従えば、分析フェーズ②のデータ収集整理に関係する作業になります。

分析ツールを導入する

はじめに、Excelに分析ツールをアドオンとして追加しましょう。Excelのバージョンによっ

▼表2　レストラン「らうす」の売上情報

月日	気温(℃)	売上(千円)
6/1	12	1,800
6/2	9	1,080
6/3	15	2,340
6/4	21	3,465
6/5	24	3,816
6/6	20	3,120
6/7	11	1,617
6/8	12	1,836
6/9	24	3,758
6/10	16	2,520
6/11	9	1,040
6/12	10	1,440
6/13	15	2,489
6/14	15	4,500
6/15	16	2,489
6/16	20	3,000
6/17	19	2,850
6/18	14	2,100
6/19	22	3,498
6/20	14	2,489
6/21	10	1,446
6/22	12	1,602
6/23	11	1,452
6/24	12	1,638
6/25	11	1,132
6/26	13	472
6/27	19	2,844
6/28	24	3,686
6/29	24	3,154
6/30	24	3,722

て導入方法が若干異なります。基本的には、アドオン一覧から分析ツールを選択するだけで使えるようになります。Microsoftの公式サイトに追加手順がありますので、こちらを参考にして設定してください(**リンク1**)。

たとえばExcel 2007では、Officeボタンから[Excelのオプション]を選択して開いたウィンドウの左側にある[アドイン]を選びます(**図1**)。アドインリストの下にある「管理：」のドロップダウンリストから「Excelアドイン」を選択して[設定]のボタンを押すと、「有効なアドイン」を選ぶためのダイアログボックスが開きます(**図2**)。リストにある「分析ツール」のチェックボックスにチェックを入れ、[OK]ボタンを押せばアドインが開始されます。完了すると、メニューの[データ]のなかに、「データ分析」が組み込まれているはずです(**図3**)。

データを眺めよう

今回は、想像上の北海道にあるレストラン「らうす」の売上データを分析してみましょう(たまたま北海道で原稿を執筆したこと以外に、これらの想定情報に深い意味はありません)。分析フェーズ①の目標設定で、「らうす」のオーナーは仕入れや今後の拡大のため、売上の予測をしたいと考えました。レストランへ足を運んでくれるお客様の人数は、気温と関係があると考えて、気温と売上を使った分析をすることになりました。

表2のデータは、気温と売上の情報です。このデータに対して、**表3**にある情報を関数を使って算出してみましょう。結果は**表4**のようになります。これらの情報は「基本統計量」と呼ばれています。中心の位置やばらつき具合を示す指標のことでデータの分布を要約して、傾向や全体像をつかむことができます。

▼表3　基本統計量と関数

項目	Excel関数	何がわかるのか
最小	min(範囲)	―
最大	max(範囲)	―
合計	sum(範囲)	―
平均	average(範囲)	データすべてを足しあわせたときの中心の値がわかる。極端に大きな値や小さな値の影響を受ける
中央値	median(範囲)	データを並べたときの中心の値がわかる。極端に大きい／小さい値があるときには、中心の値としてはこちらのほうが正確と言える
最頻値	mode(範囲)	最も多く含まれている値を算出する。最頻値が複数ある場合には、異なる種類のデータが含まれている可能性がある
標準偏差	stdev(範囲)	各値と平均値の差を計算したもの。データのばらつき具合がわかる。小さい値の場合は同じようなデータが集まっており、大きい値の場合はデータが散らばっていると判断できる

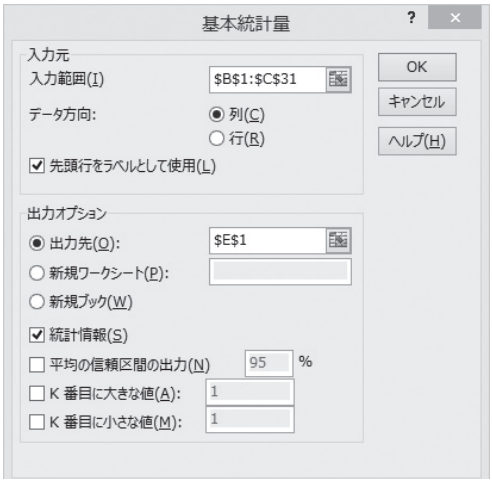
▼リンク2

サイト名	URL
科学の道具箱	http://rikanet2.jst.go.jp/contents/cp0530/start.html#

▼表4　基本統計量の算出結果

	気温(℃)	売上(千円)
min(範囲)	9	472
max(範囲)	24	4,500
sum(範囲)	478	72,394
average(範囲)	16	2,413
median(範囲)	15	2,489
mode(範囲)	24	2,489
stdev(範囲)	5	1,019

▼図4　基本統計量の設定



データを眺めてみるだけでも、いろいろな情報があることがわかっていただけたのではないでしょうか。たとえば、標準偏差と平均の値を見ると、ばらつきが大きそうに思いませんか。基本統計量は日々の業務や会話の中でも利用している身近なデータ分析情報です。

この基本統計量は、Excelの分析ツールからも算出することができます。メニュー[データ]から[分析ツール]を起動しましょう。どの機能を使うのか選択するダイアログが表示されるので「基本統計量」を選択してください。「入力範囲」には、気温と売上の列を選択し、「統計情報」にチェックを入れて「OK」ボタンを押しましょう(**図4**)。**表5**の結果が表示されることを確認してください。桁数の表現や小数点に少し違いがありますが、「セルの書式設定」を変更すれば、関数を入力して算出した**表4**の値と同じになります。分析ツールを使うと、分散・尖度・歪度などの基本統計量も自動的に算出されます。

散布図を作ろう

では次に、散布図を作ってデータのばらつきを可視化しましょう。気温と売上の値の部分を選択して、メニュー[挿入]から[グラフ]の「散布図」を選択して描きましょう(**図5**)。数値だけの情報を見るよりも、グッとわかりやすくなったのではないでしょうか。データ分析においては、「どう見せるか？」という点も非常に重要となります。データの可視化技術にも注目しておきたいところです。

Note　データ分析の参考サイト

今回のテストデータ以外にも、学習データが必要な方は「科学の道具箱」というサイトを訪問すると良いでしょう(**リンク2**)。「コンテンツ」→「データライブラリー」と選択することで、学習のためのデータをいくつか発見できるはずです。また、このサイト自身はデータ分析を学ぶためのコンテンツが多数収録されています。学習コンテンツとしては申し分ないので、ぜひ利用してみてください。

ヒストグラムを作ろう

ヒストグラムを作成することでデータの中心部分がよくわかるようになります。今回は売上のヒストグラムを作りましょう。アドオンした分析ツールを使うことになります。これまで同様にメニュー[データ]から[データ分析]を起動し、「ヒストグラム」を選択してください。

「入力範囲」に売上の値を選択します。また「グラフ作成」にチェックボックスを入れましょう。OKボタンを押下すると**表6**と**図6**が出力されると思います。

では、もう少しヒストグラムを整形していきましょう。**図6**を見てわかることは、データ区間が自動計算されているため、直感的にわかりにくいと言えるでしょう。そのため、**表6**の情報を参考に人間がわかりやすい単位に変換したいと思います。**表7**のように千単位で増えていくようなデータ区間の一覧を作成しましょう。

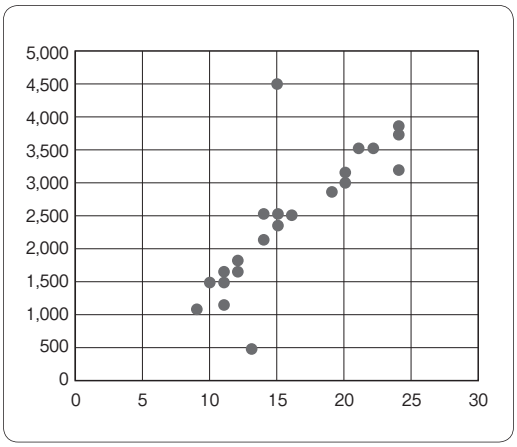
次にヒストグラムを再作成します。「入力範囲」の下にある「データ区間」に**表7**で作成した一覧を選択してください。後は新しく描画された表に対して、次の手順で作業して整形していきます。**表8**と**図7**にある状態になれば完成です。

1. タイトルを「売上の分布」にする
2. 右側と左側の「頻度」というラベルを削除する
3. 棒の部分を右クリックして書式設定から間隔を「0」にする
4. 「次の級」の部分を「5500」に書き換える
5. 縦横を調整して見やすい状態にする

▼表5　分析ツールから算出した基本統計量

	気温(℃)		売上(千円)
平均	15.93333333	平均	2413.136667
標準誤差	0.934564049	標準誤差	186.1031263
中央値（メジアン）	15	中央値（メジアン）	2488.75
最頻値（モード）	24	最頻値（モード）	2489
標準偏差	5.118818111	標準偏差	1019.328803
分散	26.20229885	分散	1039031.209
尖度	-1.246013172	尖度	-0.873000538
歪度	0.396655301	歪度	0.1082428
範囲	15	範囲	4028.1
最小	9	最小	471.9
最大	24	最大	4500
合計	478	合計	72394.1
標本数	30	標本数	30

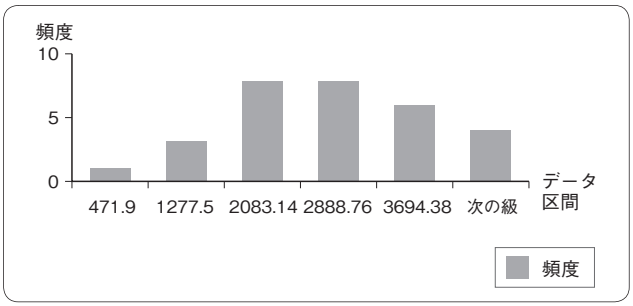
▼図5　気温と売上の散布図



▼表6　売上のヒストグラム（データ区間）

データ区間	頻度
471.9	1
1277.52	3
2083.14	8
2888.76	8
3694.38	6
次の級	4

▼図6　売上のヒストグラム（グラフ）



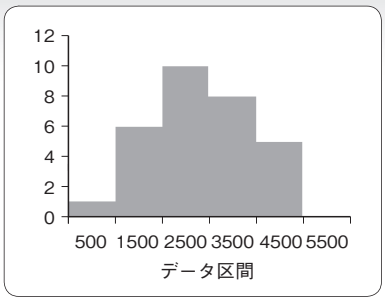
▼表7　整形したデータ区間

データ区間
500
1500
2500
3500
4500

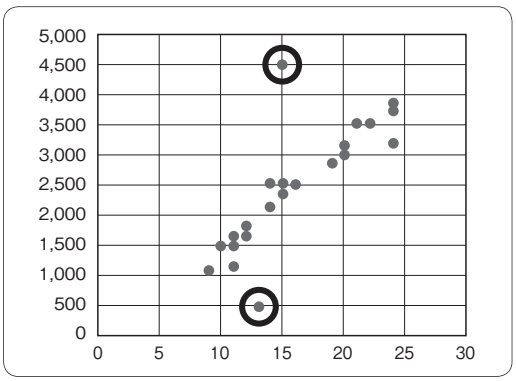
▼表8　加工後の売上のヒストグラム(データ区間)

データ区間	頻度
500	1
1500	6
2500	10
3500	8
4500	5
5500	0

▼図7　加工後の売上のヒストグラム(グラフ)



▼図8　外れ値



外れ値を取り除こう

散布図やヒストグラムを作成すると、データの一部に例外があることがわかるでしょう。サンプルを例にすると、売上が飛び抜けて高い日、低い日があります(**図8**)。これらの値が含まれていると、特殊なケースにもかかわらず、分析結果全体に影響を与える可能性があるので該当する行を削除することが一般的です。

今回は「外れ値」を取り除きましたが、ほかにも値を補正すべきケースがあります。たとえば、「欠損値」と言って値が格納されていない場合、どのように取り扱うのか決めなければなりません。平均値と仮定することが妥当なケースや、取り除くことが妥当なケースもあるでしょう。分析の内容や手法に応じて、イレギュラーな値を処理しなければなりません。

データを分析しよう(Excel編)

ここまでのデータの整理も立派な分析作業であったことがわかると思います。次は、皆さんがイメージされている数式を導き出すための作業になります。前章では、とりあえず手を動かしてもらいましたが、ここでは、データ分析の基礎となる統計解析について少し学んでから作業を進めたいと思います。

統計解析とは?

統計、統計学、統計解析などの用語が出てきていましたが、統計という言葉を何となく知っている……という方が多いのではないでしょうか。統計とは、集団(データの集まり)の属性を数量的に把握して、数値や図表で表現することになります。この統計を体系化した学問が統計学であり、統計学に基づく分析を「統計解析」と呼びます。

記述統計と推測統計

調査対象のすべてが用意されている場合、このすべて＝母集団(population)に対して分析をする行為は、記述統計学(descriptive statistics)と呼ばれる分野に属します。調査対象の一部しか用意されていない場合、この一部＝標本(Sample)を分析して、母集団を推測する行為は、推測統計学(inferential statistics)と呼ばれる分野に属します。現実を考えると、調査対象のすべてを把握していることは稀(まれ)ですから、ほぼ推測統計を行うことになります。数理統計学や多変量解析など説明を厳密にしていくと、違った関係性が見えてきますが、ここでは割愛させていただきます。

量的データと質的データ

データには、数量で表現される量的データと、分類(カテゴリ)で表現される質的データがあります。気温や体重は量的データで、天気や性別は質的データと言えます。これらのデータをもう少し細かくすると、**表9**のように分類できます。

これらの分類は、分析手法を選定するためには非常に重要なものとなります。統計学入門というページ(**リンク3**)でデータの種類に応じた分析手法の一覧が公開されていますので、全体像を把握されたい方はご覧いただければと思います。

相関

相関とは、2種類のデータが同じ傾向で変動しているかどうかを現象面から推測することを言います。同じ傾向の度合いを表す値として「相関係数」があります。同じ傾向があるということは、相互関連性があるという事実だけを表しており、一方が原因で他方の結果が表れているという因果関係を示しているわけではありません。たとえば、「気温」と「売上」は同じ傾向で変動していることが相関係数からわかった場合、何が原因なのかは人間が推測しなければなりません。

表10にあるように、2種類のデータの分類によっては相関を判断する相関分析の計算方法が異なるので注意が必要です。今回は量的データと量的データを分析するので、単相関係数を使うことになります。

この値はExcelの関数で簡単に求めることができます。外れ値を削除した気温と売上のデータに対して、CORREL関数を使います。第一引数に気温の値を指定し、第二引数に売上の値を指定します。結果は「0.9767782」となり、非常に強い相関があることがわかります。ここで注意しなければならないことは、前述のとおり相関があるからと言って、因果関係があるとは言えないことです。今回の例では、気温が上昇すれば売上が上がりますが、その理由は数値からは何もわからず、別の要素から人が考える必要があります。

回帰分析

気温と売上の関係について、相関分析を実施し、相関係数を算出すれば両者が関係していることはわかりますが、予測をすることができません。そこで1歩進めて、予測を行う分析が回帰分析になります。気温(x)が原因で、売上(y)がその結果である場合、xがyに与える影響を直線によって要約することになります。このときのxのことを「説明変数(explanatory variable)」または「独立変数」と呼び、yのことを「目的変数(criterion variable)」

▼リンク3　分析ツールの追加手順サイト

サイト名	URL
我楽多頓陳館	http://www.snap-tck.com/room04/c01/stat/stat02/stat0201.html

▼表9　データの分類

	分類	説明	例
量的データ	比例尺度(ratio scale)	値間に比例の関係があるため四則演算(加減乗除)ができる	金額、速度、時間
	間隔尺度(interval scale)	順序関係があり、連続しているため、加算、減算ができる	温度、西暦
質的データ	順序尺度(ordinal scale)	順序に大小関係がある	満足度、感情、態度
	名義尺度(nominal scale)	相当性が保証されている	血液、性別、出身

▼表10　データ分類の組み合わせに応じた手法

パターン	指標	取り得る値
量的データと量的データ	単相関係数	-1〜1
量的データと質的データ	相関比	0〜1
質的データと質的データ	クラメールの連関係数	0〜1

または「従属変数」と呼びます。今回は、Excelを使って回帰式を導き出すだけになりますが、実際の回帰分析は次の流れで実施することになります。

1. 目的変数と説明変数をプロットして散布図を作る
2. 回帰式を求める
3. 回帰式の精度を確認する
4. 母集団を推測する
5. 予測する

単回帰分析をする

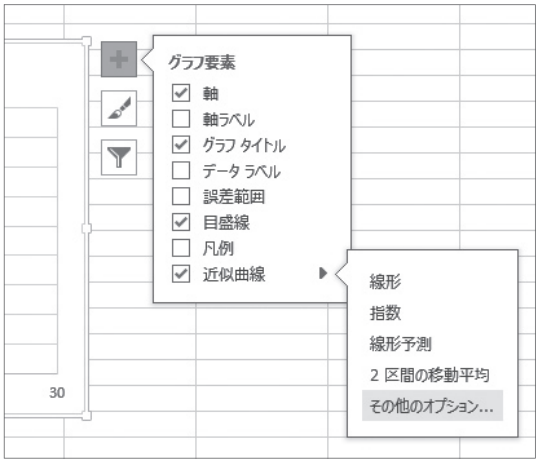
説明変数が1種類の回帰分析を単回帰分析と呼びます。この分析をExcelで実行してみましょう。

➡ 近似曲線を追加する

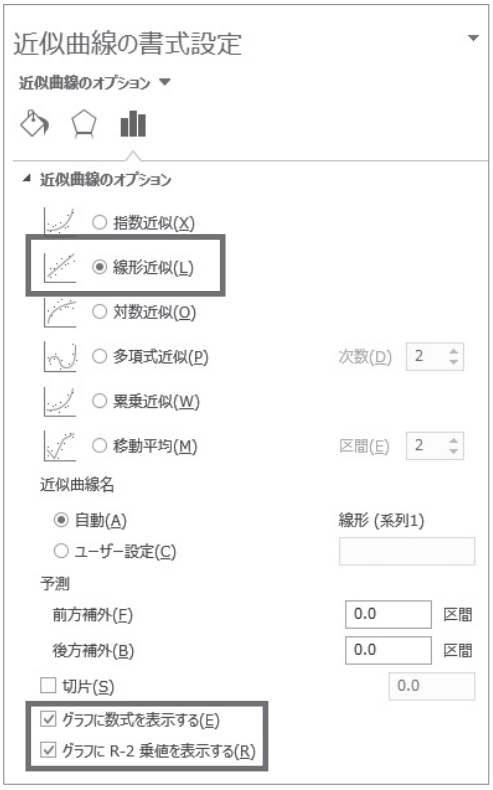
外れ値を削除したデータを使って散布図を再作成してください。この散布図に近似曲線を描きます。近似曲線とは回帰分析の結果として導き出される回帰式(y=ax+b)をプロットしたものになります。グラフを選択して、**リンク4**の手順に従い近似曲線を追加します。オプションとしては「線形近似」を選択して、グラフに数式とR-2乗値を表示するようにチェックをしておきましょう。

Excel 2013はインターフェースが大きく変わっているため具体的な手順を示すと、グラフを選択して「＋」のアイコンをクリックします。「近似曲線」をチェックして、「▶」を選択し、「その他のオプション…」を選択しましょう(**図9**)。右側に設定が表示されるのでグラフのアイコン

▼図9　線形近似の描画



▼図10　近似曲線のオプション



▼リンク4　近似曲線の追加

Ver.	URL
2003	http://office.microsoft.com/ja-jp/excel-help/HP005198462.aspx
2007	http://office.microsoft.com/ja-jp/excel-help/HP010007461.aspx?CTT=1#BMaddtrendline
2010	http://office.microsoft.com/ja-jp/excel-help/HP010342158.aspx
2013	http://office.microsoft.com/ja-jp/excel-help/HA102809798.aspx

をクリックし、「近似曲線の書式設定」に移動して、「線形近似」、「グラフに数式を表示する」、「グラフにR-2乗値を表示する」オプションを選択してください(**図10**)。

図11にある数式が回帰式となります。**気温(x)の値から、売上(y)を計算することができるようになりました。**予想される気温に167.35を乗算して、281.55を減算した値が売上の予測値になります。R-2乗値とは、簡単に言えば、回帰式の精度を表す値です。1に近づくほど精度が高いと言われているため、現在の値は非常に高く信頼できるものと言えるでしょう。

➡ 分析ツールを使う

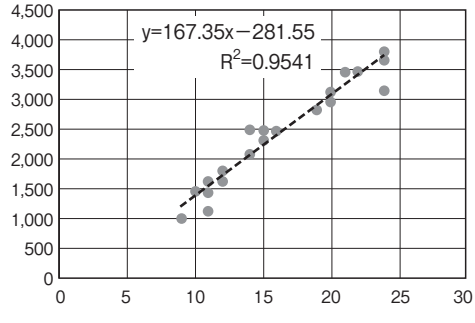
近似曲線とは別の方法で回帰式を算出してみましょう。メニューから分析ツールを起動して「回帰分析」を選択します。「入力Y範囲」に売上の値を選択し、「入力X範囲」に気温の値を選択します。OKボタンをクリックすると分析結果が表示されます。**図12**の背景色が変わっているところの値がグラフと一致していることがわかるでしょう。

なお、実業務の分析では、「補正R-2乗を使うべき」、「tやP-値を検討すべき」などの話がありますが、これらの点は紙面の都合上割愛させていただきます。

まとめ

本章ではデータ分析の基本的な知識と、Excelによるデータ分析の手法をごく一部ですが見ていただきました。続いて次の章では、さらに専門的なデータ分析ツールを使った手法を解説いたします。SD

▼図11　近似曲線と回帰式



▼図12　分析ツールによる回帰分析の結果

回帰統計

重相関 R	0.976778243
重決定 R2	0.954095735
補正 R2	0.952330187
標準誤差	197.1300769
観測数	28

分散分析表

	自由度	変動	分散	観測された分散比	有意 F
回帰	1	20999939.86	20999939.86	540.3961773	6.36544E-19
残差	26	1010366.948	38860.26722		
合計	27	22010306.8			

	係数	標準誤差	t	P-値	下限 95%	上限 95%	下限 95.0%	上限 95.0%
切片	-281.5523909	121.545944	-2.316427695	0.028679251	-531.393657	-31.71112486	-531.393657	-31.71112486
X 値 1	167.3477043	7.198858297	23.24642289	6.37E-19	152.5502392	182.1451695	152.5502392	182.1451695

CHAPTER 2

何ができるのか知らなければはじまらない

ツールを使ってデータ分析をやってみよう［R・Mahout編］

前章に続いてデータ分析を実際にやってみましょう。より専門的な分析ツールとして、本章では「R」と「Mahout」を使います。

高木 基成（たかぎ もとしげ）　サントリーシステムテクノロジー（株）

データを分析しよう（R編）

前章では、単回帰分析をExcelを使って作業しました。説明変数（前章参照）が2種類以上の場合の回帰分析を重回帰分析と呼びます。重回帰分析も、Excelで作業することができますが、今回は「R」を使って分析したいと思います。

Rとは?

R（R言語とも呼ぶ）は統計解析のOpen Source Softwareです。統計解析のプログラミング言語とその実行環境を含んでいます。教育現場や実務においても広く普及しており、最新の研究成果や便利な機能が拡張パッケージとして提供されています。R本体はCUI（簡易なGUIを含む）で操作することになるので、複雑な分析においてはプログラミングが必要となります。Rの高度なGUI環境としては、RCommanderやRStudioと呼ばれるものがありますが、今回はR単体で使います。

Rのインストール

まずはRのモジュールをダウンロードしましょう。公式サイトにミラーサーバへのリンクがあります。たとえば、兵庫教育大学のWebサイトからダウンロードできます（**リンク1**）。

インストール方法ですが、通常のソフトウェアと同じく、WindowsやMacの場合は数クリックで完了するので迷うことはないでしょう（**リンク2**）。

Rを動かしてみる

それでは、Rを起動して動かしてみましょう。Rのアイコンを起動するとウィンドウが起動します（**図1**。画面はWindows版）。Rは型を宣言する必要がありません。「`r_start<-"hello R!"`」と入力してリターンキーを押し、「`r_start`」と入力してリターンキーを押すと「`hello R!`」が表示されます。

▼リンク1　Rのモジュールダウンロードサイト

ホスト	URL
兵庫教育大学	http://essrc.hyogo-u.ac.jp/cran/

▼リンク2　インストール方法一覧

OS	URL
Windows	http://cran.r-project.org/doc/manuals/r-release/R-admin.html#Installing-R-under-Windows
Mac	http://cran.r-project.org/doc/manuals/r-release/R-admin.html#Installing-R-under-OS-X
Unix	http://cran.r-project.org/doc/manuals/r-release/R-admin.html#Installing-R-under-Unix_002dalikes

▼表1　レストラン「らうす」の売上情報（＋通行人）

月日	気温(℃)	通行人	売上(千円)
6/1	12	625	1,800
6/2	9	350	1,080
6/3	15	575	2,340
6/4	21	1,195	3,465
6/5	24	1,185	3,816
6/6	20	825	3,120
6/7	11	575	1,617
6/8	12	285	1,836
6/9	24	1,425	4,320
6/10	16	825	2,520
6/11	9	577	1,040
6/12	10	565	1,440
6/13	15	805	2,489
6/15	16	825	2,489
6/16	20	975	3,000
6/17	19	975	2,850
6/18	14	685	2,100
6/19	22	1,325	3,498
6/20	14	825	2,489
6/21	10	425	1,446
6/22	12	605	1,602
6/23	11	585	1,452
6/24	12	645	1,638
6/25	11	975	1,132
6/27	19	875	2,844
6/28	24	1,025	3,686
6/29	24	875	3,154
6/30	24	1,125	3,722

重回帰分析をする

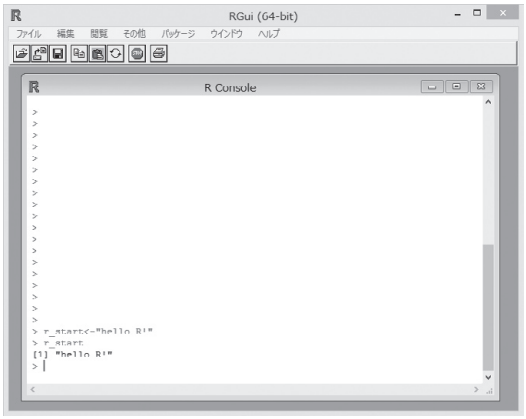
表1は、これまでの分析で使った「らうす」の売上データに、店舗前の通行人の数を加えた表になっています（前章で行った外れ値も削除済み）。このデータを使って分析をしましょう。

Rの画面で**図2**の内容を入力してください。これらのコマンドを実行した結果、**図3**と**図4**のような結果が得られたことを確認しましょう。**図3**は対散布図になり、各変数の相関関係が視覚的にわかります。右上に傾いていれば正の相関があり、右下に傾いていれば負の相関があります。**図4**の（Intercept）は切片と呼ばれ、座標軸との交点を表します。airとpasserは、気温と通行人の変数に対する傾き、偏回帰係数になります。これらの値を使って重回帰式を構築していきます。この新しい重回帰式は次のようになります。

y（売上）＝x1（気温）×162.1573＋x2（通行人）×0.1141－290.0759

なお、実際の分析作業では数十の変数を検討することになります。変数が多すぎると考慮すべき内容が増えることになるため、「説明変数ができるだけ少なく、精度が高い」重回帰式を構築することが重要となります。この最適な重回帰式を算出するための方法は次のようにいくつかあります。

▼図1　Rの実行画面



・変数増加法
・変数減少法
・変数増減法
・「情報量規準」に基づく方法
　など

Rのstep関数を使うと、「情報量規準」に基づく方法で変数を検討することができます。**図5**のコマンドを実行します。今回のテストデータでは、気温だけを使った単回帰式のほうが最適なモデルということになりました。

R言語を使うことで、簡単に重回帰分析の計算ができることがわかっていただけたでしょうか。なお、実業務のデータ分析はこんなに簡単ではありません。考慮すべき説明変数は数多くあり、どの変数を採用すべきか、回帰式の精度

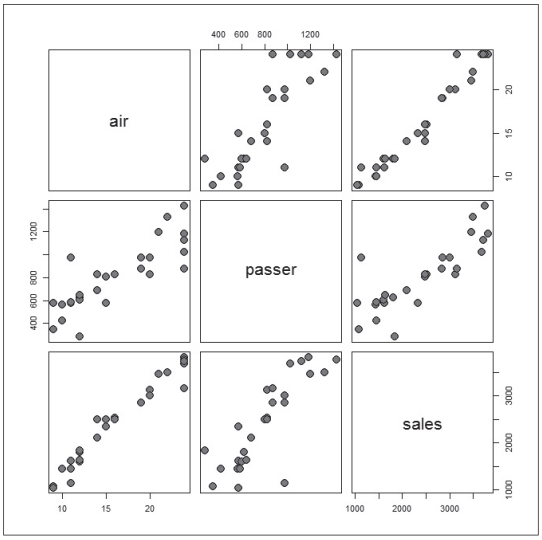
▼図2　Rのコマンド

```
❶気温の読み込み
air<-c(12,9,15,21,24,20,11,12,24,16,9,10,15,16,20,19,14,22,14,10,12,11,12,11,19,24,24,24)

❷通行人の読み込み
passer<-c(625,350,575,1195,1185,825,575,285,1425,825,577,565,805,825,975,975,685,1325,825,425,605,585,
645,975,875,1025,875,1125)

❸売上の読み込み
sales<-c(1800,1080,2340,3465,3816,3120,1617,1836,3758,2520,1040,1440,2489,2489,3000,2850,2100,3498,248
9,1446,1602,1452,1638,1132,2844,3686,3154,3722)

❹データセットを作成する
rausudata<-data.frame(air,passer,sales)

❺データセットの内容を表示する
rausudata

❻相関関数の行列を作成する
round(cor(rausudata),4)

❼対散布図により関係の全体像を見る
pairs(rausudata,pch=21,bg=" red" ,cex=2)

❽sales以外の情報を使って重回帰分析を行う
(rausudata.lm<-lm(sales~.,data=rausudata))
```

▼図4　Rコマンドの実行結果

```
Call:
lm(formula = sales ~ ., data = rausudata)

Coefficients:
(Intercept)          air          passer
   -290.0759     162.1573       0.1141
```

▼図3　対散布図



に問題はないのか、などの点を十分に検討していかなければなりません。

ビッグデータを処理しよう

ここまでの分析では、ビッグデータと言われるような大量データは処理しませんでした。ビッグデータを取り扱わなくても分析できることはたくさんあります。実業務の分析作業では、目標と照らしあわせて、どこまでのデータを分析すべきなのか、慎重に見極めなくてはなりません。しかし、スパム分類のように、大量のデータでなければ期待する結果が得られない場合もあるでしょう。ここでは、大量データの処理基盤としてMahoutを使ってみましょう。

機械学習とは

機械学習(Machine Learning)は、人工知能(Artificial Intelligence)の研究分野の一分野で、機械であるコンピュータに、人間と同じような学習能力を持たせることを目標としている研究分野です。たとえば、「複数の画像からリンゴが写っているものだけを抽出したい」場合、機械学習では大きく次の手順で処理を実装することになります。

1. リンゴを識別するためのサンプル画像を用意する
2. サンプル画像から特徴(色や形状など)を学習する。(抽出と管理)プログラムを開発する
3. 学習した情報を使って大量にある「リンゴが

▼図5　step関数の実行と実行結果

```
> rausudata.lm2<-step(rausudata.lm)

Start:  AIC=299.57
sales ~ air + passer

         Df Sum of Sq     RSS    AIC
- passer  1      8552 1009954 297.81
<none>                1001403 299.57
- air     1   5889519 6890922 351.58

Step:  AIC=297.81
sales ~ air

       Df Sum of Sq      RSS    AIC
<none>               1009954 297.81
- air   1  20995991 22005945 382.09
```

写っている画像」を判別していく

課題を直接解決する処理を作るのではなく、“解決するための方法を学習する”処理を開発することになります。この学習や判別は、統計を活用することになるため、統計的機械学習とも呼ばれており、統計と機械学習の境界は曖昧（あいまい）になっていると言えるでしょう。

機械学習を適用すべき領域には、一般的な統計解析に比べて、①学習のためのデータが多いほど精度が良くなる（望ましい結果が得られる）、②データが多いほど計算量が増加する（処理に時間がかかる）という特性が、より強く働いているため、ビッグデータを扱う処理を機械学習と呼ぶ風潮もあります。

2種類の学習

機械学習の問題は大きく分けて、**教師あり学習**（supervised learning）と、**教師なし学習**（unsupervised learning）に分類されます。両者の中間の問題を対象にした「半教師付き学習」や「強化学習」など種類もありますが、基本として、この2つを理解しておく必要があります。今回挑戦する分類は教師あり学習となります。

➡ 教師あり学習

教師あり学習では、学習データ（あるいは教師データや訓練データとも）と呼ばれる入出力ペアのサンプルデータを複数使って学習を行います。学習した結果を使うことで、入力データが与えられたときに出力結果を正しく予測することができるようになります。正しい答えを知っている「教師」に正解データを与えてもらって処理を行うため、教師あり学習と呼ばれています。

➡ 教師なし学習

教師なし学習のほうは入力データだけが与えられます。教師あり学習と違い、判断するための正解データが与えられないので適用領域も自ずと限られてきます。正解が与えられないため、何らかの判定基準を設けて処理しなければなりません。教師あり学習のほうが使い勝手が良いように思われるかもしれませんが、大量の正解データを用意するのはコストも高く、近年では教師なし学習も盛んに研究されているようです。

Mahoutとは

ご存じの方も多いとは思いますが、Mahoutとは何かを簡単に紹介しておきましょう。Mahoutは機械学習を行うためのライブラリです。Apache Software Foundationで開発されているOpen Source Softwareです（**リンク3**）。

図6はMahoutのアーキテクチャをまとめたもので、個別のアルゴリズムを大きく5個に分類しています。活用例を紹介すると、お勧め商品の提案（Recommenders）、ニュースのグループ分け（Clustering）、メーラのスパムの分類（Classification）などの処理で使うことができます。頻出するアイテムの組み合わせ抽出（Frequent pattern mining）を使えば、POSデータから“ビールとおむつを一緒に購入する消費者が多い”ことなどの知見を得ることができます。宅急便のトラック配車計画の最適化（Genetic）に利用できる可能性もあります。

Mahoutは40個を超える機械学習のアルゴリズムを内包しており、大量のデータを分散環境で処理するための基盤であるApache Hadoop（**リンク3**）に対応しているため、近年注目を集め

ています。

環境構築

では、Mahoutを動かすための環境を作っていきましょう。次の手順で作業を進めることになります。MacやLinuxの環境であれば、すぐに環境を構築することができますが、Windowsの場合はcygwinをインストールして、シェルの環境が使えるようにする必要があります。Windowsの環境は一癖ありますので、筆者はAmazon EC2でCentOSを動かして環境を作っています。

1. Java 6をインストールする
   - java -versionで情報が表示されるよう実行パスに追加する
   - yum seach jdkでOpenJDKを探してインストールしても良い
2. Apache Mavenをインストールする
   - mvn -versionで情報が表示されるよう実行パスに追加する
3. Apache Mahoutをインストールする

ここでは、3番目のMahoutのインストールから説明していきます。原稿執筆時点のMahoutの最新バージョンは0.7になります。

リンク4のダウンロードページからOfficial Releaseの「mahout-distribution-0.7-src.tar.bz2」をダウンロードします。モジュールがダウンロードできたら、**図7**の手順で作業していきましょう。

Mahoutで分類する理由

前述したように、MahoutはHadoopに対応している点が特徴です。分類は教師ありの機械学習になるので、たくさんの学習用データを用意

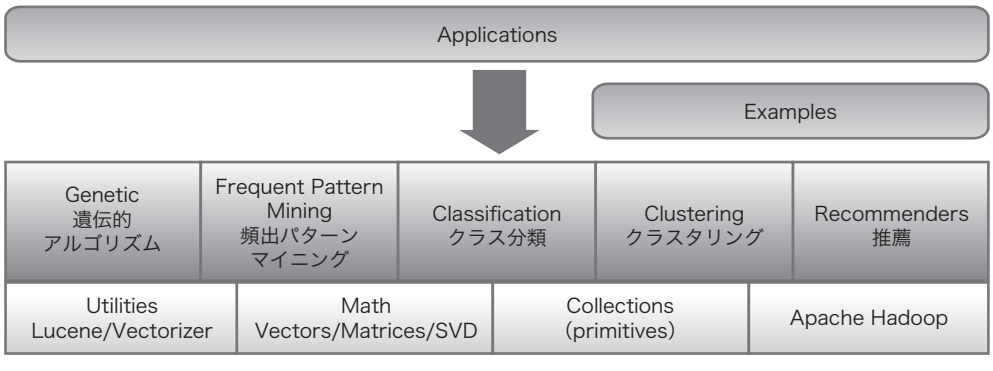
▼リンク3

サイト名	URL
Java SE Downloads	http://www.oracle.com/technetwork/java/javase/downloads/index.html
Apache Maven	http://maven.apache.org/
Apache Mahout	http://mahout.apache.org/
Apache Hadoop	http://hadoop.apache.org/

▼リンク4　Mahoutダウンロードサイト

サイト名	URL
Apache Mahout	https://cwiki.apache.org/confluence/display/MAHOUT/Downloads

▼図6　Mahoutのアーキテクチャ



※参考：http://www.slideshare.net/hamadakoichi/mahout-recommendation-mahout-jp-tokyowebmining-14th
https://cwiki.apache.org/confluence/display/MAHOUT/Algorithms

して作業を進めることになります。学習データが100万件を超えるような場合、学習処理だけで多くの時間が必要となります。Hadoop上で動作することで、この時間を短くすることが可能です。分類処理を実際に実行するときの時間が短縮できることもHadoopに対応していることのメリットでしょう。

また、今回のサンプルのように、MahoutはHadoopがなくても動作します。Mahoutは、Javaで開発されており、メンテナンス性やAPIの使い勝手の面からも採用するメリットがあると思われます。

Mahoutの分類アルゴリズム

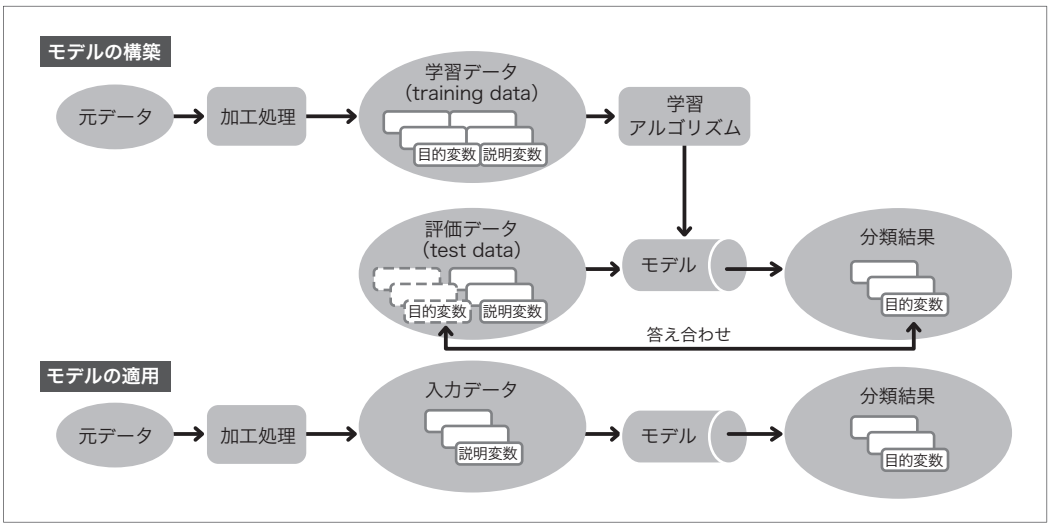
Mahoutには次のアルゴリズムが含まれており、分類に利用できます。今回は単純ベイズ法を使います。Mahoutにおける単純ベイズ法の実装は、テキスト形式のデータにのみ対応しています。大雑把に言えば、100万件以上の大規模のデータを効率よく処理できます。

1. 確率的勾配降下法（Stochastic Gradient Descent）
   OnlineLogistic、Regression、AdaptiveLogistic Regression、CrossFoldLearner
2. サポートベクターマシン（Support Vector Machine）
3. 単純ベイズ法（Naive Bayes）
4. 補完型単純ベイズ法（Complementary baive Bayes）

▼図7　Mahoutのインストール

```
❶モジュールを解凍する
$ tar jxvf mahout-distribution-0.7-src.tar.bz2

❷解凍したフォルダの中に移動する
$ cd mahout-distribution-0.7

❸Mavenのコマンドでインストールする
  非常に時間がかかりますが、mvn install -Dmaven.test.skip=trueのように
  テストを後回しにしてインストールすると早く終わります（おすすめはしません）
$ mvn install

❹mahoutコマンドを実行する
$ bin/mahout
```

▼図8　分類作業の流れ



▼表2　学習用学習用コンテンツ一覧

ディレクトリ		ファイル(.txt)	URL	内容
wiki	math	mathematics	http://en.wikipedia.org/wiki/Mathematics	数学
		probability	http://en.wikipedia.org/wiki/Probability_theory	確率論
		statistics	http://en.wikipedia.org/wiki/Statistics	統計学
		vector	http://en.wikipedia.org/wiki/Vector_calculus	ベクトル解析
		algebra	http://en.wikipedia.org/wiki/Algebra	代数学
		geometry	http://en.wikipedia.org/wiki/Geometry	幾何学
		analysis	http://en.wikipedia.org/wiki/Mathematical_analysis	解析学
		set	http://en.wikipedia.org/wiki/Set_theory	集合論
		info	http://en.wikipedia.org/wiki/Information_engineering	情報工学
		calculus	http://en.wikipedia.org/wiki/Calculus	微積分学
	jura	jurisprudence	http://en.wikipedia.org/wiki/Jurisprudence	法学
		judiciary	http://en.wikipedia.org/wiki/Judiciary	司法
		legislation	http://en.wikipedia.org/wiki/Legislation	立法
		public	http://en.wikipedia.org/wiki/Public_law	公法
		civil	http://en.wikipedia.org/wiki/Civil_law_(area)	民法
		criminal	http://en.wikipedia.org/wiki/Criminal_law	刑法
		international	http://en.wikipedia.org/wiki/International_law	国際法
		philosophy	http://en.wikipedia.org/wiki/Philosophy_of_law	法哲学
		history	http://en.wikipedia.org/wiki/Legal_history	法史学
		comparative	http://en.wikipedia.org/wiki/Comparative_law	比較法学
	medi	medicine	http://en.wikipedia.org/wiki/Medicine	医学
		wmedicine	http://en.wikipedia.org/wiki/Western_medicine	西洋医学
		tcmedicine	http://en.wikipedia.org/wiki/Traditional_Chinese_medicine	東洋医学
		cardiology	http://en.wikipedia.org/wiki/Cardiology	循環器学
		gastroenterology	http://en.wikipedia.org/wiki/Gastroenterology	消化器学
		pulmonology	http://en.wikipedia.org/wiki/Pulmonology	呼吸器学
		nephrology	http://en.wikipedia.org/wiki/Nephrology	腎臓学
		endocrinology	http://en.wikipedia.org/wiki/Endocrinology	内分泌学
		hematology	http://en.wikipedia.org/wiki/Hematology	血液学
		neurology	http://en.wikipedia.org/wiki/Neurology	神経学

5. ランダムフォレスト(Random forests)

※上記1～5は数字が大きいほうが学習コストが高くなる

作業の流れ

分類を実現するためには、前ページ**図8**にある作業を進めることになります。まず、学習と学習した結果を評価するためのデータを用意します。このデータを目的変数(分類で判断したい結果)と説明変数に加工して、学習用と評価用にデータを分けます。学習データでモデルを構築して、評価データを使ってモデルの正解率を評価します。評価データには正解となる目的変数が含まれているので、分類結果で推定された目的変数と答え合わせができます。

分類しよう

今回は学習データとしてWikipediaを使ってみましょう。日本語の場合、加工処理でMeCab[注1]などを使って形態素解析を行う必要

注1) 京都大学情報学研究科と日本電信電話(株)コミュニケーション科学基礎研究所の共同研究ユニットプロジェクトを通じて開発された、オープンソースの形態素解析エンジン。

▼図9　分類処理の実行手順

```
❶テストデータをシーケンスファイルに変換する
  （Keyがディレクトリ、Valueが単語の配列のデータが生成される）
bin/mahout seqdirectory -i wiki -o wiki-seq
bin/mahout seq2sparse -i wiki-seq -o wiki-vectors -lnorm -nv -wt tfidf

❷学習用のデータと、評価用のデータに分ける
  （randomSelectionPctで指定して20%をテスト用に分割している）
bin/mahout split -i wiki-vectors/tfidf-vectors --trainingOutput wiki-train-vectors --testOutput
wiki-test-vectors --randomSelectionPct 20 --overwrite --sequenceFiles -xm sequential

❸学習処理を行う
bin/mahout trainnb -i wiki-train-vectors -el -o wiki-model -li labelindex -ow

❹学習結果を使って評価を行う
bin/mahout testnb -i wiki-test-vectors -m wiki-model -l labelindex -ow -o wiki-testing/
```

がありますが、ここでは簡素化のため英語のコンテンツを使うことにします。

表2のURLからブラウザに表示されているテキストすべてを選択してコピーし、ファイル名と保存先のディレクトリをあわせてテストデータを作成してください。作成が終わったら、Mahoutのインストールディレクトリ直下にコピーしましょう。たとえば、/home/mahout/mahout-distribution-0.7にMahout本体がインストールされていれば、/home/mahout/mahout-distribution-0.7/wiki/にコンテンツをコピーします。

では、**図9**のコマンドを実行して、分類の処理を実行していきましょう。**図9**の最後のコマンド❹を実行することで、**図10**のように分類の結果が表示されたと思います。この実行結果では、約50%に相当する18ファイルが評価用に使われました。正しく分類できたファイルは16ファイルで、約90%と非常に高い正解率となっています。間違った2ファイルは医学に分類されるべきものですが、数学に分類されています。医学や数学が法律に分類されるような大きなミスマッチは起きていないように思えます。

実際の開発ではより複雑な問題を解くことになるので、ここまでの正解率を出すためには工夫が必要でしょう。今回は単純ベイズ法を採用してWikipediaのわずかなデータを分析してみました。ぜひ、アルゴリズムや学習データを変えてみたり、Hadoop上で動かしてみたりと、さまざまな観点からMahoutを学習していただければと思います。

▼図10　分類の結果

```
Summary
-------------------------------------------------
Correctly Classified Instances   : 16  88.8889%
Incorrectly Classified Instances : 2   11.1111%
Total Classified Instances       : 18

=================================================
Confusion Matrix
-------------------------------------------------
a      b      c      <--Classified as
4      0      0       |  4      a      = jura
0      7      0       |  7      b      = math
0      2      5       |  7      c      = medi
```

まとめ

本章では、Rを使った重回帰分析とMahoutによるビッグデータ処理を実際に動かしてみました。皆さんに、データ分析で使われるソフトウェアの操作イメージを持っていただければ、今回の目標は達成できたかな、と思っています。今回の簡単なサンプルをキッカケにして、Excel、R、Mahoutを使った分析にいろいろと挑戦してみてはいかがでしょうか。SD

CHAPTER 3

数式を使わなくても学べる!?

機械学習を学習するための手引き

本章では、社内の勉強会を通じて機械学習を学び、ついには学習書まで発行してしまったサイボウズ・ラボの社員自ら、データサイエンティストに一歩近づくための心構えや学習方法を紹介します。

竹迫 良範(たけさこ よしのり)　サイボウズ・ラボ株式会社
Mail takesako@shibuya.pm.org

データサイエンティスト

ビックデータの流行によって、企業内に蓄積された巨大なデータの中から経営に有益な情報を客観的に分析して、必要な施策を打ち、収益に結び付ける「データサイエンティスト」の職業が注目されるようになりました。

『ハーバード・ビジネス・レビュー』[注1]によると、米国では「データサイエンティストはスーパースターであり、21世紀で最もセクシーな職業」と呼ばれるにいたっています。日本ではまだデータサイエンティストの専門職名は一部の企業のみで使われる程度でそこまで普及していませんが、なぜここまで注目される職業になったのでしょうか。

ユーザ人口と時間の飽和

このようなデータ分析の職業が注目されるようになった理由の1つは「国内市場が飽和した」という状況が発生しているためです。日本の人口は約1億2千万人で、そのうちのインターネット利用人口は平成23年末の統計で人口比率約79%の9,610万人[注2]と推定されています。

この中で数千万人のユーザ数を獲得したインターネットサービスの成長は、どの程度見込めるでしょうか。国内総人口は急に増えることもなく、国内のユーザ数の伸びは頭打ちになることが予想されます。それを打開するには、海外市場に進出して日本人以外もマーケットの対象に含めるか、あるいは、1人で複数のアカウントを使い分けるようにして見かけ上のユーザ数を増やす施策が必要となります。携帯電話の普及率は人口比率100%を超えていますが、これは1人でガラケーとスマートフォンを2台持ちしたり、個人の私用と会社で支給される業務用の端末がそれぞれ別にある状況が発生しているためです。メールのアカウントも1人で複数持つのが当たり前になってきました。

昔のファミコンのようなコンソールゲームは自宅のTVに接続して遊ぶタイプのゲームでしたので、在宅中の時間しかプレイできませんでした。ガラケーやスマートフォンで遊ぶことのできる最近のソーシャルゲームは、自宅以外の場所でも、通勤移動途中や、学校や会社の休憩時間を利用してプレイできるようになりました。1人あたりのプレイ時間の機会を増やし、かつ自宅以外でゲームをプレイするユーザを新規獲得して市場の拡大に成功しています。

しかし、最近海外進出する日本企業が増えてわかってきたことですが、世界総人口も無限ではなく約71億人と有限で、人間に与えられた

注1）Thomas H.Davenport and D.J.Patil, "Data Scientist:The Sexiest Job of the 21st Century," Harvard Business Review,Oct.,2012.
注2）【出典】総務省「平成23年通信利用動向調査」 http://www.soumu.go.jp/johotsusintokei/statistics/statistics05.html

時間は平等で1人当たり24時間しかない、という事実があります。このようにユーザ人口と時間が飽和している状況の中で、企業が収益を上げるには、顧客一人一人がもっとお金をたくさん使ってもらう状況を増やす施策が必要となります。

アップセル

マーケティング用語の1つに「アップセル」という言葉があります。購入希望者に対して、上位モデルを提示することによって、より高価な商品を購入してもらい、1人あたりの売上高を増やす施策です。パソコンを購入するときに、どうせ買うなら、できるだけ高速なインテルの最新CPUモデルを選んでもらうのがアップセルに該当するでしょう。

クロスセル

もう1つ「クロスセル」というマーケテイング用語があります。これは既存顧客に別の商品の購入を勧めて、1人あたりの売上高を増やす施策です。ファストフード店でハンバーガーを購入するときに「一緒にポテトとジュースはいかがですか」と勧められるのがクロスセルに該当します。また、Amazonのオンラインショップなどで、ショッピングカートに商品を追加したときに「この商品を買っている人はこの商品も購入しています」とお勧めの別の商品がレコメンドされるのも有名なクロスセルの施策となります。

このようなお勧めの商品を表示するためには、過去の売り上げデータを解析して、関連度の高い商品と商品の組み合わせを分析して候補を提示してあげる必要があります。

今までは現場の店頭での売れ行きを把握している店員が直感で商品の配置の並び替えを行って店内のレイアウトを変えたりしてクロスセルの向上を行っていましたが、今はレジの裏に蓄積されるPOSデータを分析することで、人間の経験や直観に依存しない意外な関係性を発見できるようになりました。POSデータの分析で「紙おむつを購入する中年男性は同時に缶ビールを買う率が高い」という傾向が分析された逸話は有名で都市伝説となっています。

しかし、商品と商品の関連性を分析するには、すべての商品の組み合わせについて頻度分析を行う必要があり、商品が1万点であれば2つの商品の組み合わせの数は約5千万通りのパターンになり、商品の数が10倍に増えると、その組み合わせの数は100倍になります。

データの数が増えれば増えるほど、今まで人間の直感では把握できなかった意外な関係性が新しく発見できるかもしれないのですが、現実的には計算量が膨大となり、通常のアルゴリズムでは1台のコンピュータで処理できない規模に達してしまいます。HadoopやMahoutでは現実的な時間内で計算させるために、分散処理を行うことによってその問題を解決しようとしています。

KPIという1つの指標

マーケティングの現場では、KPI(Key Performance Indicator)と呼ばれる重要業績評価指標を設定することがあります。アクセス数やユニークユーザ数、ARPU(加入者1人あたりの月間売上高)などがKPIの指標としてよく定められ、定期的に数値目標を達成できているかどうかを確認されます。データ分析の現場では、このKPIの目標を達成できたかどうかが第一の評価の対象になります。

ここで大事なのがKPIよりも上位の概念です。KGI(Key Goal Indicator)と呼ばれる重要目標達成指標がKPIの上位に存在し、これは最終的な目標を定めたものです。KGIの目標を達成するために、具体的に実現可能なレベルの手段としてブレークダウンしたKPIの中間目標を設定しているという観点を忘れないでほしいということです。短期的にKPIの目標を達成できたとしても、長期的に安定した収益を上げる

ことができなければ、企業活動の健全な運営ができなくなってしまいます。

データサイエンティストの心構えとして、KPIよりも上位の概念があることを忘れずにビジネスにコミットしていく姿勢が大切だと言えます。

データ分析の基礎「統計」を学ぶ

さて、突然上司から「明日からデータサイエンティストの仕事をしてくれ」と言われたら、あなたはどうしますか。いきなりビックデータの解析をしようと思ってもハードルが高いと思いますので、まずは手元のスモールデータをきちんと分析できる素養を身につけておく必要があります。

そのためには高校数学の科目で勉強した（はずの）「確率・統計」の最低限の知識が必要となるのですが、入学年度によって数学III、数学C、数学Bなど確率・統計の科目名も異なり「検定」まで勉強しているかは世代によって異なります。また、高校数学は義務教育の範囲外ですので、すべての人が勉強しているとは限りません。

数式を使わないで統計を学ぶ本

数式アレルギーの人もいると思いますので、まずは数式を使わないで統計の基礎知識を学ぶ方法でお勧めの書籍を1つ紹介します。

『マンガでわかるナースの統計学 ―データの見方から説得力ある発表資料の作成まで』

田久浩志、
小島隆矢（著）、
こやまけいこ（作画）、
ビーコム（制作）、
オーム社、
2006年5月、
2,310円（税込）
ISBN＝
978-4274066498

タイトルや表紙で敬遠する人がいるかもしれませんが、「カイ2乗検定」の意味を理解していない人はまず最初にこの本を読みましょう。

データ分析を行う前にデータの収集を行う必要がありますが、アンケート項目の作り方から丁寧に解説されている良書です。名義尺度（男女の区別など）、順序尺度（良い悪いなど）、連続尺度（身長、体重など）の違いがきちんと解説され、それぞれの統計量の扱い方の注意点についても触れられています。タイトルのとおり看護師向けに書かれている統計学の学習本で、難しい数式はいっさい出てこず、単純な足し算と引き算と掛け算と割り算と、ルートの計算さえ電卓でできれば理解できる内容となっています。

➡ データ分析を始める前の心構え

本書は、とくにデータ分析を始める前の心構えが最初にしっかり書かれていることに好感が持てます。

① 問題は何か、何をしたいのか（評価項目の抽出）
② なぜそれをすると良いのか（ラダーアップ）
③ 客観的に証明するにはどうすれば良いか（ラダーダウン）

これはビッグデータで何かを解析しようとするときにも通用する普遍的な考え方です。

病院の臨床現場では、エビデンス（臨床結果）に基づく医療が推進されており、過去の臨床データから最適な治療を応用できるように統計学が積極的に利用されている分野でもあります。バイアスのかからないデータの収集方法、集めたデータからグラフ／表を作る方法、散布図の描き方、相関の見つけ方と検定の手法について一通り学べる内容となっています。

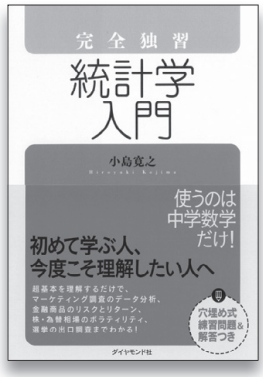
ちなみにナイチンゲールは「兵舎病院での死亡率の改善のためには衛生環境を向上させる必要がある」ことを実績とデータで示した統計学者でもあり、グラフを多用したレポートを作成し病院の縦割り行政を改革した功績があります。

確率を使わないで統計を学ぶ本

確率を使わないで統計学を学べる珍しい入門書として、帝京大学経済学部の小島寛之先生が書かれた本があります。

『完全独習 統計学入門』

小島寛之(著)、
ダイヤモンド社、
2006年9月、
1,890円(税込)、
A5判、
208ページ、
ISBN＝
978-4-478-82009-4



統計学には大きく分けて、古典的な記述統計の分野と、推測統計の分野の2つがあります。

- **記述統計**
  全体の特徴を把握する
  (人口調査、土地調査、国勢調査など)
- **推測統計**
  全体の母集団から一部の標本を無作為にサンプリングし、全体を推測する
  (選挙の当選確実など)

推測統計が現在のデータ分析で使われる基礎の統計学で、確率論と組み合わせて発展してきました。統計は「過去に起きたことに関する記述」でゆるぎないものとしてわかりやすいものですが、確率は「これから起きることに関する未来の記述」です。統計学を学び始めたときに感じる違和感が、これらの統計と確率の2つを一緒に扱って議論を進めてしまうところです。

この本の画期的なところは、確率をできるだけ使わずに推測統計の説明を試みているところです。確率変数$P(X=x_1)$や、微分積分、Σの記号も登場しませんので、数学記号の扱いに自信のない人は、統計学の復習のつもりで一通り読んでみるのが良いでしょう。

標準偏差の説明に多くのページが割かれていますので、現場の統計値を扱う際の数字の感覚を身につけることができます。

大数の法則と中心極限定理

金融工学も統計学が応用されている分野で、証券投資のポートフォリオ構築やオプション価格決定の理論を理解する際、大数の法則と中心極限定理は必須の概念になります。

- **大数の法則**
  サンプル数が十分に大きければ、観察された標本平均を母集団の平均とみなしてよい
- **中心極限定理**
  母集団の分布に関係なくサンプル数が十分に大きければ、標本平均の確率分布は正規分布に収束する

大数の法則と中心極限定理の厳密な証明を試みるためには、コルモゴロフに始まる近代確率論の教科書を勉強する必要があります。

『はじめての確率論 測度から確率へ』

佐藤坦(著)、
共立出版、
1994年2月、
3,360円(税込)、
A5判、216ページ、
ISBN＝
978-4-320-01473-2

『はじめての確率論 測度から確率へ』の本は、大学の数学を過去に1年程度学んだ経験があって、リーマン積分とルベーグ積分の違いを知っている人であれば読み進められる本です。最初の7ページまでは数学未経験者でも読み進めることができますが、それ以上は集合論の基礎や実数の連続性などについて勉強しておかないと難しいでしょう。シミュレーションによる試行結果で何となくそうなることはわかったけど、大数の法則と中心極限定理の理論的な裏付けが欲しいと思った人は、ぜひ手に取ってみましょう。

機械学習のモデルを適用する場合、データの確率分布を正規分布と仮定して議論を進めたりしますので、中心極限定理に対する理解は必須と言えます。

ベイズの推定で未来を予測

メールのSPAMフィルタの1つとして、ベイジアンフィルタがあります。これは過去に観測されたSPAMメールの単語の特徴をあらかじめ学習して分類器を作り、新しく届いたメールがSPAMメールであるかどうかの確率を分類器で計算して推測する手法です。

➡ ベイズの定理

約250年前にアマチュア数学家の牧師であったトーマス・ベイズが発見した「ベイズの定理」があります。

$$P(B|A) = \frac{P(A|B) \times P(B)}{P(A)}$$

定　　数：P(A)=事象Aが起きる確率

事前確率：P(B)=事象Aが起きる前の事象Bの確率分布

事後確率：P(B|A)=事象Aが起きた後の事象Bの確率分布(条件付き確率)

尤度(ゆうど)関数：P(A|B)=観察結果からみて前提条件が何であったかを推測する尤(もっと)もらしさ

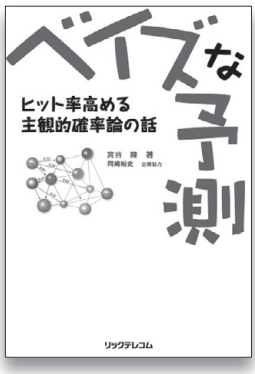
事後確率とは、ある事象が観測されたことを契機に、事前確率を高精度に修正したものです。

最尤(さいゆう)推定は、尤度関数を最大化する仮説を選択する推定法ですが、ベイズ推定は事後確率を最大化する仮説を選択する推定法です。

具体的なベイズの推定の応用実例を把握したい場合にお勧めできる本として『ベイズな予測』があります。数式はほとんど出てこないので、読み物として軽く読み進められる本です。

『ベイズな予測
―ヒット率高める主観的確率論の話』

宮谷隆(著)、岡嶋裕史(企画協力)、リックテレコム、2009年3月、2,205円(税込)、A5判、192ページ、ISBN＝978-4-89797-816-1



具体的なベイズ推定の適用例として、携帯電話の他社乗り換え要因の分析や、犯罪捜査、沈没船の捜索などの事例が紹介されています。

音声認識で使われる隠れマルコフモデル(HMM)や、ナイーブベイズによるマーケットの予測にも応用されています。

パターン認識と機械学習

筆者の所属するサイボウズ・ラボ株式会社では「自然言語処理に必要そうな機械学習の基礎知識を身につける」という目標のもと、2011年からビショップ先生の書かれた日本語訳の上下本『パターン認識と機械学習』を教科書として輪

読する社内読書会を中谷秀洋氏(@shuyo)が主催していました。

当時は機械学習の考えに慣れていない社員がほとんどで、教科書を読むのにとても苦労していましたが、途中の数式展開などのアンチョコを光成滋生氏(@herumi)が書いてくれて、それを製本して暗黒通信団から出版したものが『パターン認識と機械学習の学習』という同人誌です。

なぜか、発売当初に一部業界で話題となり、ジュンク堂書店池袋本店では『リーダブルコード』を上回る週間売上数を達成し、同店の2012年コンピュータ書売上冊数ランキングの第3位となりました。

『パターン認識と機械学習の学習 ― ベイズ理論に挫折しないための数学』

光成滋生(著)、暗黒通信団、2012年7月、1,050円(税込)、B5判、99ページ、ISBN＝978-4873101668

この本が売れてしまうということは、みんな機械学習の勉強に苦労しているんだ、と初学者同士で仲間意識を感じることのできた瞬間でした。

原書の基本的な2章から5章、そして後半の難関である9章と10章の範囲をカバーしています。11章の物理学とサンプリング法の関連についてゲストによる書下ろしがあります。

機械学習の理論を学習するためには、大学の数学でやったはずの微分積分と線形代数の復習が必要で、積分の変数変換や行列の各種操作の基礎を忘れてしまったエンジニアが思い出しながら学習する際にお勧めできます。

自然言語処理

データ分析の現場では、アクセスログの中にも検索キーワードの日本語や英語などが含まれていたり、自然言語で書かれた文書のデータを解析の対象とすることがあります。そのときは自然言語処理の基礎を勉強する必要があるのですが、東北大学の乾・岡崎研究室で公開されている「言語処理100本ノック」がお勧めです。

「言語処理100本ノック
―東北大学 乾・岡崎研究室」
http://tinyurl.com/gengosyori100

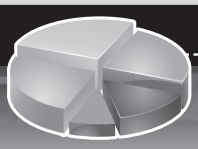
- 第1セット：テキスト処理の基礎
- 第2セット：正規表現・日本語の扱い
- 第3セット：文分割・トークン化(英語)
- 第4セット：(前半)辞書引き
- 第4セット：(後半)Nグラム言語モデル
- 第5セット：形態素解析／グラフ描画
- 第6セット：係り受け解析／クラス
- 第7セット：文脈類似度、クラスタリング
- 第8セット：機械学習／分類器
- 第9セット：データベース
- 第10セット：構造化データ(XML)の処理／CGIによるデモシステム

標準入力からタブ区切り形式のテキストを読み込んで適切な処理を行うUNIXライクな基礎的な課題から初めて、Twitter用のデータ解析の練習などの実践的なプログラミングの課題を通じて、段階的に自然言語処理の技術を学習することができます。

言語処理の最先端では、統計的機械学習の技術が応用されています。更なる発展として高村大也著『言語処理のための機械学習入門』で学習のキーワードを把握すると良いでしょう。

最後に

ここで紹介したように、機械学習やデータ分析の技術や理論を勉強するための教材はすでにたくさんそろっています。「機械学習の勉強をいつやるか？」「今でしょ！」 SD



Column

広がる機械学習の応用とその未来

鹿島 久嗣（かしま ひさし） 東京大学

身近に広がってきた機械学習

少し乱暴な言い方をすれば、機械学習はデータ解析技術の一流派とも言えるものですが、もともとは人工知能の一分野として研究が始まったもので、50年来の研究の歴史があります。その時代時代にさまざまな形で利用されてきました。最近はビッグデータの波に乗り、かつてない注目が機械学習に集まっています（筆者自身、テレビから「機械学習」という言葉を聞いたときには大きな衝撃を受けました）。今では多くの企業がデータ解析技術をその競争力の源泉として位置づけようと力を入れています。

身近なところで機械学習による予測が最も役立っている例が、オンラインショッピングサイトなどで用いられている推薦（レコメンド）システムでしょう。推薦システムはまさに機械学習技術の主戦場のひとつで、米国のオンラインビデオレンタル会社である米Netflix社などは、数年前に100万ドルもの賞金をかけ推薦アルゴリズム開発コンテストを開催しています。

推薦技術とは、過去の購買履歴や個人情報から次に顧客が何に興味を持つかを事前に予測し提示するものです。それがデータの大量化に伴いその予測を個人個人の嗜好に適応させ、今ではよりきめ細かい推薦を実現しつつあります。

ショッピングだけではありません。みなさんの中にはTwitterやFacebookなどのソーシャルメディアを利用している方も多いでしょう。そこでしばしば見られる「この人も知り合いでは？」という新たなつながり先や興味を持つかもしれないイベントの提示にも、推薦技術は用いられています。そこでは「友達の友達は友達」といった社会科学的な知見も利用されています。

みなさんが普段Webを閲覧する際に目にするインターネット広告の配信にも機械学習が利用されています。いくつかある広告のうちどの広告を提示するのが最も見込みがあるか、つまり広告がクリックされ、購買に結び付くのかを機械学習を用いて予測します。

限られた掲載面や表示回数で最大の広告効果を上げるためには、閲覧者の傾向を把握するための情報収集とそれに基づく最適なリソース投入の両方を同時に行う必要があります。これを実現するのがバンディット予測と呼ばれる機械学習技術です。バンディット予測は比較的長い研究の歴史のある技術ですが、いわゆるA/Bテストなどに代わるWebサイト最適化のための強力なアプローチとして、にわかに注目を浴びています。

異常検知に利用される機械学習

発見型の機械学習の応用で、産業上非常に重要なもののひとつに異常検知があります。「異常」とは、たとえばITシステムにおけるコンピュータウィルスへの感染や不正侵入、クレジットカードの不正利用、生産ラインやプラントの故障などのことです。高頻度では起きないが、いったん起こるとサービス停止などの大きな損失を引き起こすため、できることなら起こる前に、あるいは不幸にして起こってしまったとしても、なるべく早い段階で検知することが重要です。

重大な障害であるほどこれまでに起こったことは、ほとんどありません。過去のデータをもとにこれを予測することは困難です。そこで発想の転換、つまり異常を定義するのが難しければ、逆に正常時をとらえるというように考えるのです。システムが正常に稼働しているときのデータはふんだんにあるので、

ここから正常時の特徴をとらえておきます。そしてシステム監視時には、現在のデータの振る舞いが正常時のモデルを逸脱したら警告を発するようにしておくのです。この方法では、どのような異常があるのかまでを知ることはできません。しかし、何らかの異常が起こる可能性に対して警告を発することによって、システム管理者などが追加調査を行うことはできます。

コンピュータは人間を超えたか?

さて、データ解析技術はいまやさまざまな分野において急速に普及が進み、一定の成功を収めていると言えます。これは言い換えれば機械の知が人間の知を脅かしつつある状況とも言えます。最近の話題では、米IBM社の質問応答システム「ワトソン」が人間のクイズ王に見事勝利したという話は記憶に新しいでしょう。ワトソンの実現には自然言語処理技術や音声認識技術はもとより、莫大な知識を蓄え、さらに未知の状況にも適応するためのコア技術として機械学習が用いられているといいます。彼らはこのシステムをさらに医療や金融といった分野に適用することを計画しているようです。

こうなってくると、これまで人間にしかできないと思われてきた知的な仕事がビッグデータを携えた機械達に奪われてしまうのではないか、という不安を抱く人がいるかもしれません(同様の話は産業革命のときにもあったようです)。これは半分正しいとも半分間違いであるとも言えます。以上で触れた例は、かつての人工知能技術が対象にしていた世界と比較すると相当に複雑でオープンな世界ですが、一方でそれでもやはり相当にクローズドな世界です。価値や目的にあいまいさがあったり、ルールを超えた創造性を必要としたりするようなタスクにおいては、まだまだ人類に分があるでしょう。

機械と人間の協調問題解決をめざして

ところで、何年か前に開催されたフリースタイルのチェストーナメント(人間とコンピュータで自由にチームを編成して参加できる)での優勝チームは、数人のアマチュアプレイヤーと数台のパソコンからなる混成チームであったそうです。このことは今後の機械と人間の付き合い方についてある示唆を与えてくれます。つまり、機械と人間、どちらか一方だけでなく、両者が互いを補い合うことによって最高のパフォーマンスが発揮されるという可能性です。

実はこのような考え方は近年、コンピュータ科学分野において注目されている「ヒューマンコンピュテーション」と呼ばれるものです。メディア理解や検索など、ある程度のレベルまでは機械的に可能だが完全に行うには難しいようなタスクを、部分的に人間に依頼することで解決するという、いわば計算機システムの中に人間が明示的に取り込まれた形です。これをリアルタイムに近い形で実現するために、世界中のオンライン労働力にアクセスできるクラウドソーシングなどのインフラが利用されています。

データ解析に話を戻すと、実はデータ解析のプロセス全体において機械学習などの自動的な解析の部分は一部であることがわかります。ビジネス理解や課題設定、データの収集/整理、結果の解釈などデータ解析業務の多くの部分は、いまだ極めて属人的で労働集約性の高いものです。ヒューマンコンピュテーションの例のように、人と機械が互いの利点を活かして協力しあうことでデータをより高い価値につなげていく。それが将来のデータ解析の姿になるのではないでしょうか。SD

CHAPTER 4

ビッグデータだけにとらわれてませんか?

関心事を明確にしたデータ分析と議論の繰り返しこそ本質

チャンス発見技術／データジャケット市場の構想と活動

「ビッグデータ」さえあれば有益な情報が得られると思っていませんか？　それ以前に大切なことを、きちんと押さえておきましょう。

大澤 幸生(おおさわ　ゆきお)　東京大学大学院工学系研究科 教授
URL http://www.panda.sys.t.u-tokyo.ac.jp/

本音から：「何をいまさらビッグデータ?」

ビッグデータという語が流行するにつれ、最近はデータマイニングという言葉が新聞などでも賑わっています。新しくて当たり前ではないパターンをデータから抽出し、有益な知識を得ることがデータマイニング(正しくはKnowledge Discovery and Data Mining)の意味です。

このデータマイニングの分野では、「大きなデータにはチャンスがあるかもしれないが、小さなデータこそチャレンジに値する」という名言が伝えられています。大きいだけのデータから有益な情報を見つけるのは、山の中から特定の1匹の蟻を拾うように至難の業です。鋭い目的意識を持って収集、選別したデータの背景にあるストーリーを読み取ることは、ビッグデータを扱うときも必須になるのです。筆者らが2000年から開発してきたチャンス発見手法(参考文献[1]など)が成果をあげた事例は、次のような質問から始まりました。

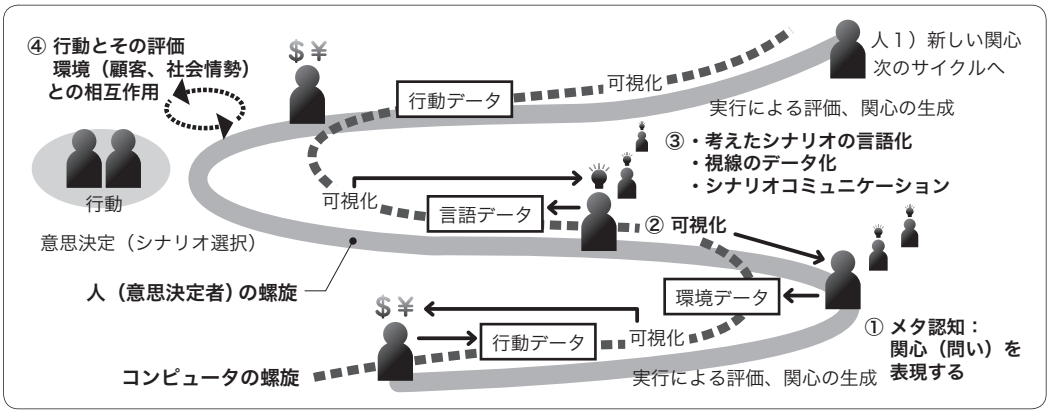
A「売上げにかかわる異常気象を予測できるか？」
B「新製品の商品化率を向上させられるか？」
C「製品改良で競争力を強化できるか？」
など

このように明確な関心を持つことが成功の鍵です。よく研究室に「データが溜まっているんですが、どう使いましょう」と相談に来る人がいますが、相談の前にまず関心を明確にすべきです。

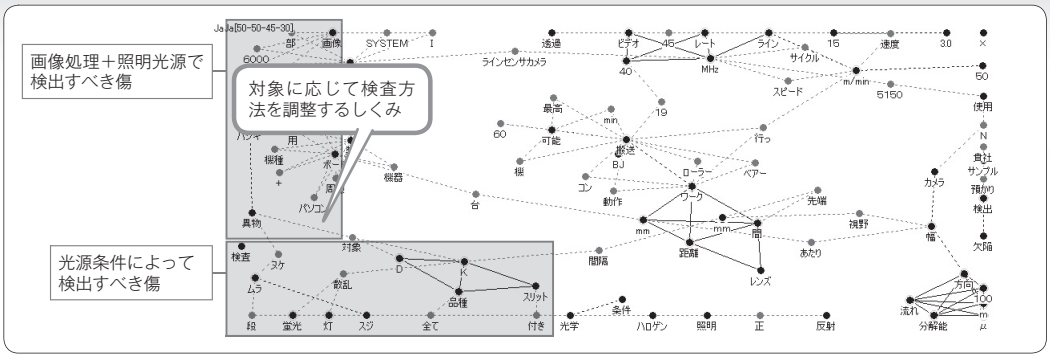
チャンス発見とデータ選択プロセス

図1は、①明確な問いで表される関心を反映

▼図1　チャンス発見の二重螺旋プロセス



▼図2　KeyGraphによる営業報告書集の可視化（図左下の「スジ」「ヌケ」「ムラ」「ハジキ」は傷の種類）



してデータを集め、②可視化して③その結果を見た人が状況理解や行動計画を言葉で表現し、④その内容を議論したり実行したりする中で新しい問いを生み出すという、チャンス発見のプロセスです。ビジネスチャンス発見を持続させるためには、このプロセスの持続が必要です。

たとえば先のCに該当する例として、ある企業でフィルム表面の傷検査装置を製造していました（参考文献[2]）。これは特殊な装置で販売担当者もユーザも専門的な技術者でしたから、ユーザの意見を聞いて販売担当者が書いた報告書は、それぞれ専門的視点にはまり過ぎ、専門領域の壁を超えて開発費を要求したり、新しいユーザ層を開拓したりするような改良案には達しませんでした。

そこで、さまざまなユーザの声を集めたデータを**図2**のように可視化ツールKeyGraph（参考文献[3]）で可視化したところ、フィルム表面を照らす光源の専門家と画像処理ソフトの専門家が技術知識を持ち寄り組み合わせれば、さまざまな傷を自動識別できることに気づきました。そして、光源の専門家と画像処理ソフトの専門家が各自の意見を書き出しながら議論した結果、「2種類の光源を用い、スリット幅を調整しながらフィルム表面を照明して写真を撮り、多様な傷を区別する」という技術で装置を改良し、売上を伸ばすことができました。

その後、自社技術の強みを発揮するために何をすれば良いかという新たな関心が生まれました。関連特許明細を集めたテキストデータを可視化し、自社技術を組み合わせた「検知した傷の近傍に熱硬化性樹脂でマーキング」という案を生み、新特許申請に進むことができました。

このプロセスを**図1**に照らせば、①で自社製品の競争力を高めるための問いを発して集めたデータを②で可視化し、③と④で可視化した結果を元に製品改良案を言語化して話し合い、自らの思考の中のつながりを自覚しています。**図1**のプロセスを2周回していることがわかりますね。

豊かなデータから分析的に高頻度パターンを求めて未来を予測するデータ駆動的な考え方は止めて、「長期間の経験を俯瞰する大局視点と、低頻度の事象に着目する局所視点の相互作用によって低頻度事象の意味を理解する」というヒューマン駆動プロセスにデータを投入したことが成功の鍵となりました。

ゲーム化されたチャンス発見プロセス

チャンス発見のプロセスにおける議論の品質に持続性を持たせるため、遊びの要素を取り入れたのがイノベーションゲーム（**図3**）です（参考文献[4]）。これは、市場を模したボードゲームです。英語名はInnovators Marketplace（以下、IMと省略）と言いますが、厳密にはIMはゲーム前後の処理も含めて持続的に循環させるプロセスを指します。

IMの参加者は「起業家」か「消費者」のいずれ

▼図3　イノベーションゲームのシーン

かの役割に沿って発言します。既存の知識や技術の関係を可視化した図を見ながら商品や概念を組み合わせてアイデアを作る「起業家」（3〜4名）がおり、彼らから提言される製品やサービスの案を「消費者」らが評価してゆきます。アイデアが気に入った消費者は、提案した起業家と価格交渉して（通貨：モンキー）購入します。

起業家間では稼いだ金額を競い、消費者間では購入したアイデアで生活品質が向上した度合いを競います。そのため、起業家はアイデア実現手法の効率化によるコスト節約や、価格に見合う品質向上の努力を行います。一方、消費者は買うアイデアの価格を下げ、質を上げるために批判的発言も行いますが、楽しいゲーム感覚なので嫌な雰囲気はほとんど起きません。参加者は批判を改善に生かして、発想と競争に引き込まれてゆきます。

市場化：データ共有・選択への現実的路線

ところで、読者の周りではどんなデータが手に入りますか？　実際、Facebookから集めた大量のエントリーや公開された個人情報を元にニュース記事を推薦する会社もあり、地上から見た気象状況から高精度の天気予報に成功した例もあります。しかし、もし個人が公開しないような情報――医療検査結果や仕事のスケジュールなど――も入手できれば、気象情報とも組み合わせてその人の体に必要な飲食物も推薦できるはずです。このようなビジネスを興したい人にとっては各種データが公開されると好都合でしょう。

しかし、有用なデータの提供を受けるのは実際は困難です。個人が医療検査データや1日のスケジュールというデータを公開したり、企業が顧客の行動を公開することは難しいものです。個人情報を含まない場合でも、商品価値のあるデータなら無料では提供されません。

こうした場合、データの元となった行動主体にデータ提供条件を直接交渉したり、研究目的のみにデータを使うことを約束したり、要求される金額を支払ったりと、個別の対応によってデータ提供を受けられる可能性は残っています。つまり、データの価値や、その価値評価の元となる利用シナリオについて十分に話し合える環境が存在していないことが本質的な問題です。ここでいう利用シナリオには、他のデータとの組み合わせや、多様な分析ツールと結合して発揮しうる事業推進効果などさまざま考えられます。

このようなニーズを背景に、データの市場が生まれています。KCAN、KDnuggetsなど（Google検索してみましょう）では、誰でもデータ所有者に連絡して提供を受けられる環境が整っています。条件さえ整えばデータを入手できるのなら、売り手と買い手が同意した価格で物を売買する市場のようなものです。必要な人にデータについての情報提供や取引の橋渡しを行うブローカー事業も欧米では盛んです。データを無理にオープン化するよりも、自由市場の原理でユーザが必要なデータを選び入手できるようにしたデータ市場は、今後普及するでしょう。しかし、まだ大切なことが欠けています。

「データジャケット」で市場をつくろう

本来、市場というものは、提供者と消費者の間での「提案」「評価」というコミュニケーションが活性化すればイノベーションの場となり得ます。ところが、Webページ上でデータの表層的な情報を列挙しただけのデータ市場では、

価値あるデータを選んで入手するために必要なだけの交渉や熟考ができません。これを可能にするのが「データジャケット」です。

DVDやCDの店舗の棚に、よく「これはジャケットだけで中身は入っていません」という陳列を見かけます。もし万引きされてもジャケットだけならまた作れば良い——この考えを、多様なデータの共有と価値評価、選択のしくみに発展させるのが「データジャケット上のIM(Innovators Marketplace on Data Jackets：略してIMDJ。参考文献[5])」の考え方です。IMDJは、次の3ステップからなります。

Step(1)

まず、各データの所有者は公開できる範囲でデータ中の変数をリストアップします。この公開可能な変数のリストと、その他のデータについて宣伝しておきたい情報を書き込んだセットを「データジャケット」(DJ)といいます。

データの内容は所有者の都合で隠しても良く、DJにはデータ内容の大枠だけを書いて公開しましょうというわけで、いわばDJはメタデータを書いたDVDジャケットのようなものです。たとえば、何かの配管についてのデータなら“腐食の度合い”などが変数でしょうし、POSデータなら“顧客の年齢”や“購入品目”などが変数です。地震履歴データに対応するDJは**図4**のようになります。データが論文のようなテキストになっている場合には、@keywordという変数を作成し、代表的な語を値としても構いません。各変数について公開可能な度合いを設定しても良いでしょう。DJは、公開可能変数リストと特徴記述(自由文による概要や詳細度等)などからなります。

Step(2)

次に、DJとDJの間を、近い意味の公開可能変数を媒介として結びつけたマップを作成します。これはさまざまなデータのメタ情報を公開するCKANのようなデータカタログや、データ間の関連性を書いて宣言しておくLinked open Dataとは異なります(リンクを書いても良いですが、データは公開を前提としないのでリンクづけも強く推奨しません)。近い意味の変数を結びつけるため、辞書などを介在させて関連を可視化する技術にも需要が生まれるでしょう。

Step(3)

データを使うユーザと、データを提供する所有者が価格を交渉し、取引が成立する条件(価格など)を認めていくという原理で、データの利用法と評価価値を決めていきます。**図5**のように、Step(2)で作成したグラフをゲーム盤として、IMの方法を適用するわけです。その結果、一般ユーザにとってのデータの価値は定価となり、個別ユーザにとっての価値は交渉によって決めることができます。これによってユーザは自分にとって価値のあるデータを選択でき、データ所有者は条件を吟味して提供できます。

ただし、公共的な価値のあるデータ(道路交通における事故履歴など)は需要者が多いから

▼図4　地震履歴データに関するデータジャケット例

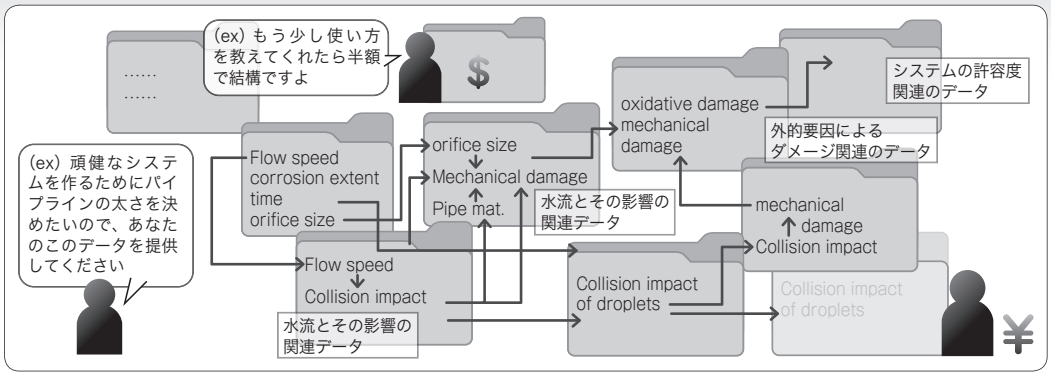
@data_sample ——— 変数が“@”で開始する書き方。その他@data_abstractなど、適当に記述内容のカテゴリ名を作ってOK

日付	発表時刻	発生時刻	震源地	マグニチュード	最大震度
2011年12月31日(土)	23時56分	23時51分	種子島近海	M3.0	1
2011年12月31日(土)	22時08分	22時03分	福島県中通り	M2.7	1
2011年12月31日(土)	21時20分	21時16分	浦河沖	M3.9	1
2011年12月31日(土)	19時33分	19時29分	宮城県沖	M3.9	2
2011年12月31日(土)	14時54分	14時48分	福島県沖	M3.8	1

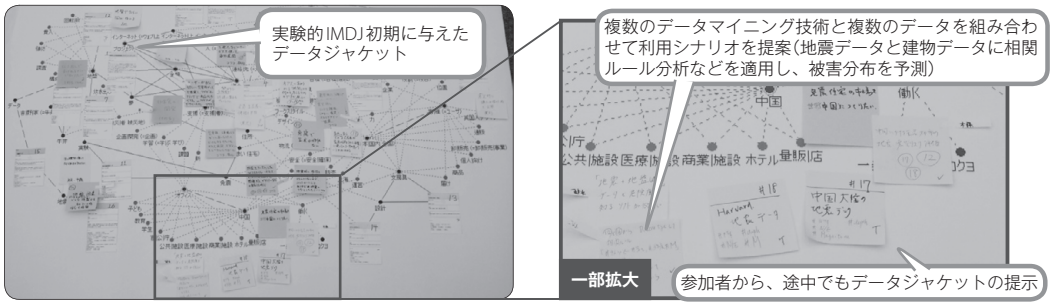
@data_name 日本の地震
@provider サル丸
@日付 real
@発表時刻 real
@発生時刻 real
@震源地
@マグニチュード real
@最大震度 real
変数名

@potential_value
ある地点を境目に揺れ方が違っていることが発見されたなど、予知に関連する結果を得られると期待される
——— potential_value(データの持つ潜在的価値)という変数を作成して文章を変数値とした例

▼図5　IMDJにおけるStep(3)の様子



▼図6　紙上でのIMDJ実験例：実験前はPOSデータなどの評価が期待されたが、IMDJの実施結果、地震被害データや文具利用者アンケートが予想外に高評価を得た（大澤研究室・劉暢君らが主催した例）



高額になるかというと、必ずしもそうではありません。「このデータを分析することによって交通事故を減らすことができます」という買う側の説明が通れば、社会的責任を重視する企業なら無料でデータを提供してくれるでしょう。筆者は、社会貢献こそ利益の元だと考えて成功している企業経営者と数知れず交流してきました。

おわりに

現在、IMDJについては研究室で基礎実験を行い、効率的で効果的な運営の必要条件を研究中です（**図6**）。しかし、市場ですから、実現段階では参加者が自分の利益のためにDJを作成してゆくことが前提です。参加者駆動の体制づくりのため、国内外でのワークショップなど、IMDJの良さを知ってもらう目的を含む活動を進めています[注1]。さらにIMDJの実現と推進のためには、データ可視化、変数辞書の精緻化などが今後必要と考えられます。これらに加え、データの分析者が他のデータで成功した分析事例を購入して学ぶような場づくりもIMDJに期待されます。その意味で、データサイエンティストを志す人たちにもぜひ、筆者たちの活動に加わってほしいと考えています。SD

参考文献
[1] 福田寿ほか「マーケティングにおけるチャンス発見」大澤幸生（監修・著）：「チャンス発見の情報技術」17章：東京電機大出版（2003）
[2] 堀江健一・大澤：製造業における組織的チャンス発見によるデザイン事例：電子情報通信学会技術研究報告. KBSE, 知能ソフトウェア工学 105（546）, 7-12, 2006-01-17（2006）
[3] Ohsawa, Y., Benson, N.E., and Yachida,M., KeyGraph: Automatic Indexing by Co-occurrence Graph based on Building Construction Metaphor, Proc. IEEE Advanced Digital Library, pp.12-18（1998）
[4] 大澤：イノベーションの発想 技術，日経出版（2013）
[5] Ohsawa, Y., et al "Data Jackets for Synthesizing Values in the Market of Data" 16th International Conference on Knowledge-Based and Intelligent Information & Engineering Systems（2013）

注1）http://www.panda.sys.t.u-tokyo.ac.jp/MoDAT

第2特集

原因

自分の定規（ものさし）を持っていますか？

ボトルネックを探れ！

「ベンチマーク」活用テクニック

新しいサーバマシンを購入し、いざ現場投入！　思ったほどのパフォーマンスが出ない……ということはありませんか？　カタログスペックを調べて、その性能を期待しながら購入して、まず行うべきは「ベンチマークテスト」です。 本特集では、まずは自分の身の回りで使うPCを題材に、そのアーキテクチャを知ることでどこがボトルネックになるのか、その原理を学びます。そしてネットワーク環境でサーバマシンがどのように実力を発揮するのか、さまざまなベンチマークテストを通して、エンジニアとしてマシンを正しく評価するノウハウを習得していただきます。ベンチマークテストの基礎を押さえ、どんなマシンでも評価できる自分だけの定規を作ってください！

CONTENTS

Part 1 ベンチマークの基礎を学ぶ[PC編]

本PartではPCのベンチマークテストについて、その必要性と利用方法について解説していきます。たいていのベンチマークテストは、クリック1つで勝手にいろいろなデータ処理や表示のテストを行い、その結果を表示してくれます。しかし、ベンチマークをより深く理解すれば、そのベンチマーク結果が目的に合うのかを知ることができるようになり、さらに有効に活用できます。

Writer 円藤 優沙（えんどう ゆうさ）／ Twitter @YuusaEndo

ベンチマークの必要性

ベンチマークというものを何度か耳にしたり使ったりしたことがある人もいるかと思います。ベンチマークテストとはコンピュータの処理速度を計測する試験のことで、テスト用に作成されたソフトウェアを対象のコンピュータ上で実行してその処理速度＝性能を求めることです。

少し前「2位じゃダメなんでしょうか？」という言葉が流行りましたが、あれもスーパーコンピュータ上で「LINPAK[注1]」というベンチマークプログラムを動作させたときの速度で順位を付けていて、それがどうのこうの、といった話でした。

ベンチマークはインテルのパソコンばかりでなく、いろいろな構成をしているコンピュータの性能を測る指標としても扱われます。ベンチマークのスコア（テスト結果）を知ることで、コンピュータの性能の高さを共通のモノサシのうえで知ることができるようになります。

しかしTOP500の競争のためにベンチマークを使うのは納得できても、一般的な組織で使うコンピュータの性能はベンチマークをしてまで知る必要はないのではないか、と思われるかもしれません。それは一面では正しい考えです。しかしベンチマークテストを使うことで、業務の処理の速度を上げたり、コストを抑えたりすることも可能になることがあります。本稿ではベンチマークをうまく使うことで、限りのあるリソースを有効に活用しようというのが狙いです。

ベンチマークを使う目的

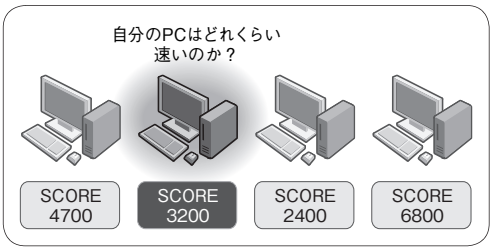
ベンチマークのスコア（結果）は単独では何の意味もありません。モノサシであるわけですからあくまで何かと比較したうえで良いのか悪いのかの判定が可能となり意味を持つのです。

ベンチマークを使う目的としておもに次のようなことが挙げられます。

ケース1「利用中のコンピュータの性能を周囲の環境と比較する」

図1のように、PCのハードウェアの進化とソフトウェアの複雑化は相互依存関係にあります。今使用中のPCが快適に使えたとしても何年

▼図1 利用中のコンピュータの性能を周囲の環境と比較する



注1）http://ja.wikipedia.org/wiki/LINPACK もしくは、http://www.top500.org/project/linpack/

かすると、あるいは新しいソフトウェアを導入すると、動作が重くて使いにくいという状況になることがあります。ベンチマークはそのようなタイミングでさらに使い続けるのか、そろそろ替え時なのかを定量的に推し量る指標とできます。

簡単に言うと、今使っているソフトウェアはこれだけ重くなったので、それと見合う性能のものにしたい、という時にその感覚的な動作の重さを定量的に表すことが可能になります。単にパソコンが重くなったので変えたいというのではなく、たとえばこの数値は世間に比べてこんなに見劣りするからこうしたい、と定量的に話をできるようになるということです。

ケース2「使用中のコンピュータの性能を従来より改善することによって安定動作や高速な処理を実現する」

後の節で説明しますが、コンピュータの処理は突き詰めた言い方をすれば「データをバケツリレーで出力まで運んでいくようなスタイルの動作」をします。ですからこのバケツリレーの役割をする要素に遅れがあれば、処理全体に遅れが生じることになります。この遅れの部分をボトルネックと呼びます。ベンチマークテストのもう1つの目的は、PCの性能上のボトルネックを探し、性能とコストのバランスを考えながら改善していくことにあります。

たとえばサーバであれば、トランザクションが急に増加した時でも安定動作できるマージンができたり、ワークステーションであれば、シミュレーションの計算時間を短縮できるというようなケースです。このケースでの究極のベンチマークテストは、実際使用するプログラムを動作させ処理速度をはかるものですが、とはいえ実際の環境を構築するのはとても困難であったり、単独のプログラムの速度向上をしたとしても類似する条件で必ず性能が高いとは限りません。ですからある程度汎用的なプログラム＝ベンチマークでPCの性能を測定することに意義があります。

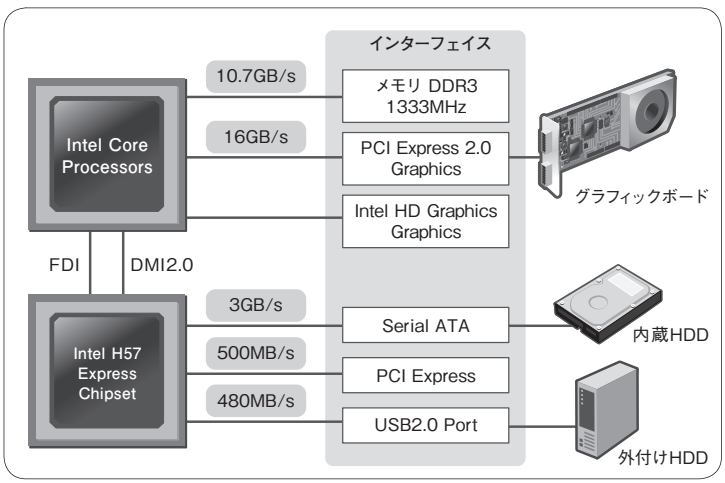
ベンチマークスコアを知るための予備知識

一般的に、ベンチマークプログラムでは実際の利用でよく用いられる複数の関数を組み込んだり処理を想定して構成されており、総合的な結果としてスコアという1つの数字が出てくるものが少なくありません。ただし最終的なスコアを算定するにいたった背景は詳細を読み解かなければなりません。それを理解するために、まずPCの大まかなデータ処理の流れについて説明をしておきましょう。

転送速度を確認しよう

図2は、インテルのCore iシリーズの各種のバスの転送速度をまとめて示したものです。各インターフェースの転送速度はインテルの資料から読み取ったものですが、実際には転送用のラインが複数本あったり、効率良くデータを転送させるしくみがあったり（たとえばメモリイ

▼図2　コンピュータのブロック図と転送速度（インテルH57 Expressチップセットの例）



ンターリーブなど)するため、必ずしもこの数値そのものが実際のシステム内でのデータの移動可能速度ではないことは理解してください。

この構成では、CPU(Processor)は内部にわずかながらキャッシュメモリを持っておりその中で命令を処理し、処理されたデータはI/Oをコントロールするチップセットを介してHDDなどの記憶装置にデータを送ります。HDDに記録されたデータは命令に応じて再度CPU内部で処理されグラフィックとして出力します。各部へはさまざまな速度を持つバスを経由してデータが送られます。また、各部で受け取ったデータはそれぞれの要素によって異なる速度でデータの読み書きが行われます。

ですから、PCの性能は、

- CPUの処理能力
- 各種バスの転送速度
- 各要素の処理速度(記憶装置のRead/Writeなど)

の組み合わせで決まります。

このような流れがデータの基本的な流れなのですが、この流れは処理するデータサイズや他の命令との優先度の兼ね合い、各要素が利用される頻度などで大きく変化します。

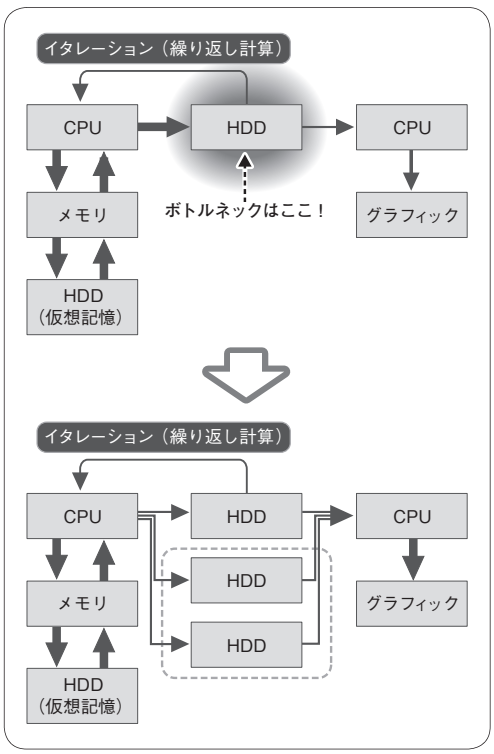
処理が遅くなる原因とは何か？

たとえばCPU内部のキャッシュメモリサイズよりデータが大きい場合、外部メモリにアクセスします。外部メモリはキャッシュメモリより大容量ですが途中の転送速度や記憶速度はキャッシュメモリより劣ります。この外部メモリでも処理できないほどのデータがある場合には、今度は外部メモリから溢れたデータを仮想記憶領域としてハードディスクに保存します。このようなケースではメモリよりさらに低速なバスを経由して一部のデータがハードディスクへと移動し低速なデバイスでの読み書きが行われるために極端にパフォーマンスが低下します。さらにこの仮想記憶用のハードディスクが外付けのハードディスクだった場合はどうでしょうか？　外付けのハードディスクも内蔵ハードディスクと同じRead/Writeの性能だったとしても、途中のUSBの転送速度が遅いため内蔵ハードディスクよりさらにパフォーマンスは低下します。ただしこれはあくまで仮想記憶が使われる場合の話です。

つまり、コンピュータの性能は使い方と構成によって大きく変わってくるということが言えます。ですからベンチマークでは実際の使い方に近い処理を負荷としてかけ、各要素での性能を評価することを行うことが重要なのです。

この各要素の性能を改善する必要があるとわかった場合はその部分のパフォーマンスの改善を図ることになります。これをもう少し具体的な例で示すと、たとえば流体解析のような計算では1つの状態を計算した後にその計算結果からの変化をさらに計算しその計算結果からの変化を……、といったイタレーション計算が必要となります(**図3**)。このような計算を想定した

▼図3　繰り返し計算



ベンチマークを行った結果、たとえばハードディスクの部分で速度が遅いとわかった場合、複数のハードディスクをRAID構成にして読み書きの速度を上げる対策を行います。

ベンチマークテストの種類と注意点

ベンチマーク用プログラムは多くの種類があります。本稿ではベンチマークの次の3種類について紹介します。

●統合ベンチマーク
CPU、ストレージ、グラフィックといったPCのすべての要素をまとめてパフォーマンスを評価するものです。

●構成要素別ベンチマーク
ストレージ、グラフィックなど個別要素のパフォーマンスを評価するものです。

●目的特化型ベンチマーク
実際に使用するアプリケーションに近い機能のパフォーマンスを評価するものです。

ベンチマークプログラムによっては、たとえばDirect XやOpenGLなどのグラフィックライブラリの性能を評価するものがありますが、これらはゲームやCADなどで必要とするものでサーバとしては不要です。ビジネスでMS Office一揃えを使う人は浮動小数点演算が速いという必要もありません。そういったベンチマークの取捨選択をすることによりはじめて適切な評価ができるようになります。

またベンチマークプログラムの中には、64bitに対応していないものや特定のマルチコアCPUに対応できないものなどの制限があるものもありますので、ベンチマークプログラムの使用前にはReadMeをよく読んでおくことをお勧めします。

ベンチマークをやってみる

ここから、さまざまなベンチマークを紹介していきその使い方を説明します(おもにWindows版を紹介します)。

統合ベンチマーク

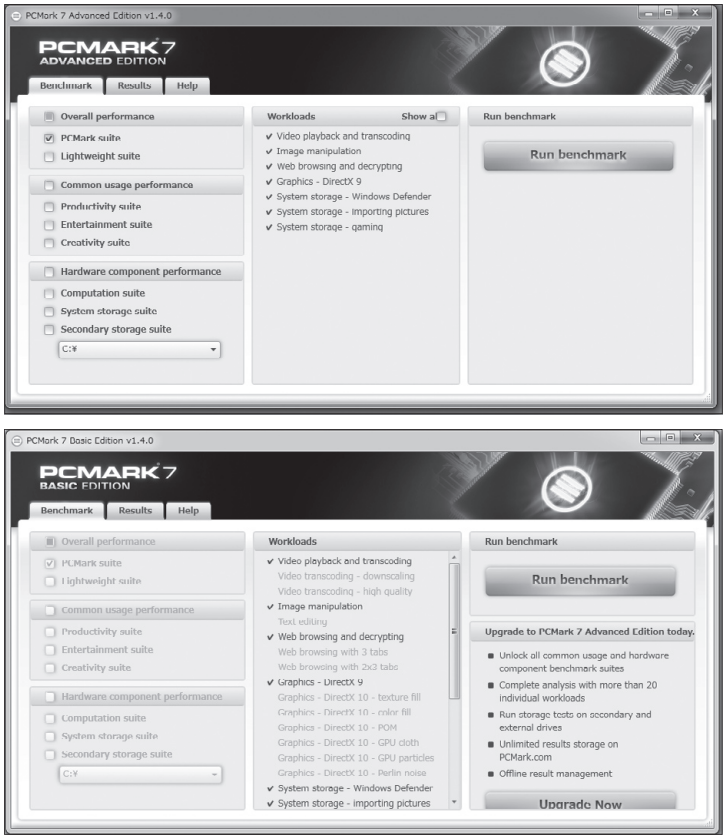
◆ PCMark 7

PCMark7(**図4**)はFuturemark社が開発したPCのアプリケーション実行時の総合的なパ

▼図4　PCMark(Videoテスト中の画面)

フォーマンスを計測できるベンチマークです。3つのEditionがあります。いちばん基本のBasic Editionは非商用に限り無償です。さらに詳細なテストが実行できるAdvanced Editionとコマンドライン自動実行やサポートが受けられるProfessional Editionがあります(**図5**)。Basic Editionでは、Video playback and transcoding、Web Browsing、DirectX 9、System Storageなどの個別結果とそれらの統合したPCMarkスコアが得られます。またAdvanced EditionではPCMarkスコア以外にCommon usage performanceと呼ばれる用途別のテストとComputation、System storageというPCの各構成要素ごとのパフォーマンスの計測結果も得られるので、目的に応じたテストが実行できます(**表1、図6**)。

▼図5　Basic EditionではPCMark Scoreだけが計測されるが(上)、Advanced Editionでは詳細結果を表示できる(下)



◆結果を見る

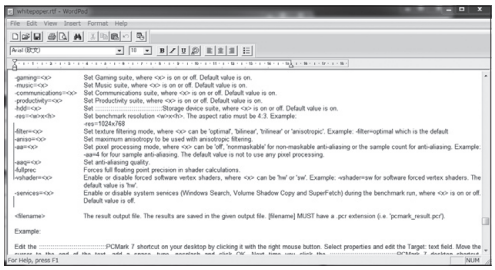
PCMarkの計測の結果は、FuturemarkのWebサイト上に記録されている他のシステムのスコアとの比較ができます。PCMarkのソフトウェアのview Result on PCMark.comのボタンをクリックすると自分のテスト結果が表示されます。この結果を、たとえば他者が行った結果を比較するには、サイトにあるResultsタブに移動し、比較したいシステムを検索し、[Compare]ボタンで比較することで比較対象のシステム構成の選択ができます。

図7は同じCPUを持つスコアの高い例との比較ですが、ほとんど互角のパフォーマンスなのに、著者のシステムのグラフィックの性能が極端に悪い結果となっていることが

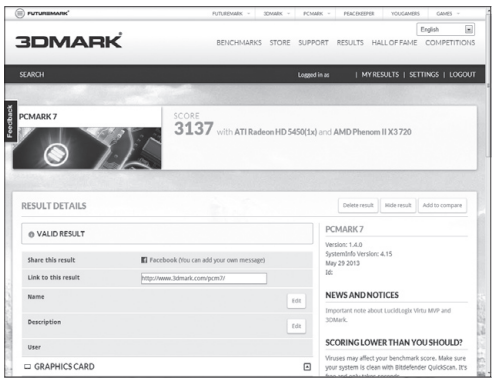
▼表1　PCMark7の評価項目(評価項目のセットは次のようなスイーツの形で提供されています)

PCMark suite	数値演算、ストレージ、表示などの統合的なテスト
Productivity suite	Text Editing, Web Browsing and decrypting, System storageなど事務系処理のテスト
Entertainment test	Video playback and transcoding, DirectX 10などのエンターテイメント系のテスト
Creativity suite	High Quality Video transcoding, Image manipulationなどの画像処理系のテスト
Computation suite	Video transcoding, Image manipulationのテスト
System Storage suite	各種データ(pictures、video、music、gamingなど)の入出力テスト

▼図6　Text Editingのベンチマーク結果画面



▼図7　PCMarkの結果はWeb上に記録される



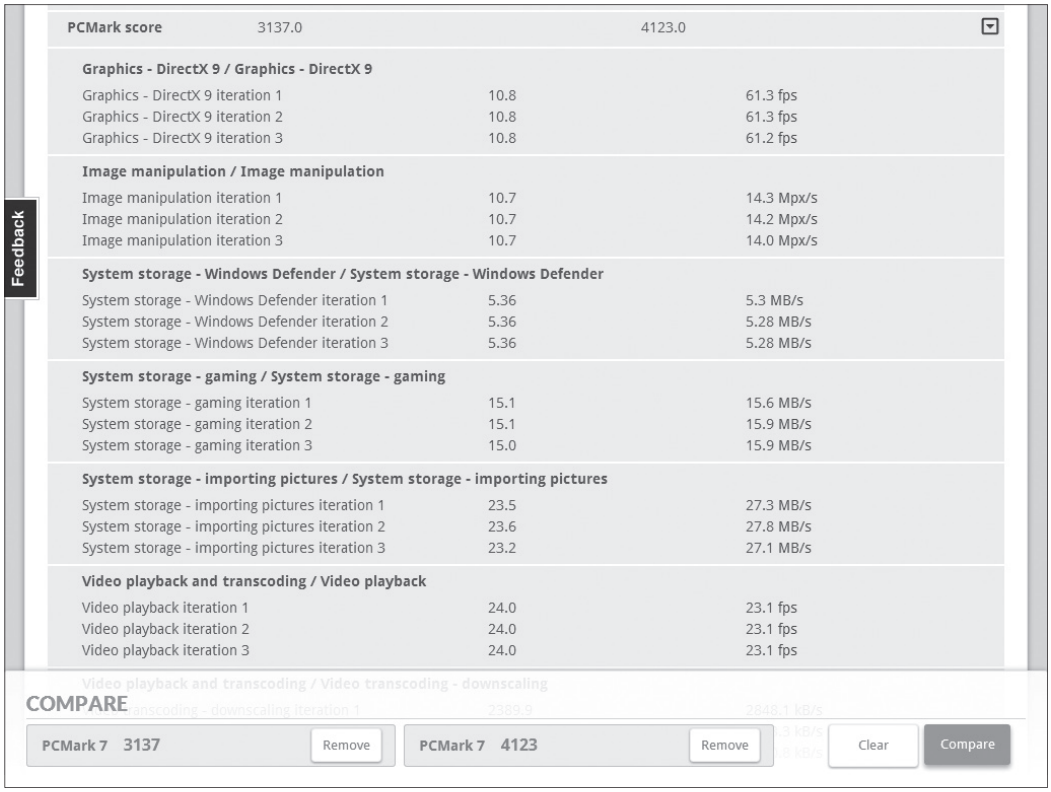
わかります(図8)。

ちなみにこの結果を異なるCPU(Intel Core i7)の最もスコアの高い結果と比較したところグラフィック(DirectX 9)で9倍、web browsingで20倍の差があり、ショックを受けました。ただしDirectX を使ったゲームはしませんしWebブラウジングでも遅くて支障を感じているわけではありませんのでさほど気にはしていません。

構成要素別のベンチマーク

PCの要素別のベンチマークで有名で派手なものはグラフィック関連にいくつもあります。グラフィックのベンチマークはOpenGLとDirectXの2系統があります。OpenGLはオープンでマルチプラットフォームのグラフィックライブラリでおもにCADやCGなどで用いられ、DirectXはMicrosoft Windowsのゲームでおもに使用されますが最近ではCADなどでもサポートされるようになってきています。

▼図8　自分のシステムのスコアは、Web上の他のシステムとの比較が可能



◆OpenGL用ベンチマーク

MAXONのCINEBENCH(**図9**)はOpenGLグラフィックとCPUの2つの要素のベンチマークテストが可能なベンチマークソフトウェアで、WindowsとMax OS Xの両方に対応している無償のツールです。実行はCINEBENCHの実行ボタンを押すと、ベンチマーク測定が始まるというシンプルなものです。スコアは左側のウィンドウにランキングとして表示されます。またCINEBENCHの結果はPC評価の雑誌やWebサイトでもよく掲載されています。CINEBENCHではマルチコア時のテストに加え、シングルコア時の評価もできます。これはマルチコア動作時とシングルコア動作時でどの程度処理速度が変わるのかを知るのに有効です。また、コマンドラインから動作させることもできます。

◆Direct X用ベンチマーク

Futuremarkの3DMark(**図10**)はCINEBENCHと同様に多くのPC雑誌やWebサイトでDirectX用のベンチマークソフトウェアとして利用されています。執筆時点ではDirectX10までに対応する3DMark VANTAGEとDirectX11に対応する3DMark11があります。また同社のPCMarkと同様に無償版のBasic Editionとカスタムセッティングが可能なAdvanced Edition、コマンドライン自動実行が可能でデモ用の無限ループが可能なProfessional Editionがあります。おもに負荷の高いゲームを想定したベンチマークテストです。3DMarkもPCMarkと同様にFuturemarkのサイトに多くのベンチマーク結果が掲載されています(**図11**)。

◆ストレージ用ベンチマーク

HDDやSSDなどのストレージもPCのパフォーマンスを決める大きな要因であることは先に書いたとおりです。無償でストレージのパフォーマンスをわかりやすく評価するのに便利なものに、Crystal Diskmark(**図12**)があります。シーケンシャルRead/Write、ブロックサイズを変えたランダムRead/Writeの評価がで

▼図9　MAXON CINEBENCH(OpenGL用ベンチマーク)

▼図10　Futuremark 3DMark(Direct X用ベンチマーク)

き、結果はデータ転送速度の実行値で表示されるので、結果を客観的に理解しやすいのが特徴です。またテストに用いるファイルサイズを変えられるのも特徴です。この機能は、ストレージが非常に大きなキャッシュメモリを持っているものがあり、ファイルサイズが小さい場合にはキャッシュメモリの性能が評価結果として測定されてしまうことを避けるために搭載されています。以前、実際のテストでキャッシュから溢れた場合に極端に書き込み性能が落ちるといったケースがあることがわかり、ファイルサイズを変えたテストが追加されたという経緯があるようです。

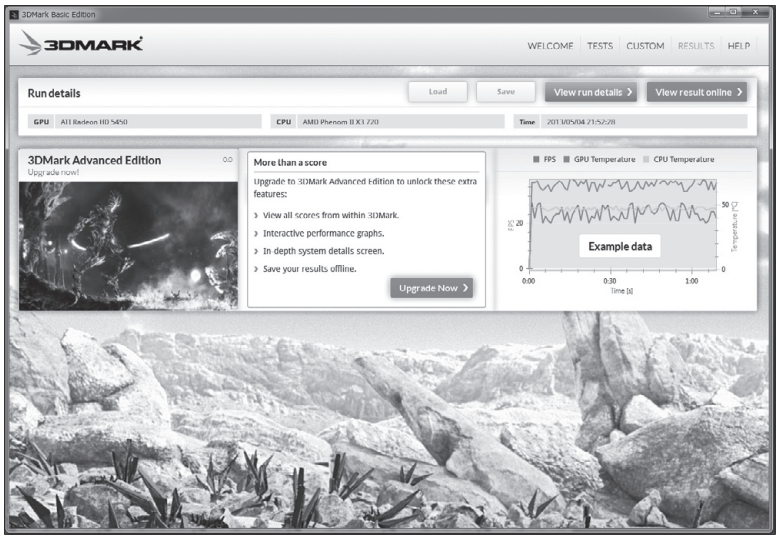
アプリケーションに近いベンチマーク

◆姫野ベンチマーク(himenoBMTxp)

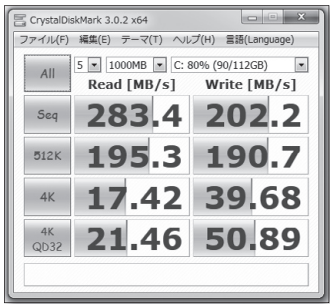
一般的なPCの使用では整数演算が主体ですが、流体解析のようなシミュレーションでは1つの状態について浮動小数点演算を行い、その結果を基に次の状態を計算するというイタレーション計算が必要になります。この計算は何万回以上というスケールでの計算が必要になるため計算速度が非常に重要です。姫野ベンチマーク(**図13**)はこのようなケースを想定して理化学研究所情報基盤センターで作成された非圧縮流体解析コードのベンチマークです。

計算された結果はFLOPS(フロップス、Floating-point Operations Per Second：1秒間に浮動小数点数演算が何回

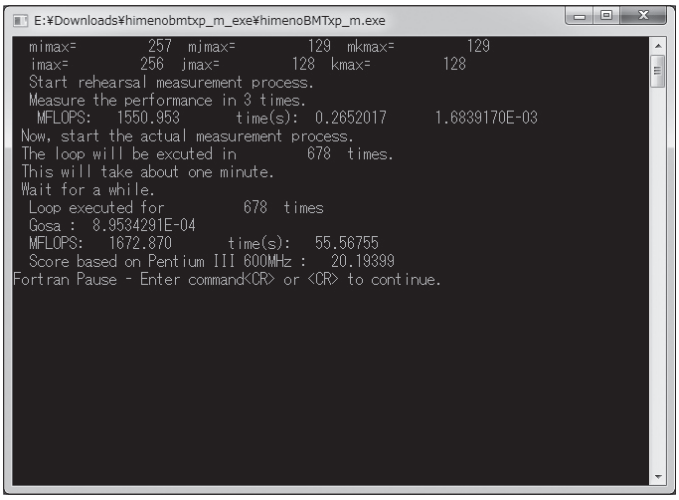
▼図11　3DMarkのWeb上のスコアの表示



▼図12　CrystalDiskmark



▼図13　姫野ベンチマーク



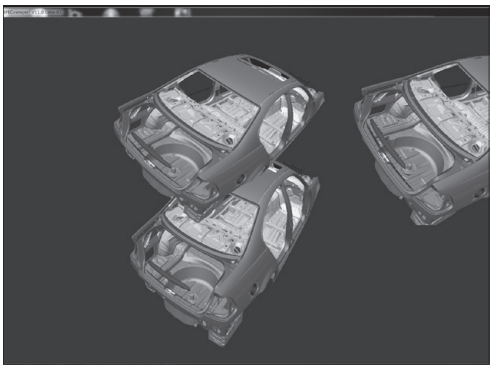
できるか)という単位で表されます。Windows、Mac版ばかりでなくFORTRANやCのソースコードも公開されており、広く利用できます。

◆SPECviewperf

SPEC(Standard Performance Evaluation Corporation)が提供するOpenGLの性能評価を目的としたベンチマークですが(**図14**)、最大の特徴はいくつものアプリケーションがベンダーによって提供されている点です。したがって、実際に使用したいアプリケーションそのものでのパフォーマンスを測定できます。ただし、グラフィックの評価だけで、結果はframes/secでフォルダ(¥SPECv¥SPECviewperf¥viewperf¥viewperf11.0¥results)の中に書き込まれます。アプリケーションはCAD、CGなど次のようなものがViewsetという形で提供されています。

＊()内はviewset名
LightWave (light01)
CATIA (catia-03)
EnSight (ensight-04)
Maya (maya-03)
Pro/ENGINEER (proe-05)
SolidWorks (sw-03)
Siemens Teamcenter Visualization Mockup (tcvis-02)
Siemens NX (snx-01)

▼図14　SPECviewperf



その他

究極のベンチマークは利用したいアプリケーションそのもので行うことです。たとえば非常に有名なシミュレーションソフトウェアANSYS Fluent(**図15**)ではベンチマークのコードを提供しています[注2]。この実行結果を、既存のデータと比較することで性能をベンチマークすることができます。

◆TPC-C

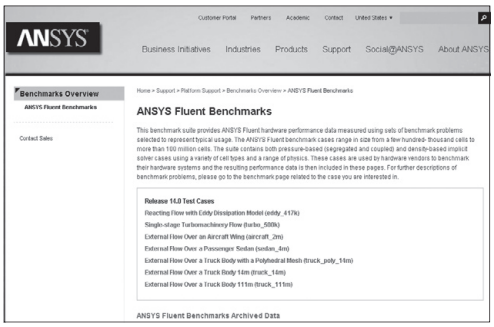
サーバのトランザクション処理のベンチマークの1つです。Transaction Processing Performance Council[注3]が提供するベンチマークです。PCやデータベースのパフォーマンスを測定することができますが、ここでは詳細は割愛します。

おわりに

以上ベンチマークを紹介してきましたが、興味の対象が専門的になればなるほどベンチマークを行ううえで必要な専門的な知識も深くなります。よくベンチマークを理解して、PCリソースの有効活用を進めていってください。SD

注2) http://www.ansys.com/Support/Platform+Support/Benchmarks+Overview/ANSYS+Fluent+Benchmarks
注3) http://www.tpc.org/tpcc/

▼図15　ANSYS Fluentのホームページより



Part 2 ベンチマークテスト入門［サーバ編］

Part2ではサーバに関するベンチマークを取り上げます。ベンチマークをするにあたってどういった項目を、何のために行うか計画を立て、どんなところに注意したら良いか。そしてサーバを構成する、CPU、ディスク、メモリ、ネットワークについて個々のベンチマークを解説します。

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

サーバのどこを見るか

サーバはCPU、メモリ、HDD、ネットワークカードなど、さまざまなパーツから構成されています。サーバを購入もしくは利用するとき、少なからずサーバのスペックを考えると思います。その際、重要視する一番目か二番目には処理性能が出てくると思います。

サーバの処理性能に最も大きく影響を及ぼすパーツはCPUです。CPUを選ぶためにコア数や動作周波数を決めることでしょう。

次に処理性能に影響を及ぼすパーツはメモリやディスクです。必要な容量だけで選ぶかもしれませんが、その処理性能を考慮することもあるでしょう。とくにディスクはほかのパーツと比べて速度が遅いためボトルネックになりやすいです。

処理性能を知るためにどこを見るか

では、それらの処理性能がどれくらい速いのか、どうやって調べれば良いでしょうか。

まずは、サーバやパーツのメーカーサイトを見てみましょう。スペックシートがあるかもしれません。そこには、理論的なピークの性能値や、理想的な環境で動作させたときの性能値があるかもしれません。これらの情報は、参考資料としては役に立ちますが、実際に自分たちがサーバを利用する環境とはかけ離れていることが多いです。

自分が利用する環境に合わせた性能値が知りたい。そのような場合は自分でベンチマークをとってみましょう！

どこのベンチマークをとるか

ベンチマークをとるとき、何について実行するか決める必要があります。つまり、どんな情報が知りたいかです。

まずは、CPUやディスクといった各パーツのベンチマークをとります。これによって各パーツの単体性能を測定できます。

次に、ミドルウェアをインストールして、そのミドルウェア越しにベンチマークをとります。たとえば、「Apache httpd」をインストールしてHTTPサーバとしての性能を測定したり、「MySQL」をインストールしてデータベースサーバとしての性能を測定したりします。より実際の環境に近い結果が得られます。

最後は、実際に利用するアプリケーションや完成したシステムに対して負荷をかけて、期待した性能が得られているか、どこがボトルネックになっているか確認します。

実際に利用するアプリケーションが始めからあれば、手間をかけず求める性能値を測定できますが、サーバ選定時にアプリケーションが完成していないこともよくあります。そこで、各パーツ単体やミドルウェアを含んだ構成に対してベンチマークをとり、目的に近い結果や応用が効く結果を得ておくと良いでしょう。

また、ボトルネックの原因を掘り下げていく

には、単体での各パーツの性能や特性を知っておく必要があります。たとえばデータベースサーバは、CPUとディスクの性能に依存する部分が大きいです。そこで、今回は、CPU、ディスク、メモリ、ネットワークインターフェースへのベンチマークをとってみましょう。

ベンチマーク! その前に気を付けること

ベンチマークをする前に、注意することがいくつかあります。そうしないと正しい結果が得られません。

目的を決める

まずはベンチマークの目的を決めます。つまり、どんな結果や情報がほしいのか定めます。

たとえば、「サーバをリプレイスするにあたって、古いサーバと新しいサーバで、性能が何パーセント向上したか確認する」「新規にサーバを導入するにあたり、サーバメーカーやCPUのモデルを選定する」といった機器の性能を比較する場合は、複数のサーバやCPUでベンチマークをとり、それらの結果を相対比較します。つまり、「このサーバAはサーバBより性能が何パーセント高い」といった結果が出せるように項目を事前に洗い出し、ベンチマークをとります。

ほかの例として、「消費電力あたりの性能値を得る」「1ラック4kVA以下で最大性能を出す」といった電力量を考える必要がある場合は、ベンチマークの結果だけでなく、そのときの消費電力も測定しておく必要があります。

また実際に利用するアプリケーションを動かす事前のテストということであれば、可能な限り本番環境に近づけたOSやミドルウェア環境を用意すると良いでしょう。ベンチマークツールも、実際に利用するアプリケーションと挙動が似ているものが使えれば、より本番に近い結果が得られます。

以上のように、やみくもにベンチマークをとるのではなく、事前に計画することで、測定時間の短縮や測定項目の漏れを防ぐと良いでしょう。

BIOSの設定を確認

サーバのBIOS設定も、ベンチマークをとる前に見直しておきましょう。

とくに省電力に関する項目は性能に大きく影響します。「消費電力あたりの性能最大」「最大性能」「消費電力優先」といった選択肢があるかもしれません(**図1**)。本来の最大性能を計測したい場合は、「最大性能」を選びましょう。この省電力設定のデフォルト値は、サーバメーカーによって異なるので必ず確認しましょう。

環境は同じに

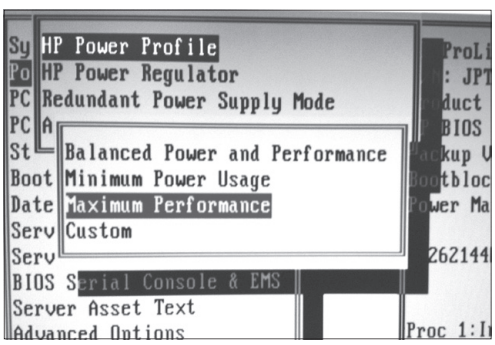
複数の対象にベンチマークをとる場合、それぞれ同じ環境にしましょう。つまり、ベンチマークツール、OS、コンパイラ(GCCなど)のバージョンや設定は同じにして、これらによる差が結果に出ないようにします。

また、ベンチマークの途中に、余計な負荷がかからないようにもしましょう。LinuxなどUNIX系OSであれば、psコマンドやtopコマンドを実行して、ベンチマークに不要で負荷をかける恐れがあるプロセスは停止し、結果に影響しないようにします。

ボトルネックがないか

ベンチマークツールを実行し、本番のデータを取得する前に何度かテストとして実行します。

▼図1　BIOS設定の参考例



このとき、想定外のところにボトルネックがないか、想定したところに負荷がかかっているか確認しましょう。たとえば、ベンチマークツール自体のCPU負荷が大きいとか、キャッシュにヒットしてしまいディスクまでI/Oが到達していないといったことがあるかもしれません。また、クライアント／サーバ型アプリケーションにベンチマークをかける場合は、クライアント側がボトルネックになることもよくあります。

また、物理的なボトルネックにも注意しましょう。たとえば、ネットワークのベンチマークで1Gbps接続している場合は、もちろん1Gbpsまでしか出ないです。同様に、ディスクのSATAやSASのインターフェースやPCI Expressにも転送速度の限界があります。ここがボトルネックになるのはまれですが、ベンチマーク対象の物理構成は把握しておきましょう。

仮想化していると難しい

仮想マシンのベンチマークは難しいです。ベンチマークツールの基本動作として、ある一定の処理を実行し、それにどのくらい時間がかかったか、またはこの時間と処理した数をもとに単位時間（たとえば1秒間）あたりどのくらい処理できたかという測定結果が出てきます。つまり、何秒動いたかの時間が重要です。一方で、仮想化環境での時刻は不正確なことが多いです。仮想マシンは常にCPUの上で動いているわけではないですし、クロックソースがエミュレートされている場合もあります。ハイパーバイザーの種類やバージョンによって正確さは異なり、CPUの負荷状況にも影響されます。ほかの仮想マシンからの負荷も影響するかもしれません。

対策としては、クライアント／サーバ型のベンチマークツールを選択することが挙げられます。クライアントを物理マシン、サーバ側を仮想マシンとすることで、仮想環境による影響は受けなくなります。

しかし、いつも利用しているベンチマークツールを使いたい場合もあると思います。この場合はしっかり事前のテストをして、計測した時間に狂いがないかを確認しましょう。単位時間あたりの結果ではなく、処理した時間が結果として出てくるツールですと、狂っているかどうかの確認はしやすいと思います。長い時間動かして誤差を少なくすると効果的です。昔は、時間のずれが大き過ぎてストップウォッチ片手に確認するなんてアナログなこともしました。

以上のとおり、仮想化環境でのベンチマークは、事前のテストを入念に行い、出てきた結果が正しいかの確認もしっかり行いましょう。

CPUをベンチ!

ここからは実践編です。CPUのベンチマークをとってみましょう。なお、以降のベンチマークの結果は**表1**のサーバにて実行した結果です。

コア数？動作周波数？

CPUの性能を見るとき、コア数と動作周波数（GHz）のどちらを見ているでしょうか。基本的にすべてのソフトウェアは動作周波数が高ければ高いほど、高速に動作します。一方、コア数は多ければ高速になりますが、コア数に応じた速度の向上は、ソフトウェア側の並列数に依存します。たとえば、8コアのCPU上で4つのプロセスしか動かないのであれば、CPUの半分の性能しか使いきれません。具体的には「Apache httpdでプロセスをたくさん生成する」「1つのサーバ上で複数のソフトウェアを同時

▼表1　マシンスペック

機種	HP ProLiant DL360p Gen8
CPU	Intel Xeon E5-2670（2.6GHz/8コア）2個
Memory	PC3-10600R 8GB 4本
Disk	SAS 15000rpm 4本のRAID10、 RAIDキャッシュ1GB
NIC	Intel X520-DA2 10Gbps
OS	CentOS 6.4 x86_64（最小インストール） GCCはOS標準パッケージ利用

に動かす」「マルチスレッドで処理されるアプリケーション」といった場合はコア数のメリットを享受しやすいです。

ベンチマークのときも、並列数に注意する必要があります。並列でないときは、1コアあたりの性能を測定することになります。マルチコアに対して測定したい場合は、並列数を増やします。

マルチコアに対して並列にベンチマークする際にもう1つ注意する点があります。IntelのHyper-Threading Technology（以下、HTT）のようなハードウェアマルチスレッディングです。HTTは1つの物理コアを2つに見せかけるため、コア数は倍になりますが、性能は倍になるわけではなく、10〜30%程度の性能向上となります。ソフトウェアしだいでその性能向上率も異なり、同様にベンチマークツールによっても異なってきます。ベンチマーク結果を見るときは、HTTが有効になっているか無効になっているかは必要な情報となります。

以上のとおり、CPUのベンチマークするときは、1コアあたりの性能なのか、マルチコアの性能なのか注意しましょう。

CPU性能値の単位は？

CPUの性能を表す基本的な単位として、MIPS（Million Instructions Per Second）とFLOPS（FLoating-point Operations Per Second）があります。その名前のとおりで、MIPSは1秒間に実行できる命令数（100万）。FLOPSは1秒間に実行できる浮動小数点演算です。浮動小数点演算は、HPC（High Performance Computing）ではよく使われますが、一般的なサーバ用途では重要になることは少ないと思います。

では、簡単にMIPSの値を見てみましょう。次のコマンドを実行してみてください。

```
$ cat /proc/cpuinfo
〜略〜
bogomips    : 5187.33
〜略〜
```

コアごとにBogoMips値が出てきます。値が大きいほど速いということになります。動作周波数が上がればこの値も大きくなります。ただし、Bogoはbogus（にせもの）という名のとおり、処理内容は極単純な命令の繰り返しであり、CPUの性能を示すには不十分です。

そこで定番のツールを使いベンチマークしていきましょう。

UnixBenchでベンチ！

UnixBench[注1]は、昔からよく使われてきたベンチマークツールです。Dhrystone[注2]やWhetstone[注3]といった伝統的なCPUのベンチマークツール、ファイルコピーやシェルスクリプトといったよく使う操作、プロセス生成やシステムコールのオーバーヘッドといったOSの動作に関わる部分などの測定を実施します。UnixBenchは単純にCPUのみをベンチマークするわけではないのです。

◆UnixBenchをビルド、実行

図2のとおり、ビルドに必要なパッケージをインストールします。そして、UnixBenchのソースコードをダウンロードし展開します。UnixBenchを実行すると、ビルドが始まり、完了後そのままベンチマークが実行されます。それぞれの項目に対して、時間がかかるテストは3回、短い時間のものは10回実行されます。始めに1並列で実行され、その次にCPUの論理コア数分の並列数で実行されます。全部の測定が終わるまでに1時間程度かかります。

もし、測定対象のサーバの論理コア数が16個よりも多ければ、並列でのベンチマークは走りません。実行したい場合は、UnixBenchのサイトのIssuesにある「Can't do default run completely with > 16 CPUs」というスレッドを見てください。

注1）http://code.google.com/p/byte-unixbench/

注2）Reinhold P. Weickerにより1984年に作成された浮動小数点演算を含まない整数演算や文字列演算向けのベンチマークプログラム。

注3）イギリス国立物理学研究所で作成された浮動小数点演算を主とするベンチマークプログラム。

パッチとその適用方法が書かれています。

◆ UnixBenchの結果

測定結果は、標準出力されるのと併せて、resultsディレクトリの中にテキストとHTMLにて保存されます。結果は、並列なし（1並列）での結果、次に論理コア数分の並列実行での結果です。

筆者が実行した32並列の結果は**図3**のとおりです。最終的な結果は、System Benchmarks Index Scoreの値で8613.1です。各テストの結果は、Sun SPARCstation 20 SM61という1995年頃のコンピューターでの値を10として基準にしたときの値です。

システム全体の性能を見たい場合は、このScoreの値を利用すると良いです。たとえば整数演算のCPU性能の値を見たいときは、Dhrystoneの結果単体を見ることもできます。

今回の結果は、8コアの物理CPUが2個、HTTを有効としており、最新の世代のCPUであるため、高速な結果が得られています。

ちなみに、HTTを無効にするとScoreは7412.0で、HTTを有効にすると2割弱高速になりました。

▼図2　UnixBenchのビルドと実行

```
$ sudo yum -y install perl perl-Time-HiRes make gcc   ←パッケージインストール
$ curl http://byte-unixbench.googlecode.com/files/UnixBench5.1.3.tgz | tar zx   ←ツールのダウンロード
$ cd UnixBench
$ ./Run   ←ビルドとベンチマーク実行
```

▼図3　UnixBenchの結果

```
32 CPUs in system; running 32 parallel copies of tests

Dhrystone 2 using register variables      579640366.1 lps   (10.0 s, 7 samples)
Double-Precision Whetstone                    57707.7 MWIPS (10.0 s, 7 samples)
Execl Throughput                              50979.7 lps   (29.9 s, 2 samples)
File Copy 1024 bufsize 2000 maxblocks        583796.7 KBps  (30.0 s, 2 samples)
File Copy 256 bufsize 500 maxblocks          167663.0 KBps  (30.0 s, 2 samples)
File Copy 4096 bufsize 8000 maxblocks       1829440.9 KBps  (30.0 s, 2 samples)   ←各テスト項目の結果
Pipe Throughput                            27739873.2 lps   (10.0 s, 7 samples)
Pipe-based Context Switching                6562696.1 lps   (10.0 s, 7 samples)
Process Creation                              95763.8 lps   (30.0 s, 2 samples)
Shell Scripts (1 concurrent)                  99406.3 lpm   (60.0 s, 2 samples)
Shell Scripts (8 concurrent)                  14489.4 lpm   (60.1 s, 2 samples)
System Call Overhead                        5456787.3 lps   (10.0 s, 7 samples)

System Benchmarks Index Values               BASELINE       RESULT    INDEX
Dhrystone 2 using register variables         116700.0  579640366.1  49669.3
Double-Precision Whetstone                       55.0      57707.7  10492.3
Execl Throughput                                 43.0      50979.7  11855.7
File Copy 1024 bufsize 2000 maxblocks          3960.0     583796.7   1474.2
File Copy 256 bufsize 500 maxblocks            1655.0     167663.0   1013.1
File Copy 4096 bufsize 8000 maxblocks          5800.0    1829440.9   3154.2
Pipe Throughput                               12440.0   27739873.2  22298.9
Pipe-based Context Switching                   4000.0    6562696.1  16406.7
Process Creation                                126.0      95763.8   7600.3
Shell Scripts (1 concurrent)                     42.4      99406.3  23444.9
Shell Scripts (8 concurrent)                      6.0      14489.4  24149.0
System Call Overhead                          15000.0    5456787.3   3637.9
                                                                   ========
System Benchmarks Index Score                                       8613.1   ←最終的な結果
```

ディスクをベンチ！

ディスクやディスクを束ねて構成した、RAIDアレイまたは外部ストレージにも使えるディスクI/Oを測定するベンチマークを行います。

ディスクI/Oはどうやってみるか

ベンチマーク対象はいくつかの構成があると思います。ディスク単体の場合、SATAのディスクかSASのディスクか、また回転数（RPM）も速度に影響してきます。RAIDを構成したとき、ディスクの本数やRAIDレベルも速度に影響してきます。もちろんRAIDカードも性能にかかわってきます。外部ストレージも同様に、ディスクの本数やRAIDレベルはありますし、コントローラの性能も影響してきます。

いずれの構成も、OS上にデバイスとして認識している必要があります。ベンチマークツールによっては、デバイスとして認識しているだけで測定できるものもありますが、一般的には何らかのファイルシステムでフォーマットし、マウントしておく必要があります。このファイルシステムもディスクI/Oの速度に影響します。

ディスクI/Oは4パターンに分けられます。

・読み込み（Read）か書き込み（Write）か
・シーケンシャルかランダムか

の組み合わせです。読み込みや書き込みはそのとおりですので良いでしょう。具体例で話すと、大きなサイズのファイルを扱うときや大量のログを吐くときはシーケンシャルI/Oを重視します。一方で、大量の小さなサイズのファイルを扱うときやデータベースはランダムI/Oを重視します。

◆ディスクI/Oの指標

ディスクI/Oを見る指標としては、転送量、IOPS（Input/Output Per Second）、レイテンシあたりを見ます。

転送量（帯域幅）は、1秒間に転送したデータ量で単位はByte/sec（B/s）となります。シーケンシャルI/Oの場合は、転送量がボトルネックになりやすいのでこの値を見るとわかりやすいです。たとえば、iSCSIやNASといった外部ストレージを1Gbpsで接続している場合、Byteに換算すると125MB/sとなるので、物理的なボトルネックが帯域幅にあり、その上限値がわかります。

IOPSは言葉のとおり、1秒間に処理したI/O数となります。ランダムI/Oの場合、ディスク性能自体がボトルネックになりやすいので、この値を見るとわかりやすいです。

通常は上記2つで十分ですが、性能問題が起きたときに調査するうえで、レイテンシまで見るとなお良いです。また、I/Oの処理量よりもミリ秒単位で処理時間短縮が必要な場合はレイテンシまで見る必要があります。レイテンシとは、その名前のとおり遅延時間です。つまり、I/Oをリクエストしてから処理を完了し、返ってくるまでの時間です。

fioでベンチ！

fio[注4]は、強力なディスクI/Oのベンチマークツールです。とても多機能でオプションが数多くあり一見扱うのが難しいです。そこで、今回は手軽にfioを使ってみましょう。Windowsでは定番のディスクI/OベンチマークツールとしてCrystalDiskMarkがあります。このCrystalDiskMarkの測定項目に近いジョブファイルがWinKey氏のサイト[注5]で公開されていますので今回はそちらを利用させていただきました。

◆fioをビルド、実行

図4のとおり、ビルドに必要なパッケージをインストールします。そして、fioのソースコー

注4） http://freecode.com/projects/fio
注5） http://www.winkey.jp/

ドをダウンロードし、展開します。ビルドし、インストールすると完了です。

fioを実行するとき、2種類の方法があります。1つは、fioコマンドの実行オプションにずらずらと書く。もう1つは、実行したいオプションや複数のテストをジョブファイルとして作成しておき、fioコマンドの実行時にそのジョブファイルを指定する方法です。今回は、ジョブファイルを先ほど紹介したWinKey氏のサイトよりダウンロードし、ジョブファイルを指定してfioを実行します。

また、「--output」オプションを付けて実行しています。このオプションを指定すると、結果が指定したファイルにテキスト出力されます。複数のテストを実行すると結果も多くなりますので、ファイルに書き出すと良いでしょう。

◆ジョブファイルの大事なオプション

実行結果を見る前に、ジョブファイル内のオプションについて、重要なところを説明します。ほかのディスクI/Oを測定するベンチマークツールでも注意すべき点です。

ダウンロードしたジョブファイルを見てみます(**図5**)。

「size=1g」は、ベンチマークで利用するデータサイズです。RAIDカードや外部ストレージの場合、キャッシュメモリを搭載していることがあります。最近のディスクですとキャッシュメモリが数GB搭載されていることもあるので、1g(1GB)だとほとんどがキャッシュに載ってしまい、ディスク自体の性能が測定できないかもしれません。たとえば、キャッシュメモリが1GB搭載されているとき、全部キャッシュに載ったときの性能を測定したい場合は512m(512MB)とし、キャッシュに載りきらずディスク自体の性能を測定したい場合は10g(10GB)と指定します。今回は、10gに変更し、測定しました。

「direct=1」は、ディスクに直接アクセスします。ディスクI/OはOS上のメモリにもキャッシュされるため、このキャッシュを使用しないI/Oでベンチマークされます。ディスクの性能を評価したい場合は1を指定して有効化しましょう。

▼図4　fioのビルドと実行

```
$ sudo yum -y install libaio-devel make gcc   ←パッケージのインストール
$ curl http://brick.kernel.dk/snaps/fio-2.0.15.tar.bz2 | tar jx   ←ツールのダウンロード
$ cd fio-2.0.15
$ make
$ sudo make install   ←ビルド
                                                    ↓ジョブファイルのダウンロード
$ curl -o crystaldiskmark.fio http://www.winkey.jp/downloads/visit.php/fio-crystaldiskmark
$ fio --output=./fio_result.txt ./crystaldiskmark.fio   ←ベンチマーク実行
```

▼図5　ジョブファイル

```
[global]
ioengine=libaio
iodepth=1
size=1g
direct=1
runtime=60
directory=/tmp/
filename=fio-diskmark
~以下テスト項目~
```

◆ fioの結果

測定結果は、実行時に指定したファイルに保存されます。筆者が実行した結果の抜粋は**図6**のとおりです。

転送量(帯域幅)を見るときは、Seq-ReadやSeq-Writeのテスト結果を確認します。このbwの値が結果となるので、シーケンシャルRead時の転送量の結果は、381536KB/sとなります。

IOPSを見るときは、Rand-Write-4K-QD32やRand-Write-4K-QD32のテスト結果を確認します。このiopsの値が結果となるので、ランダムWrite時のIOPSは、2452IOPSとなります。

今回の値は、1GBのキャッシュに、15000rpmのSASディスク4本でのRAID10のため、ローカルディスクへのベンチマーク結果としては、高速な結果です。

fioにはたくさんのオプションがあります。今回のジョブファイルを参考に、変更してみてはいかがでしょうか。

メモリをベンチ!

メモリは容量だけ見ておけば大丈夫か

利用するサーバスペックを決めるとき、メモリは容量で選ぶことが多いと思います。メモリを速度で選ぶことはまれで、メモリの速度をベンチマークする機会も少ないかもしれません。というのもメモリは高速でその速度がボトルネックになる前にほかの個所がボトルネックになりやすいからです。そのため、メモリの速度によるアプリケーションの速度への影響は小さいのです。

メモリの速度は、動作周波数に依存します。たとえば、PC3-12800は1600MHzで動作します。この値が大きいほど速いのですが、チャネルも考慮する必要があります。つまり、メモリをどこのスロットに挿すか、そしてその枚数にも注意が必要です。これはサーバの世代によっても異なってくるため、購入前にサーバメーカーのカタログやドキュメントを確認すると良いでしょう。メモリのベンチマークをとり、チャネルが正しく構成されているか確認するのも良いかもしれません。

SysBenchでベンチ!

SysBench[注6]は、メモリだけでなく、CPUを始めOLTP(データベース)など多くの対象を測定可能なベンチマークです。

◆ SysBenchをビルド、実行

図7のとおり、ビルドに必要なパッケージを

注6) http://sysbench.sourceforge.net/

▼図6 fioの結果

```
~略~
Seq-Read: (groupid=0, jobs=1): err= 0: pid=4286: Fri May 17 04:34:23 2013
  read : io=10240MB, bw=381536KB/s, iops=372 , runt= 27483msec
    slat (usec): min=58 , max=458 , avg=61.34, stdev= 6.55
    clat (usec): min=165 , max=218217 , avg=2621.01, stdev=4375.96
     lat (usec): min=231 , max=218287 , avg=2682.53, stdev=4376.68
~略~
Rand-Write-4K-QD32: (groupid=7, jobs=1): err= 0: pid=4296: Fri May 17 04:34:23 2013
  write: io=588888KB, bw=9811.3KB/s, iops=2452 , runt= 60022msec
    slat (usec): min=4 , max=17111 , avg= 9.92, stdev=103.52
    clat (usec): min=1 , max=40593 , avg=13034.62, stdev=6690.25
     lat (usec): min=24 , max=40601 , avg=13044.69, stdev=6689.90
~略~
```

インストールします。そして、SysBenchのソースコードをダウンロードし、展開します。ビルドし、インストールすると完了です。なお、ここではOLTPのベンチマークは実施しないため、configure時に「--without-mysql」オプションを付けています。もし、MySQLへのOLTPベンチマークも実施する場合は、MySQLパッケージをインストール後「--without-mysql」を外して実行してください。

メモリI/Oもディスク I/O同様に、読み込み(Read)か書き込み(Write)か、そしてシーケンシャルかランダムかの4パターンに分けられます。メモリはランダムアクセスに強いため、シーケンシャルかランダムかを意識する必要は少ないです。

実行時のオプションとして、メモリのテストの場合「--test=memory」と指定します。読み込みか書き込みかは「--memory-oper」でreadまたはwriteと指定します。「--memory-access-mode」でseq(シーケンシャル)またはrnd(ランダム)を指定します。

◆ SysBenchの結果

測定結果は標準出力されます。筆者の環境で実行したランダムWriteでの結果は**図8**のとおりです。最終的な結果は転送量として得られ、ここでは3591.55MB/secとなります。

今回のメモリ構成では、プロセッサあたり4

▼図7　sysbenchのビルドと実行

```
$ sudo yum -y install libtool make gcc   ←パッケージのダウンロード
$ curl -L http://downloads.sourceforge.net/project/sysbench/sysbench/0.4.12/sysbench-0.4.12.tar.gz | tar zx   ←ツールのダウンロード
$ cd sysbench-0.4.12
$ libtoolize --force --copy
$ ./autogen.sh
$ ./configure --without-mysql   ←ビルド
$ make
$ sudo make install
$ sysbench --test=memory --num-threads=1 --memory-oper=read --memory-access-mode=seq run   ←ベンチマーク実行
$ sysbench --test=memory --num-threads=1 --memory-oper=write --memory-access-mode=seq run
$ sysbench --test=memory --num-threads=1 --memory-oper=read --memory-access-mode=rnd run
$ sysbench --test=memory --num-threads=1 --memory-oper=write --memory-access-mode=rnd run
```

▼図8　sysbenchの結果

```
Operations performed: 104857600 (3677744.60 ops/sec)

102400.00 MB transferred (3591.55 MB/sec)   ←結果

Test execution summary:
    total time:                          28.5114s
    total number of events:              104857600
    total time taken by event execution: 20.8725
    per-request statistics:
         min:                                  0.00ms
         avg:                                  0.00ms
         max:                                  0.30ms
         approx.  95 percentile:               0.00ms

Threads fairness:
    events (avg/stddev):           104857600.0000/0.00
    execution time (avg/stddev):   20.8725/0.00
```

チャネルあり、そのうちの2チャネルのみ使用していますので理想的な構成ではありません。動作周波数は1333MHzです。

SysBenchはメモリ以外にも多くのベンチマークができますので、CPUなどほかも試してみてはいかがでしょうか。

ネットワークをベンチ!

サーバからネットワークを見ると、最初に出てくるのはネットワークインターフェースつまりNIC(Network Interface Card)です。

NICのベンチマークは必要か

サーバのNIC(Network Interface Card)は1000BASE-Tつまり1Gbpsが当たり前になりました。最近のサーバ向けのNICではどんなモデルを選んでもたいてい1Gbps近い速度が出ることでしょう。そのため、NIC自体のベンチマークは不要かもしれません。一方で、Flashストレージをはじめ周辺のデバイスが高速化してきたことや、価格が低下してきたこともあり10Gbpsが選択肢に上がることが今後増えてくることでしょう。

ここでは、10GbpsのNICについてベンチマークをとります。今回の方法は、NICだけでなく、構成したネットワークに対しても有効です。たとえば、L2スイッチやL3スイッチで構成したLANの中で測定したり、インターネット越しに測定したりすることもできます。

netperfでベンチ!

netperf[注7]は、強力なネットワークのベンチマークツールです。とても多機能でオプションが数多くあり、一見扱うのが難しいです。しかし、転送量(帯域幅)を測定するだけであればデフォルトオプションで簡単に測定できます。

◆ネットワーク構成

netperfはクライアント/サーバ型のネットワークツールです。つまり、ベンチマークするには2台のマシンが必要になります。今回の構成は**図9**のとおりで、Intelの10Gbps NICであるX520-DA2を搭載したサーバ2台をDACで直結しています。今回はSFP+の10Gbps NICのためDACで接続していますが、もし1000BASE-Tや10GBASE-Tでのテストで直結する場合は、ストレートのツイストペアケーブルつまり、普通のLANケーブルで接続できます(最近のNICはクロスケーブルでなくても直結可能です)。

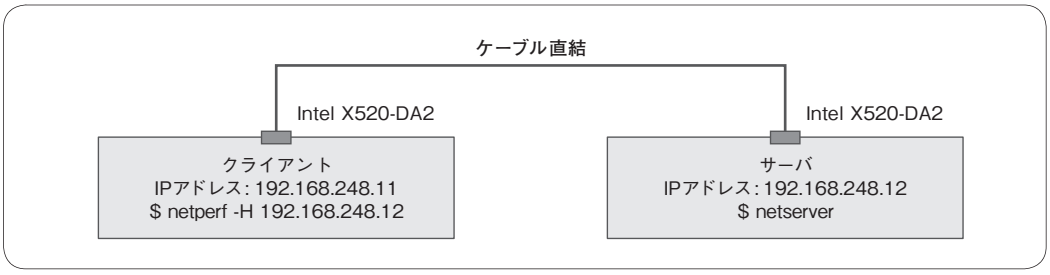
また、今回は直結ですが、もちろん間にスイッチが入っても測定できます。

◆netperfをビルド、実行

図10のとおり、ビルドに必要なパッケージをインストールします。そして、netperfのソースコードをダウンロードし展開します。ビルド

注7) http://www.netperf.org/netperf/

▼図9 ネットワーク構成図



しインストールすると完了です。インストールまでは2台のマシンとも完了してください。

まず、サーバとなる側で、netserverコマンドを実行してください。デフォルトで12865番ポートを使うので、もしiptablesなどでフィルタリングしている場合は、そのポートを開けるか、iptablesを一時的に止めてください。

サーバ側が準備できたら、クライアント側で実行しベンチマークを開始します。デフォルトでは、10秒間TCP通信して測定されます。

◆ netperfの結果

測定結果は標準出力されます。

筆者が実行した結果は**図11**のとおりです。最終的な結果はThroughputの項目で単位は10^6 bit/secつまりMbpsということになります。そのためここでの結果は9.4Gbpsということになります。10Gbps接続ですので、ほぼ物理限界に近い値が出ているとわかります。10Gbps NICは本当に10Gbps出るのですね。

netperfには豊富なオプションがありますので、いろいろ試してみてはいかがでしょうか。

おわりに

以上で、パーツ単体のベンチマークはとれたと思います。大事なことは得られた結果が正しいかの確認です。もし想定より良かったり悪かったりしたときに、なぜそうなったか調査してみましょう。どこかにボトルネックがあったり、どこかでキャッシュが効いていたりするかもしれません。こういった経験からサーバの構成や挙動の理解を深めることにつながり、最終的に安定したサーバ運用ができるようになります。SD

▼図10　netpefのビルドと実行

```
・サーバーとクライアント両方で実行
$ sudo yum -y install make gcc   ←パッケージのダウンロード
$ curl ftp://ftp.netperf.org/netperf/netperf-2.6.0.tar.bz2 | tar jx   ←ツールのダウンロード
$ cd netperf-2.6.0
$ ./configure
$ make                            ←ビルド
$ sudo make install

・サーバー側で実行
$ netserver

・クライアント側で実行
$ netperf -H <サーバーのIPアドレス>

・サーバー側の終了
$ killall netserver
```

▼図11　netperfの結果

```
MIGRATED TCP STREAM TEST from 0.0.0.0 (0.0.0.0) port 0 AF_INET to 192.168.248.12 () port 0 AF_INET
Recv   Send    Send
Socket Socket  Message  Elapsed
Size   Size    Size     Time     Throughput
bytes  bytes   bytes    secs.    10^6bits/sec

 87380  16384  16384    10.00    9410.82
```

分散データベース「未来工房」新連載

第1回 RiakとNginxだけを使って作る静的ファイルホスティング
――その構成とベンチマーク結果

Writer 上西 康太(うえにし こうた)　Bashoジャパン株式会社　kota@basho.com

更新が多くないからといって、静的ファイルホスティングをスケールさせて、巨大なトラフィックを捌くのは容易ではない。負荷分散、ディスクの冗長化、キャパシティプランニング、スケールアウト、スケールダウン、エンジニアが考えなければならない問題は多岐にわたる。ここでは、GitHub pagesでも実績があるRiakを使った方法をもとに、安定したレイテンシで、高速でスケールする静的ファイルホスティングサイトをノンプログラミングで構築する方法を紹介する。

注)本稿は、筆者の意向により常体で表記している。

概要

はじめてWebサーバを立ち上げたとき、だれしもが最初にApache httpdをインストールし、helloと書いたindex.htmlを指定のフォルダに置いて、ブラウザで開いた記憶があるだろう。もしくは、GeocitiesにHTMLファイルをFTPクライアントでアップロードした記憶のある読者も多いだろう。

静的ファイルホスティングはWebサイトを作るときの基本だ。一見簡単なようで案外奥が深い。たとえば、LinuxのISOイメージをHTTPでダウンロードするために一晩待ったり、いざtarball[注1]をダウンロードしようとしたらディスク障害や過負荷でサーバが落ちていたという読者も多いだろう。実際、Webサイトのホスティングは負荷分散、データの冗長化、キャパシティプランニング、スケールアウト、ネットワークコストなど多くの技術を必要とする。

実はこれらの問題は、ネットワークコストの節約を除いて、たった2つのソフトウェアで解決することができる。Nginx[1]とRiak[2]だ。Nginxで負荷分散をし、データの冗長化、キャパシティプランニング、スケールアウトはすべてRiakで実現できる。RiakはただのKVS(Key-value Store)と思われがちでRDBと比較されてしまうが、インターフェースがHTTPであること、データをレコード単位で冗長化していること、マシンを追加すればスケールアウトできること、データベースにしてはある程度大きなレコードも扱えることなどから、KB～MBオーダーの小さなものをターゲットとしたオブジェクトストレージとしても応用できる。

ここでは、GitHub pages[3]でも実績があるRiakを使った方法をもとに、安定したレイテンシで高速でスケールアウトできる静的ファイルホスティングサイトをノンプログラミングで構築する方法を紹介する。

注1) tarコマンドで固めたファイルのこと。エンジニアスラング。

サイトの設計

静的ファイルホスティングのサイトをfoo.example.comでホストすることとしよう。このサイトでやりたいことは、

1. 複数の静的ファイルをホストする
2. 個々の静的ファイルはfoo.example.com/path/to/the/fileなどのURLによって一意に

特定される
3. 静的ファイルのサイズはたかだか数MB程度、平均で数百KB
4. 静的ファイルの個数は1,000～1000,000,000だが、最初はミニマルなスタートをしたい
5. すべての画像に、均等かつランダムにReadアクセスが来る
6. とくに画像を消すことはしない（簡単のため）
7. レスポンスタイム目標は、99%が500ms以内に収まること
8. サービス停止時間は0を目指す

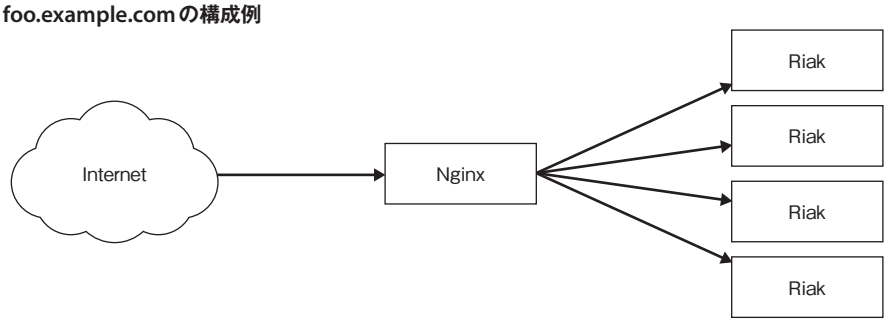
である。とくに4.のスケールアップ、8.の停止時間は通常のファイルサーバとApacheを使ってHA（High Availabilty）構成をするだけでは難しい。Nginxなどのロードバランサを導入したとしても、シャーディングを自分で設計・運用なければならないし、8.の無停止運用はディスク故障・冗長化の構築やロードバランサの振り分けの設定が複雑になる。そこで、そういった役割を、もともと機能として実装しているRiakに任せることでシステム全体を単純化できる。

Riakは2006年に発表されたAmazonのDynamoを参考に設計された[4]。おもな特徴として、Consistent Hashingによる水平分散・スケールアウト、レプリケーションによる冗長構成、安定したレスポンスによる高可用性などがある。ほかにも、JavaScriptで実行できるMapReduceにより、joinを除いてSQLとほぼ同等のクエリ表現力がある。

RiakはHTTPインターフェースを持っており、Riakに登録されているデータをそのままブラウザで表示できる。したがって、そのままでもインターネット上に公開すればRiak上にアクセスすることは可能になるが、HTTPメソッドのうちGETだけでなくPUTやDELETEもとくに認証なく実行できてしまう。不特定ユーザによるデータの削除や更新を防ぐために、まず、Riakの通常の運用として、NAT（Network Address Translation）などを用いてインターネットから直接はアクセスできないネットワーク構成にしておくことが必要だ。

たいていのロードバランサは特定のHTTPメソッドだけを処理し、他を拒否するような設定が可能であるので、読者が慣れているもの

▼図1　foo.example.comの構成例



Note

クラウドサービスを用いる場合は、それぞれの環境でアプローチが異なる。AWSのEC2であれば、Marketplaceにあるamiを利用するのがよい。EC2ではすべてのノードにグローバルIPアドレスが割り振られてしまうので、IPSecなどでアクセスが最初から制限された公式amiがよい。IDCフロンティアのクラウドセルフタイプ公式APIや、VPC(Virtual Private Cloud)などを使うのがよい。、NTTコミュニケーションズのCloud nやNiftyのクラウドなど、CloudStackベースのものであれば仮想マシンがあらかじめプライベートなネットワークに隔離されている。

があればそちらを利用してもよい。ここでは、もっともポピュラーで高速なロードバランサとしてNginxを例とする。実際にはNginxの他にもフリーなもの／商用製品含めてHAProxy、Varnish、LVSやBigIPなどさまざまなロードバランサがあり、ここでのものとほぼ同様のことができるので、読者の慣れたものを利用してもらうのがよい。

Nginxには、GETだけを許可し、PUT、DELETEなど他のHTTPメソッドはすべて拒否する設定にする。実際のファイルの更新は、ここでは簡単のためロードバランサを介さず、直接Riakクラスタにアクセスしてcurlを用いて行う運用を想定する。

本来であれば、これから解説するRiakクラスタに加えて、次のものが場合によっては必要になるだろう。

- ファイルアップロード用のページ(認証、アップロードのフォームなどが含まれる)
- Riakクラスタを監視するシステム(Zabbixなど)
- スケールアウトするシステム向けのデプロイシステム(Puppet, Chefなど)
- アクセスログの取得・管理と不正アクセスのカット
- CDN(Contents Delivery Network)による負荷の軽減

また、動作確認したソフトウェア環境は次のとおりである。

- Ubuntu Linux LTS 12.04 / x86_64
- Riak 1.3.1
- Nginx - 1.1.19(Ubuntuのaptのバージョン)
- curl

上記のソフトウェアが動作する環境であれば、たいていのハードウェアで動作する。ほかにもRiakはUNIX系のほとんどのプラットフォームをサポートしている。Riakが動作するハードウェアの最低条件は特にないが、たとえば

- 8コアのx86 CPU
- RAM：16～32GB
- HDD：8TB
- 1GbE NIC(Network Interface Card)

や、

- 32コアCPU
- RAM：64～128GB
- HDD：72TB
- 10GbE NIC(Network Interface Card)

といった構成がよいだろう。ディスク領域はRAID0やLVM、ZFSなどで1つのディレクトリとして扱えるようにしておく必要がある。また、今回はバックエンドにBitcaskを採用するので、静的ファイルの数が多いようであればメモリ消費量をドキュメント上で計算しておくとよいだろう[注2]。ディスク容量に関していえば、ホストしたいファイルの平均サイズと個数を見積もって、

1台あたりのディスク容量≧ファイルの平均サイズ×1台あたりのファイル数×3＋マージン(20%)

とすればよい(この数字は故障率やネットワーク構成にも依存するので一概には言えない。運用の安定性とサービスコストが常にトレードオフの関係にあることは読者もよくご存じかと思う)。実際には、ハードウェアのMTBF(Mean Time Between Failures)と MTR(Mean Time To Recovery)、ネットワーク帯域を考慮しながら設計すると最大限に性能を生かすことができるだろう。

Riakの準備と起動

ここからは、実際にサイトを構築するため

注2) http://docs.basho.com/riak/latest/references/appendices/Bitcask-Capacity-Planning/

の手順を解説していきたい。Riakが動作する最小環境[注3]は64ビットCPU、4GB RAM、複数のHDD、高速なネットワークなどである。Bashoが推奨するRiakの最低台数は5台だが、ここでは手順の確認が目的なので3台(10.0.1.11, 12, 13)での例を示す。 また、Nginx用に1台のマシン(10.0.1.10)を用意しておく。読者の環境に合わせて適宜読み替えていただきたい。

2013年の4月時点でのRiakの最新版は1.3.1だ。Ubuntu Linux 12.04またはAPTを利用できる環境なら、Bashoが提供するapt-lineからのインストールが可能[注4]だ。

```
$ sudo curl http://apt.basho.com/gpg/basho.apt.key | sudo apt-key add -
$ sudo bash -c "echo deb http://apt.basho.com $(lsb_release -sc) main > /etc/apt/sources.list.d/basho.list"
$ sudo apt-get update
$ sudo apt-get install riak
```

これでRiakのインストールは完了だ。もしインターネットにアクセスできない環境で利用するなら、Bashoのサイト[注5]からダウンロードし、**dpkg**コマンドでインストールする。

```
# sudo dpkg -i riak_1.3.1-1_amd64.deb
```

とすれば、そのマシンへのインストールは完了する。

Riak自身、ほとんどの環境ではパラメータのチューニングは必要ない(ここでは、最適なパラメータを求めて性能測定を繰り返すことをいうが、Linuxにおいては、最もよい性能を出すためにいくつか変更すべきパラメータがある。詳しくはパラメータの設定[注6]を参照いただきたい。

```
# sudo sysctl -w net.ipv4.tcp_tw_reuse=1
```

次に、プロセスがオープンできる最大のFD数を大きくするに、/etc/security/limits.confを編集する。

```
riak   hard   nprc   100000
riak   soft   nprc   100000
```

また、ディスクI/O性能に影響するため、カーネルのI/Oスケジューラをdeadlineにしておく。**sda**の部分は、**/etc/riak/app.config**で設定したデータディレクトリが載っているデバイスを指定しておくこと。

```
# sudo echo deadline > /sys/block/sda/queue/scheduler
$ cat /sys/block/sda/queue/scheduler
```

環境に合わせて設定すべきファイルは**/etc/riak/app.config**と**/etc/riak/vm.args**の2つである。ここで必要なのは、HTTPリクエストを受け付けるためにbindするIPアドレスと、Erlangの通信ライブラリが利用するIPアドレスを設定する(デフォルトでは127.0.0.1のループバックアドレスが設定されており、外部から利用できない)。

app.configは33行目(riak_coreセクションのhttpエントリ)にHTTPのIPアドレスを設定する個所があるので、これを自分のRiakサーバーのIPアドレスに設定する。

```
{http, [ {"10.0.1.11", 8098 } ]},
```

また、データディレクトリは**/var/lib/riak**になっているので、データ用のパーティションが設定されている場合はこれを変更する。デフォルトのbitcaskセクションのdata_rootエントリである。

次に、**/etc/riak/vm.args**を設定する。

注3) http://docs.basho.com/riak/1.3.1/tutorials/System-Planning/#Hardware
注4) http://docs.basho.com/riak/1.2.0/tutorials/installation/Installing-on-Debian-and-Ubuntu/
注5) http://docs.basho.com/riak/latest/downloads/
注6) http://docs.basho.com/riak/latest/cookbooks/Linux-Performance-Tuning/

```
## Name of the riak node
-name riak@10.0.1.11
+zdbbl 131072
```

Erlangの通信ライブラリにIPアドレスを明示的に設定しておくためだ。2つめの**+zdbbl**[注7]は、Erlang/OTPの通信ライブラリの送信キューのサイズ(KB単位)だ。デフォルトは1,024KBで、最大は2GB程度を設定できる。

これで設定は完了したので、すべてのサーバでRiakのプロセスを起動する。

```
[10.0.1.11]
$ sudo riak start

[10.0.1.12]
$ sudo riak start

[10.0.1.13]
$ sudo riak start
```

これら3台は起動したが、まだお互いを別のRiakクラスタと認識しているので、**cluster**コマンドにより、1つのクラスタとして認識させる。考え方としては、**図1**で示すサーバの10.0.1.11が1台で構成しているクラスタにもう2台が参加・合流するイメージである。

```
[10.0.1.12]
$ riak-admin cluster join riak@10.0.1.11

[10.0.1.13]
$ riak-admin cluster join riak@10.0.1.11

[10.0.1.11]
$ riak-admin cluster plan
$ riak-admin cluster commit
```

cluster planコマンドはvnodeといわれるデータの管理単位の配置を計画するコマンドだ。何もデータが入っていない状態で最初にクラスタを構築するときはとくに必要性を感じないが、ある程度データがたまって、システムを止められない状態になってから再配置をコントロールするためには非常に役立つ。**cluster commit**を実行すると、planで計画された再構築を実行する。具体的には時間をかけてvnodeの移動が行われる。移動の様子は、

```
$ riak-admin transfers
```

を実行することで観察することができる。もし再配置の速度を調節したいなら、

```
$ riak-admin transfer-limit
$ riak-admin transfer-limit 1
```

などとして、再配置の帯域を確認・調節することができる。**transfer-limit**のデフォルトは2だが、これは同時にコネクションを張る数(=同時移動できるvnode)なので、全体の再配置に必要な時間と、ネットワーク帯域やマシンの負荷とのトレードオフとなる。なお、再配置中にもこの値は変更できるので**transfers**の様子やマシン負荷をみながら値を調節するとよいだろう。

これで、マシンが起動し、無事にクラスタとして動作していることを確認しよう。方法はいくつかある。まず、Riakの運用コマンドを使う方法は次のとおりだ。

```
$ curl -X PUT http://10.0.1.11:8098/buckets/spam/keys/ham -d 'egg'
$ curl http://10.0.1.12:8098/buckets/spam/keys/ham
egg
$ curl -X DELETE http://10.0.1.13:8098/buckets/spam/keys/ham
$ curl http://10.0.1.11:8098/buckets/spam/keys/ham
not found
```

Bucket名が **spam**、キー名が**ham**というレコードに、3台のマシンすべてからアクセスできることが確認できた。

注7) http://docs.basho.com/riak/latest/cookbooks/faqs/logs-faq/#riak-logs-have-busy_dist_port-messages

最初のファイルのアップロード

まず、Nginxを挟まずにサンプル画像(**図2**)をテスト用にアップロードしておこう。

```
$ curl -X PUT http://10.0.1.11:8098/buckets/static/keys/Lenna.jpg \
  -H "Content-type: image/jpeg" \
  --data-binary @Lenna.jpg
$ gnome-open http://10.0.1.11:8098/buckets/static/keys/Lenna.jpg
```

もしgnomeでないウィンドウマネージャや、他のOSを利用している場合は、URLをブラウザで開いてみよう。アップロードした画像を確認できるはずだ。

Nginxの設定

Riakの基本機能が動作することは確認できたので、次はNginxを設定する。Nginxにしてもらうことは、

1.GETのリクエストを、Riakクラスタに均等に配分する
2.PUT、POSTなどの更新系のリクエストを拒否する

の2つである。まずは、Nginx用のマシンを10.0.1.10にインストールする。

```
[10.0.1.10]
$ sudo apt-get install nginx
```

上記の設定を行うために、**/etc/nginx/site-enabled/default**を編集する。

まずHttpUpstreamModule[注8]の設定を**upstream backend**で記述することで、HTTPリクエストを配分するRiakサーバを指定する。

```
upstream backend {
      server 10.0.1.11:8098;
      server 10.0.1.12:8098;
      server 10.0.1.13:8098;
      keepalive 4096;
}
```

このように、IPアドレスとポート番号を並べていくだけでよい。また、**keepalive**[注9]は4,096本のTCPコネクションをHTTP KeepAliveで保持できるという意味である。

次に、GET以外のリクエストを通さない設定を、**limit_except**[注10]を用いて**server**セクションに記述する。サイトのFQDNはfoo.example.comとすると

```
server {
      listen   80;
      server_name foo.example.com;

      location / {
            limit_except GET {
                  deny   all;
            }
            proxy_pass http://backend;
      }
}
```

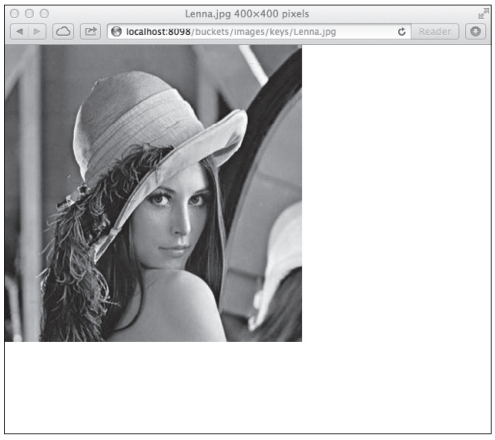
とする。上から意味は、「80番ポートをlistenする」このサーバのFQDNはfoo.exmample.

注8) http://wiki.nginx.org/HttpUpstreamModule

注9) http://nginx.org/en/docs/http/ngx_http_upstream_module.html#keepalive

注10) http://wiki.nginx.org/HttpCoreModule#limit_except

▼図2　RiakはHTTPインターフェースを持っているので、データをブラウザから参照できる



▼図3　1,048,576個の同じ画像を異なる名前でアップロードするためのRubyスクリプト

```
n = 1024 * 1024
(0..n).each do |i|
  url = "http://10.0.1.11:8098/buckets/static/keys/#{i}.jpg"
  print "#{i}/#{n} th image to #{url}\n"
  print `curl -X PUT #{url} -H "content-type: image/jpeg" --data-binary @lenna.jpg`
end
```

comで、受け付けるメソッドはGETのみで、**proxy_pass**で指定するサーバへロードバランシングしながら転送する、となる。

http://foo.example.com/buckets/static/keys/Lenna.jpgで、指定の画像にアクセスできるようになる。実際のアプリケーションであれば、もうすこし柔軟なパス指定ができるようになるとよいだろう。RiakでいうKeyやBucket名に'/'の文字を含めるようにしたい場合は、Nginx側などでいったんURLをrewriteをしてURLエンコードされた文字がBucketやキー名になるようにしておくとよい。

basho_benchによる負荷テスト、サーバ増設でスケールアウト

実際に構築した静的ファイルホスティングの性能を評価する。HTTPサーバの性能を評価するには、ab(apache bench)を利用するのが容易だが、ここではディスクキャッシュなどの効果をなるべく打ち消すためにランダムに画像にアクセスするのがよい。そこで、今回はBashoがRiakの性能を測定するために開発・提供しているbasho_benchを利用した方法を説明する。

準備

まず、静的ファイルのサンプルとなる画像をRiak上に100万枚アップロードする。

実際にすべて異なる画像である必要はないし、名称を自動で決めてよいが、ここではbasho_benchで使いやすくするため、数字の連番のファイル名、1.jpg、……1048576.jpgをRiak上に保存する。画像のサイズは25KB程度で3重にコピーされるので、合計で75GB程度のファイルがクラスタ上にアップロードされることになる。

簡単なスクリプトを用意し、これでファイルをアップロードする。言語は何でもよいが、ここではRubyを選んだ(**図3**)。

同時に、basho_benchの準備を行う。負荷をかける別のマシンを用意し、GitやErlangをインストールしておく。

```
$ git clone git://github.com/basho/basho_bench.git
$ cd basho_bench
$ make
```

これでbasho_benchという実行ファイルが作成される。basho_benchの設定ファイル(httpraw.config)は以下のようにしておく。

```
{mode, max}.
{duration, 30}.
{concurrent, 64}.
{driver, basho_bench_driver_http_raw}.

{key_generator, {int_to_str, {uniform_int, 1048576}}}.

{http_raw_ips, ["10.0.1.10"]}.
{http_raw_port, 80}.
{http_raw_path, "/buckets/static/keys"}.
{http_raw_params, ".jpg"}.

{operations, [{get, 1}]}.
```

ここでは個々の項目を細かく説明することはしないが、Nginxが入っているサーバに、**http://10.0.1.10/buckets/static/keys/{i}.jpg**というアドレス(iは1〜1048576からランダムに選ばれる)に64並列で30分間アクセスし続けると理解しておくとよい。

また、読み出しの負荷を減らすために、読み出しアクセス数を1にしておく。書き込みと読

み出しがよく競合する場合には過半数を設定しておけばよいが、想定するアクセスはWrite once、Read manyなので必要ない。

```
$ curl -X PUT http://10.0.1.11:8098/buckets/static/props \
  -H "content-type: applicaiton/json" -d '{"props":{"r":1}}'
```

これにより、**static**という名前のBucketに**r=1**が設定されたことになる。

実際に使用したハードウェア環境は、

・Intel Xeon CPU 2.40 GHz 4 Core
・RAM：8GB
・HDD：データ領域 128GB

というスペックの仮想マシンだ。

実験

basho_bench のベンチマークを実行するには

```
$ ./basho_bench httpraw.config
```

と、スクリプトを実行するだけでよい。すぐにグラフをコマンドラインで表示するためには

```
$ ./priv/gp_throughput.sh -t dumb
$ ./priv/gp_latencies.sh -t dumb
```

とする。これにより、ターミナル上にテキストでグラフが描かれる。Xフォワードなどをしていないサーバ環境上で有用なコマンドだ。

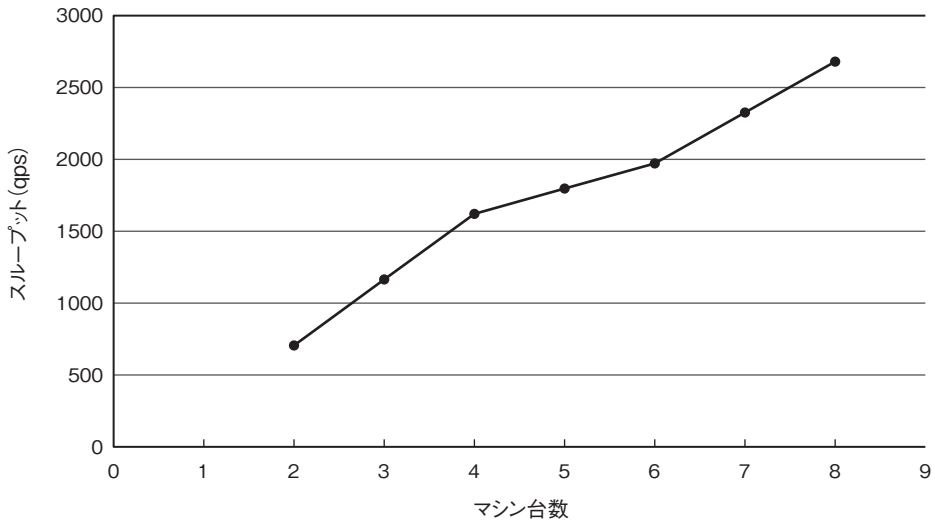
この測定を、2〜8台とRiakの動作台数を変化させながらスループットとレイテンシを測定した。Riakの動作台数が増えれば増えただけ負荷分散が可能になるため、台数に応じてリニアにスループットが向上しつつ、スループットは一定以下に収まることが期待される。 同時に、実際には、6台を超えたあたりから、1台あたりのデータ量が 物理メモリを下回るようになるため、性能はそれよりもよくなることが想像される。

結果

まず、スループットは、2台のときは700qps程度、8台のときには2,700qps程度と、マシンを増やすごとに安定してスケールアウトしているということができる(**図4**)。

実際には、lenna.pngは25KB程度のサイズなので、Nginxを通してbasho_benchは67.5MB/sのデータを得ている。

▼図4　台数を変化させたときのHTTP GETのスループット



静的コンテンツなので、これ以上のトラフィックを捌きたい場合にはやはりCDN（Contents Delivery Network）を利用するのが妥当だろう。

図5にはレイテンシを掲載しておく。異なる台数に同じ負荷をかけているので、台数が少ないときが一番レイテンシが悪いことがわかる。8台のときは99%のリクエストを400ms以内に返却していることがわかる。つまり、台数が増えるとレイテンシとスループットが向上するから、負荷分散の目的が達成できていることがわかる。

また、この実験を行う際には、Nginxの調整やKeepAliveの設定、OSの設定などに苦労し、何度もRiakのサーバをクラスタに追加したり、6台まで測ったがミスがあったので2台からやりなおすといったことを何度も繰り返した。しかしその際も、100万個のファイルのデータを消すことなく、スムーズにサーバの追加・削除を行うことができた。つまり、2、4、6、8台と測定したときは一度もRiakのシステム全体は停止していない。このことからも、Riakを無停止で運用することがいかに簡単か、おわかりいただけると思う。

GitHub Pages

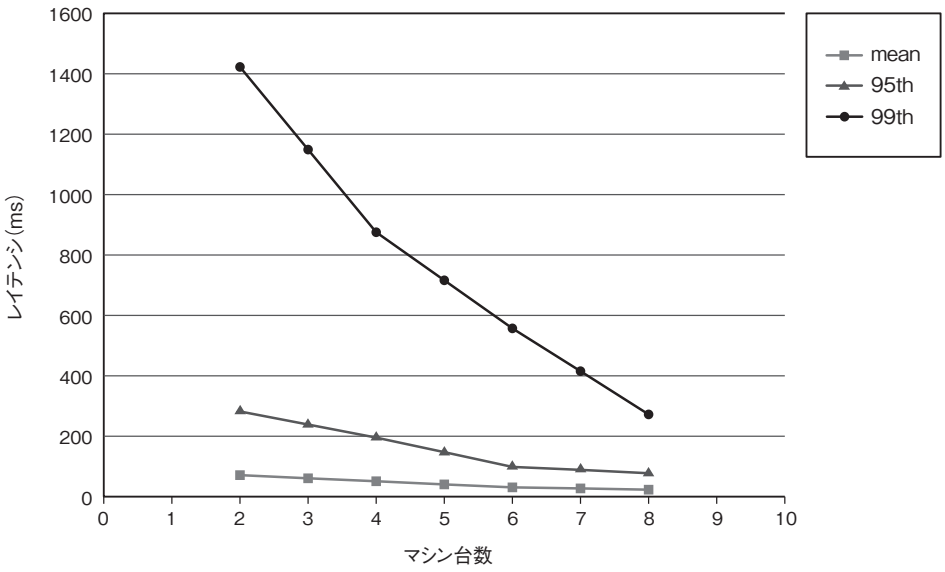
本稿では、最小限の方法だけ紹介した。実際のサービスGitHub Pagesでどのようになっているかについては、Erlang Factory 2012でのGitHubのエンジニアJesse Newlandの講演で詳しく解説されている[5]。実際にGitHub Pagesの静的コンテンツはFastlyというCDNにキャッシュされている（**図6**）。また、'/'を含むパスもあることから、Webmachineを使って簡単なHTTPサーバを実装し、Riakのデータへ1:1でのパス変換や、静的コンテンツの簡単なバージョン管理をしている。

Riakを使って、GitHub Pagesは今も高いレスポンスを維持しながらスケールアウトができている。

他のアプローチ

典型的には、Nginxそのものにも静的ファイルを配信する機能があるので、それを利用するのが一番簡単だろう。RAIDでディスクを冗長化してある程度のキャッシュを設けておけばシ

▼図5　台数を変化させたときのHTTP GETのレイテンシ



▼図6　Github PagesはCDNにFastlyを利用している

```
$ dig basho.github.io
……(省略)……
;; QUESTION SECTION:

;basho.github.io.              IN      A

;; ANSWER SECTION:

basho.github.io.       3600    IN      CNAME   github.map.fastly.net.

github.map.fastly.net.         30      IN      A       103.245.222.133
```

ンプルで高速な静的ファイル配信が可能だ。同様に、NFS上に置いたデータを複数のマシンにマウントしてApacheやNginxなどでホストしてもよい。

他にも、3月にOSS版が公開されたRiak CS[注11]を使う方法もある。こちらは Amazon S3 互換の分散ストレージで バックエンドにRiakを利用しているため、Riakと同等の耐障害性があり、S3 関連のツールを再利用することもできる。5TBまでの大きなオブジェクトもサポートされている。

まとめ

本稿では、RiakとNginxを利用してスケールアウトできる静的ファイルホスティングの構築方法を紹介した。Riakを使うことによって、量と負荷の両面からスケールアウトと負荷分散ができた。また、Riakの特徴である安定したレイテンシが実現され、Nginxの簡単な設定によって負荷分散ができた。

今回はサーバを8台しか使えていないが、50台や100台程度までなら商用で動いているクラスタがいくつかあり、スケールアウトもできている。ぜひ、より大きなクラスタにも挑戦していただきたい。また、データ量も25GB程度で固定であったが、マシン台数を増やせば性能を向上しつつトータルのデータ容量も増やすことができる。システム要件に応じてディスク容量や台数を調整するとよいだろう。

Erlang/OTPやRiak自体の設計上、読み込み性能やスループットはそこまで高くはない。しかし実際には、コストを掛けて読み込みが高速なデータベースや、オンメモリデータベースを用意するよりもCDN事業者が提供する静的コンテンツの配信サービスを利用した方が安価で高速だ。ピーク時の読み出しリクエストの処理はCDNに任せつつ、オリジナルのデータを保存し、キャッシュヒットしなかったときのためのデータストアとしてのRiakは、十分にペイするだろう。

昔のように個人でテキストや画像サイトを運営している限りはそうそう機会はないだろうが、いざ数百万個のファイルを目の前にしたときに、これまで述べたような基本的な考え方が読者の役に立てば幸いである。SD

●参考文献

[1] Nginx(http://www.nginx.org)
[2] Riak(http://github.com/basho/riak)
[3] GitHub Pages(http://pages.github.com)
[4] Giuseppe DeCandia, et al., Dynamo: amazon's highly available key-value store. In SOSP '07.
[5] Jesse Newland, Rewriting GitHub Pages with Riak Core, Riak KV, and Webmachine. In Erlang Factory SF2012. (http://www.erlang-factory.com/conference/SFBay2012/speakers/JesseNewland)

注11) http://basho.co.jp/products/riakcs/

マニュアルどおりでホントに使える?

小規模プロジェクト現場から学ぶJenkins活用

第1回 それほんとにプログラマがやる必要あるんですかね?

Jenkinsという言葉、聞いたことありますか? Hudsonなんて呼ばれていた時期なら知っているという方もいるかもしれません。世の中にJenkinsの解説をしているサイトは本家(jenkins-ci.org)を始めとしてたくさん存在します。しかし、そういったサイトを参考にこそすれ、自分の環境で便利に使おうと思うとやっぱり少し勝手が違っていました。プログラマだけが使って、毎日の作業が便利になって良かったね!では済まなかったのです。

嶋崎 聡(しまざき さとし) ㈱XVI Twitter @sato_c

小規模プロジェクトでの悩み

Jenkinsを使う環境と聞くとどうしても大規模なプロジェクトを想像しがちですよね。雑誌やWebで紹介されている利用法は、複数のプログラマが細かいパーツを作って、SubversionやGitでソースをコミット。その後の単体テストや結合テストはJenkinsが肩代わり、といったものが大半です。でも、いま筆者が使っている環境はむしろ逆です。のちほど少し紹介しますが、現状はプログラマの数はそれほどいないし、プログラムは必要最小限、むしろデータをまとめている時間のほうが長い状況です。

ここでいうデータとは、映像や画像、音楽や効果音、メッセージ用の文字列などです。それから、映像などのデータをどういった流れで再生するかを定義したものもデータです。このようにデータの種類、量ともに多い状況をひとりで切り盛りするには……、すべてを自分でやっていると時間がなくなってしまいます。単純な繰り返し作業が多くなればなるほど、時間がかかればかかるほど、ケアレスミスが増えていくので確認作業の時間が延びていきます。こんなめんどくさい状況、どうにかして作業を肩代わりしてくれるようなしくみがほしくなります。

また、データを作る作業は何度も何度も繰り返す必要があります。データの制作が1回で済むなんてことは、テスト中でもほとんどないでしょう。そういった状況を考えると、バッチファイルを実行する手間やコマンドラインからmakeと打つ手間すら省きたくなってきます。そんな手間すら省略してしまって完全に自動化できれば、構築にかかる手間と単純作業を繰り返す手間を比べても前者のメリットのほうが大きいと考えました。準備が整えば、あとは勝手にPCがデータを作ってくれるんです。そういう環境どうですか?

筆者はプログラマとしてプロジェクトに参加していますが、こういったシステムの構築/管理業務も兼任しています(小規模プロジェクトではプログラマが兼任することはよくあることでしょう)。そこで、そんなしくみを作るのにJenkinsを使うことを考えたわけです。

たとえばこんなとき、どうしてますか?

さて、プログラムを作っている最中にちょっとしたデータが必要になったとき、どういう手順で作っていますか? だいたいは「少ししかないから」ということで、プログラマ自身の手でテキストエディタに打ち込んで作っていたりしませんか。テキストエディタじゃなければExcelでしょうか。でもこれ、

結構大変ですよね。そのまま開発に入ってしまって、ずるずると自分の手で作り続けることになったりして。さらに、本番環境に移行する段階になってもそんな状況のままだとデータがそろえられずに「どうしよう!?」となったり、データの作り方に融通が利かなかったりして結局苦労することになって大変です。

「バッチファイルで処理するからいいよ」とか、「スクリプトで自動的に作るからいいよ」と最初からやっている人はかなりすばらしい環境にいると思います。しかし、あれば良いというものでもなかったりします。筆者の以前の職場では、手でやらずにすまそうと作ったはずのバッチファイルが、callを使っていくつも他のバッチファイルを呼び出し、呼び出されたバッチファイルでは結果を環境変数にセットして使い回す、なんていう構造化バッチファイルとでも呼ぶべき代物だったことがありました。こうなると、うまくいっているときはいいのですが、いざ問題があったときにはバッチファイルを作った人以外は理解不能で大変です。データを作りたいのか、バッチファイルのメンテナンスをしてるのかわからなくなったり……。

そのバッチファイルは後々ほかの人によって、もっとわかりやすいスクリプトとして生まれ変わりました。誰もが、これなら最初からスクリプト言語で書いてくれていたら良かったのに……というくらい実行時間も短縮されて、かなりの環境改善になったのです。見通しのよい形にしておくことで、データ形式や仕様変更などがあってもメンテナンスもしやすくなり、テスト期間が終わって、いざ本番というときに変更があっても楽になるのではないかと思います。

共同作業で感じる意識の差

現在の筆者の環境でもそうですが、デザイナがいる開発現場でよく聞く話は「自分たちは絵は描ける(描く)けど、それ以降のデータの使い方はわからない(知らない/どうでもいい)から、やっておいてよ」でしょうか。データをどういう手順で加工しているか、1から覚えてもらうのは大変です。一度説明したとしても、わからないと言われるたびに手順を最初から説明していくのは双方の負担になりますよね。お互いに貴重な時間を削り……と思ったら、質問してきた人はいつの間にか「自分には関係ない」的な感じになったり、用事があるから帰っちゃったりして……こっちも早く帰れるなら帰りたいです。そしてこんな悩みは、プロジェクト後半になったところでさらに増えていきます。

また、タスク管理でも問題が起こります。あの画像が抜けてる(だいたい大事な場面です)、画像や映像がなぜか古いものになってる、映像をエンコードする/しなおす……こうした未処理タスクはプロジェクトが佳境に入れば入るほど増えていきます。修正が入ってきたりすると、ますますデータ管理の負担が増します。もうやだなーって何度思ったかわかりません。

Jenkinsで解決してみました!

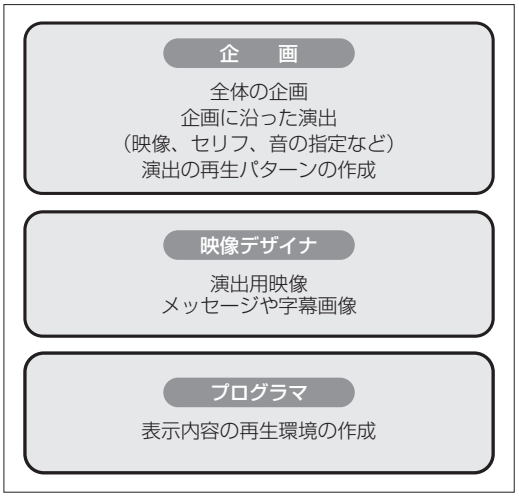
ちょっと自分の置かれてる状況に対する感想が混ざってしまいましたが、この連載ではこんな状況を打破するためにJenkinsを導入してみました!という事例を紹介していきたいと考えています。

筆者自身の置かれている状況から転用してますので、Jenkins入門みたいなのとはだいぶ違うかもしれません。また、本来のJenkinsの使い方とも違うかもしれません。あくまでも人数の少ないプロジェクトで、どうやったら効率的に動けるんだろうという話に絡めたJenkinsの運用方法ということでやっていきます。

こんな環境でやってます

現在の筆者がいる環境は、アミューズメント系システムの製作です。アミューズメント系システムというのをおおざっぱに説明すると、ユーザからの入力に反応して映像を流す、そういうシステムを構築しています。スタッフは筆者がプログラマで参加しているほか、デザイナが複数、企画・プロジェクト管理といった人たちがいます(**図1**)。規模としては、内部スタッフは多くてものべ10人程度。映像

●図1　プロジェクト内の仕事分担



や音楽、音声の素材については専門の会社に依頼することが多く、できあがってきた素材を社内で編集します。何から何まで内製でということは現状はほとんどありません。

デザイナは、画面に表示するもののデザインや映像編集作業など表示に関する作業を行います。彼らは映像や画像を扱う環境については、相当の知識を持っています。しかし、プログラムに関する知識を持っている方はあまり多くないです。何年か前は、PCの起動と終了方法、Photoshopの使い方くらいしかわからないなんて人もいました。最近はPCも便利になってきて仕事以外でもいろいろと使うことが増えましたから、そこまで極端な人はかなり減ったと思います。でも、自分が作った映像データをプログラムで使うデータにまで変換する作業については知らない人も多いです。こうしたデータの変換までやってくれる人はまれです（ぜんぜんいないとは言いません）。

企画担当者は、作るものの内容を考えることが仕事になります。制限のある中でどれだけ内容をおもしろくできるかが勝負です。ですから、普段からあまり細かいことは考えずに自分の作業に集中してもらうようにこちらとしても考えるようにしています。

プロジェクト管理者は、文字どおりプロジェクトの進捗について把握することが仕事です。決められた納期内にどうやってうまくプロジェクトを回して完成させるかに頭を使ってもらうよう情報を密に交換する必要があります。

このようにいろいろな業種の人が集まり、それぞれの作業はお互いに連動していますから、スムーズな連係ができるように気を使うところです。

プログラマが全部やるのは負担でしょ？

こういう環境で仕事をする場合、デザイナには最低限どういった情報を共有してほしいかを事前に説明して理解してもらわないといけません。企画の人が「今実装されている内容を確認したいんだけど、どうすればいいの？」と言ってきたときも対応しないわけにはいきません。相手の環境は自分が使っている環境と同じとは限りません（むしろ同じなわけがありません）から、どうにかして解決する必要があります。たとえば、その人専用のプログラムやExcelのマクロを組んで渡すとか。

そんな感じでデザイナや企画担当が「わからない」といっていることをすべてプログラマが肩代わりすることを考えてみてください。自分の仕事をする時間が……ありますか？　自分の仕事をする時間は、ある程度落ち着いた夕方以降……ああ、今日も終電か……みたいなループ、経験ある方も多いんじゃないでしょうか。筆者は胸張って言えますよ。「何度もあります！」……どうにかしたいですよね。

データベースで共有

このように少人数で作り上げる環境でも、作業を円滑に行うためのルールが筆者の社内にあります。それは、

“どんな状況でもまずデータはデータベース（以下、DB）に登録する”

です（長いので呼び方は以後「DB登録ルール」と省略します。**図2**）。

画像や映像の実体はさすがにDB登録しません。Subversionで管理しています。その代わり、そのデータのファイル名と名前、どこで利用するかという情報はDBに登録して管理します。これにデータ

●図2 DB登録ルール

DB登録ルール
～とにかく、登録できるものは登録しよう～

1. 演出内容
2. 映像の名前
3. セリフ
4. 音の名前
5. 各種メッセージ
6. 数値データ
7. 映像再生データの組み合わせ
8. 映像、画像、音のファイル名と内容を表す名前
9. 各種データ作業の進捗
X. その他思いつくかぎりのもの

の状態――まだ作ってないとか、作っている最中とか、完成しているという状況――も一緒に記録していけば、作っている人以外も現在の状況がわかります。

データの確認と登録にはExcelを利用しています。開発現場では、なんだかんだ言っても利用頻度の高いExcelを使うことで、PCを使い慣れない人にも説明しやすい環境が作れます。細かい動作を長々と説明するよりもシートにデータコミット用の「転送」と書かれたボタンを用意して「データを編集したら、このボタン押してね」で済んじゃいます。

ちょっと細かい話をしますと、ExcelからDBへの接続はODBC[注1]を使っています。ExcelにもDBにアクセスする手段はあるのですが、データを書き換えたときにDBも更新されるのがイヤなのでコミット用スクリプトを作っています。

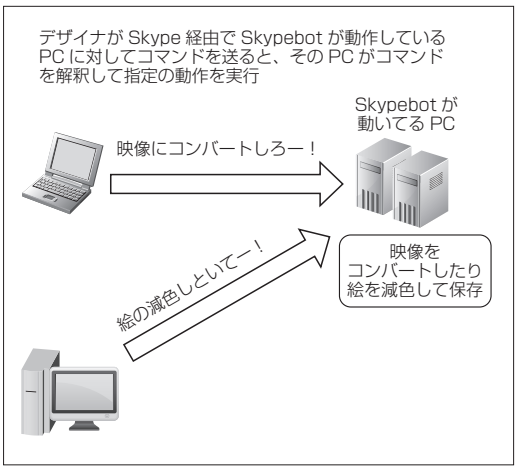
こうしてデータを手軽にDBへ転送できるようにしておけば、Excelを使って誰でも確認できるようになります。企画やプロジェクト管理者といった人たちも「あの映像できてんのかな？」と思ったときに簡単に調べられます。実際は確認のために作業している人たちに直接聞くことが必要になりますが、聞くまえにDB登録されたデータで下調べすることで、進捗確認もまず自分でできるわけです。デザイナが登録を忘れていないかもわかりますので、必然的に抜けがないかどうかを複数人で確認するようになり、結果、効率が良くなっています。

また、最初に書いたテスト用データを作る話にもつながるのですが、DBに必要なものをまとめておくといざというとき便利です。たとえば、映像を再生するときに必要な動画、画像のリストがあるとします。これをプログラムに埋め込むためには、DBではSQLを使って必要な部分と範囲を指定して抜き出してから、ソースファイルに整形して保存しておけばいいわけです。さらにこれをスクリプトにしておけば、データが更新されたらスクリプトを実行するだけでソースファイルも更新できます。これを繰り返すときに自分でスクリプトを実行しなくていい環境を作れたら、これもまた効率が良くなるんじゃないでしょうか。

Skypebotで自動処理

DB登録ルール以外にもともと社内で使っていたものとして、Skypeから命令を送るとそれに応じた作業を行う優秀な(？)Skypebotがありました。Skypebotとはなんだ？というと、IRCで特定のチャンネルに常駐して、受け取った文章によっていろいろな動作をするbot(ボット)にひっかけて命名された、Skypeによる自動実行機能のことです。Skypeのチャットを利用してテキスト指示をSkypebotが動作しているPCに送ると、これをトリガーに作業させることができます(**図3**)。このSkypebotでは

●図3 Skypebotの利用方法



注1) Open DataBase Connectivity。Microsoftによって提唱されたデータベースにアクセスするためのソフトウェアの仕様。

図4のような作業を行っていました。

便利に運用してはいましたが、いくつかの課題もありました。Skypebotがやる作業の一連の流れに新機能を追加するには、そのbotから呼び出す処理のプログラムやスクリプトを改造する必要があります。一度作った流れに別の処理を割り込ませて機能追加するとなると大規模な修正が必要な場合も出てきます。日常的に使われるツールなのですから、いざ修正したけどなんか変、なんてことがないようにするためにも神経使います。スケジュールの遅れは自分の作業時間にひびきますから……。

Jenkins導入の検討

Jenkinsは「継続的インテグレーション用ツール」ということですが、正直「何それ?」って感じではないでしょうか。ざっくり言うと、"自動でテストを動かして、結果を集計する"とか"ビルドしてできた成果物を必要に応じて展開する"とか"自動的にデータを作る"といったことができます。筆者のやりたいこともコンセプトは同じじゃない?ということで、Skypebotに置き換わるものとして使えないかと試してみることにしました。

Jenkinsというものがあることは社内でも認知されていました。しかし、Skypebotもありましたから、導入しても使い道があるのか?と、費用対効果を見積もれない状態でした。Skypebotがやっている作業でも充分省力化に貢献していましたし、プログラマが毎回使い方を説明しなくても"Skypeでコマンドを送る"という手順を一度覚えてもらうだけで済むという教育コストの低さも魅力でした。

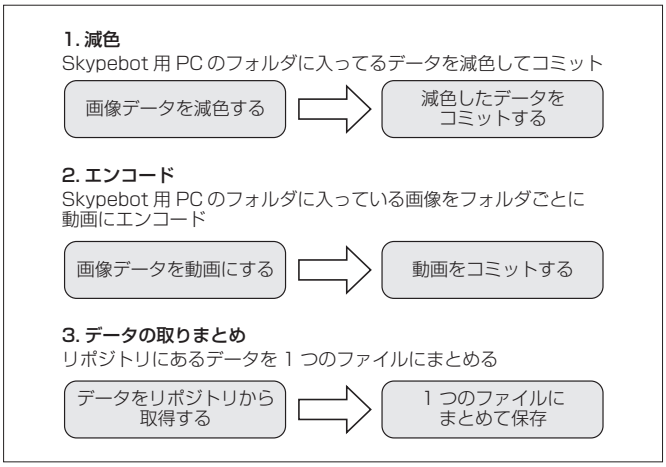
それでも置き換えることが決まったのは、前述のような課題があったことに加え、筆者が以前いた会社でJenkins用スレーブ数台とフロア内の数十台のPCを利用した分散ビルド環境という大規模なJenkinsの運用や管理に関わっていて、実作業を通してどういった使い方ができるかについて理解していたためです。規模は小さくなりましたが、筆者のやること、やりたいことは変わらないわけですから、会社としてもJenkinsの導入によって、現状よりも効率化を図れるかを見極めるいい機会と考えたようです。

必要な作業に対してどういった使い方をするかは追々考えるとして、まずは導入すると決まりました。導入してから、どう現場の作業にリンクさせていくかを詰めていったのです。フロントエンドは従来の形を使いつつ、そこから先の作業は拡張性を持たせるという前提条件はありましたが、この点についてはプロジェクトにかかわる人たちに協力してもらい、実作業に則した内容に落とし込むことができました。

いざ、導入——でも一筋縄では

Skypebotからの置き換えということで、まずは、

●図4　Skypebotの作業内容



「たくさんある映像データを1つのファイルにまとめる」ツールを動かすためにJenkinsのJOB、DATA_Compileを作りました。このDATA_Compileはツールが作ったデータファイルをJOBの成果物として保存します。また、変換中にツールが作成する設定ファイルは別形式のファイルにコンバートしてからSubversionのリポジトリにコミットします(**図5**)。

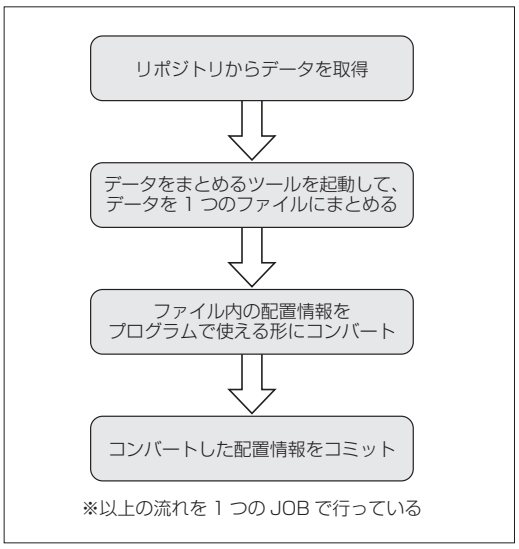
結論から言うと、これを作るのはかなり苦労しました。というのも、Windows環境のハードメーカー純正ツールを利用する場合、コマンドラインツールが用意されていることはあまりありません。使い方の説明はGUIツールのほうが便利という理由もあるのかもしれませんが、コマンドラインツールが提供されることはほとんどありません。そんなわけで、GUIツールをJenkinsから使うしかないので、いろいろ試行錯誤を繰り返しつつ、どうにか動かす環境を作りました。どのように解決するかは、次回以降で説明する予定です。

1つ解決しても、問題は次から次と出てきます。コンバートが終わり、最後にファイルをコミットするのですが、ここにも手間がかかりました。もともと存在するファイルを更新する分にはあまり苦労はありません。そのままコミットすれば、更新があったファイルがコミットされます。また、何かのタイミングでファイルの更新タイミングがずれてしまった場合は、手動にはなりますが、いったん手元のファイルとリポジトリの内容を合わせて、ファイルをコミットしてやります。これは複数人で作業する環境であれば、手元でもたまに起こることなので、メンテナンスの手間だと思えば仕方ないことだと思えます。

しかし、初めて作成されたファイルをどうするか?になると、少し話が違います。新規登録するには、そのファイルをリポジトリに追加してからコミットする必要があります。コミットするだけで自動的に追加してくれるほうが便利じゃないかと思いますが、実際は追加したくないファイルも追加されることがあって不便です。そこでいったんリポジトリにファイルが存在するかを調べて、なければ追加するスクリプトを書きました。コミットの件は解決しても、ほかにも細かいトラブルは出ました。こうしたトラブルも少しずつクリアしながら現在の環境を構築していきました。

構築した環境は、使い始めてすぐのころはいろいろと不具合が目立ちます。これは安定するまでは仕方ないことなので、そのつど直していくしかありません。数回動かせば単純な不具合はすぐ直せますから、周りの人にも協力してもらい、修正を繰り返しました。こうしてJenkins環境を作ったおかげで、プロジェクトが佳境に入ってから出た問題もスクリプトの追加やJOBの切り分けで素早く対処できています。

●図5 Jenkinsに移行したDATA_Compileで行っている作業



JOBについて

Jenkinsの運用事例を紹介する前に、基本となるJOBの扱いについて説明しておきましょう。作成したJOBを登録すると、**図6**のようにJenkinsのメイン画面(ダッシュボード)にリスト形式で並びます。JOB名の左側にある「S」には直近の実行結果がアイコンで表示されます。失敗すると赤い玉になります。玉の右隣の「W」は最近のJOBの状態を天気で表しています。何回も失敗が続くと雲行きが怪しくなる、といった具合です。

●図6　ダッシュボードにはJOBが並ぶ

JOBは単独でも複数で連動させても動かせます。1つのJOBで何から何までやらせようと考えてしまうかもしれませんが、お勧めは個々のJOBは単一機能にすることです。1つのJOBにいろいろやらせていくと、そのJOBが失敗したときにそれまでの時間が無駄になってしまうからです。たとえば、「プログラムをビルド→データをまとめる→プログラムとデータをまとめて1つのzipファイルにする→できたファイルをftpでアップロードする」という連係を考えてみてください。最後の最後、ftpでのアップロードが相手側のサーバ不調で失敗したとします。本当はアップロードだけを動かしたいのに単一JOBだった場合は最初から動かすことになります。スクリプトを直して対応しますか？　もしも、元に戻すのを忘れたら、毎回過去に作ったものをアップロードするだけ、なんてことにもなりそうです。目的は1つでも、複数の作業があります。そういった切り分けをどうするかで便利さが変わります。

Jenkins移行前は、Skypebotに作業をお願いしたあとはSkypeに出力される結果を見てから次の作業を行っていました。Jenkinsに置き換えてJOBの連動にしたことで、途中経過をあまり気にしなくてよくなっています。ダメならばダメだったところを確認して、そこからもう一度動かせるようになっているからです。これだけでも精神的な負担は軽減されるものなんです。

またJenkinsでは、結果に応じて違うJOBを起動するといった連係だけでなく、失敗通知メールを送る設定や、有志の方々が公開しているプラグインを使ってログ内容を解析して処理させたり、データ量を解析してグラフとして描画するなど、便利な設定もできます。このあたりは次回以降で徐々に紹介していきます。

今回のまとめ

かなり簡単ですが、本連載での前提となる環境とその中でJenkinsをどう使っているかをざっと紹介しました。JOBの作成と連係については日々の作業の中で組み替える必要が出てきますので、いかに細かい作業単位に切り分けられるか、それぞれを連動させていけるかが、作業のスピードに関係します。このことがわかるまでは結構苦労しました。

繰り返しになってしまうのですが、最初の失敗はどうしても1つのJOBでなんでもやらせたほうがお得な感じがしてしまうことです。実際は、1つのJOBがすべてをやったときの待ち時間は相当なものになります。その結果、JOBが失敗したときの時間を無駄にした感は相当なダメージになります。プロジェクト前半でプログラマに時間的余裕があるうちは、いくらでも実験用JOBを作って試行錯誤で

きます。どうすればうまくいくか、どうしたらダメかについては繰り返して覚えていくしかないです。トラブルがいつ起こるかは予想がつきません。そこからリカバリするにも気持ちの余裕は必要だと思いますので……。

それから、筆者がかかわっているアミューズメント系の環境においては、まだまだDBは馴染みがないものです。馴染みがないものはどうしても「難しいもの」であり、「自分には関係ないもの」となってしまう方が多いです。しかし、データの一元管理と俯瞰ができるのは実作業においてはかなり時間の短縮と手間の軽減につながります。

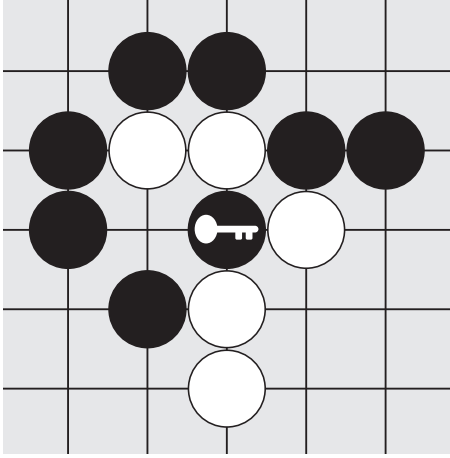
同じように、Jenkinsのような作業を自動化するツールもあまり一般的ではないと思います。「自分(もしくは人間)の手で作業しないと信用できない」という話もよく聞きます。でも、プログラマという職種は単純作業でデータを作るよりもデータは機械に作らせて、自分はもっとプログラムを書く時間を確保したほうがいいんじゃないかと思います。プログラマ自身の作業量や手間、時間を考えたら、導入することでほかの作業に使う時間をかなり作れるでしょう。

筆者の環境ではDBでのデータ管理とJenkinsを使った自動化で、プロジェクトの終盤になっても——多少の残業はあるにせよ——土日に出勤してまで作業に追われるということはありませんでした。「やっておいたほうがいいかなー」と考えたときもないことはないのですが、それでもなんとか休日出勤なしで作業できています。こうした時間的な恩恵は充分ありました。

今回は筆者の環境での導入に関する四方山話となりましたが、次回からは、構築した環境で使っているツールやスクリプトなどを紹介していきます。SD

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

【新連載】パスワードをもう一度考えてみよう

囲碁の初心者は基本的な定石（打ち方のパターン）を学び、その定石を必要な局面にあわせて使えるようになると、どんどん上達するそうです。また、定石は不変なものではなく、常に変化しているそうです。
セキュリティも同様です。基本的な知識を学ぶことは重要ですが、現在の技術やトレンドにあわせて見直していくことも重要です。第1回目のテーマは「パスワード」です。

そもそもパスワードって何？

正確には「パスワードを用いた認証方法って何？」という問いが正しいでしょう。定義を明確にするなら「知識を共有し、その知識を持っていることを確認することで、正当な相手であることを確認する」というメカニズムがあって、その「共有している知識」がパスワードです。

たとえば、忠臣蔵の赤穂浪士が吉良邸討ち入りの際に相手を確認するために「山」「川」といったのも共有している知識を確認しているので一種のパスワード認証と言えます。この場合、「山」と「川」がパスワードになります。

ですが、これは本人を認証しているわけではなく、知識を持っている者を確認しているにすぎません。「者」と言っていますが、人間じゃないかもしれません。あるいはパスワードを試して、たまたま同じだったとしても同様です。偶然であっても「知識の共有をしている」とみなされてしまいます。

パスさせるワード

コンピュータでパスワード方式の実現方法をみた場合、外形的にはだいたいこんな感じです。

8文字程度の文字列をユーザに入力させ、コンピュータ内部に事前に登録してある情報と付き合わせて同じかどうかを調べる。同じであれば秘匿されている知識を共有しているとみなす

どこまでの文字列の長さが使えるか、そしてどんな文字が使えるかはシステムに依存します。明確な基準はなく、かなり実装依存になります。UNIXのログインなどで使用するパスワードは、アルファベットの大文字と小文字、0～9までの数字、#や%といった何種類かの特殊文字が使えます。ですが、たとえばWebサイトのログインパスワードなどはアルファベット大文字小文字と数字のみというものも、かなりあります。過去にはパソコンのパスワードではアルファベット大文字小文字を区別しないというものがありました。このように使える文字、長さ1つとっても、実現方法は強くシステムに依存しています[注1]。

パスさせようとするワード

次の2つは、「悪いパスワード」の例としてよく出てくるパターンです。

注1）セキュリティポリシーといえば聞こえはいいのですが、これまでのシステムを見ていると、「とりあえずやってみた。ないよりはマシ」という程度のものが多いのが実状です。

①Hironobu1963
②12345678

①は名前と生まれた年の組み合わせです。本人は忘れることはないでしょうが、簡単に類推されてしまいます。

②は誰が考えてもひどいパスワードでしょう。でも、現実にはそんなに笑ってはいられないのです。2012年7月、CNETに「Yahoo hack reveals most-used passwords」という興味深い記事[注2]が載りました。2012年に米Yahoo!から45万件のパスワードが流出しましたが、この記事は、それを解析した結果が公開されたという内容です。わかったパスワードのうちで、筆者が興味を引いたものをリストアップしてみました(**表1**)。

2,295件の12345…というのは、そのまま1から順に数字を並べたものです。160件は111111という具合に同じ数字を並べたものです。780件はそのまま"password"という単語です。233件は"password"という単語の後ろに数字などをつけたものです。106件はbatmanやsupermanという誰でも思いつくような名前。筆者だけでなく、読者のみなさんも、「これはひどい」と思うでしょう。でも、これは現実に使われていたパスワードなのです。これだけで3,000アカウント以上が不正に使えるのです。

◆表1　米Yahoo!の45万パスワードの分析結果(一部)

発見したパスワード数	パスワードの例
2,295	12345、123456、1234567…
160	111111、0000000、777777…
780	password
233	password1、password2
106	batman、superman…
27	ncc1701、ncc1701a…

ところで27件あるncc1701、ncc1701aの並びはなんだと思いますか？　これを見て筆者は、「ほう！」と思いました。実はこれ、スタートレックのU.S.S.エンタープライズ号の登録番号なのです。つけたい気持ちはわかりますが、筆者が見て瞬時にわかる程度の情報は、すでにWikipedia上に存在します。そのWikipediaのテキストアーカイブをダウンロードし、すべての文字列パターンを抜き出し、ソートして重複文字列を消してしまえば、その中に入っているレベルの単語でしかありません。Wikipediaには大量のデータが入っているといっても、それを抜き出すのは、今やそんなに難しいことではありません。ノートパソコンでもWikipedia

◉PAMとlibpam-cracklib

GNU/Linuxのパスワード認証は現在はPAM(Pluggable Authentication Modules)というメカニズムを使っており、古典的なUNIXより高度なパスワード認証の枠組みを備えています。

名前のとおり、プラグインするような形式になっているので、必要に応じてパスワードのチェックを行う設定が可能です。

たとえばDebian GNU/Linux 6.0.7の場合、libpam-cracklibというプラグインが入っていない状態だと6文字パスワードを許しますが、libpam-cracklibを入れるとデフォルト設定で8文字になるといった具合になります。システムによってはデフォルトで8文字より大きい文字は無視するものもあるようですが(筆者は見つけたことはないのですが、このような設定の可能性はあります)、libpam-cracklibを使えば少なくともデフォルトで12文字までのパスワードは使えます。

libpam-cracklibのインストールは次のように行います。

```
# sudo apt-get install libpam-cracklib
```

インストールが完了すると、/etc/pam.d/common-passwordに次の1行が加わり、パスワードのデフォルト設定が8文字になります。

```
password requisite   pam_cracklib.so ↵
retry=3 minlen=8 difok=3
```

注2) http://news.cnet.com/8301-33692_3-57471417-305/yahoo-hack-reveals-most-used-passwords/

ベースのパスワード類推用辞書は作ってしまえます。

本来の役目を果たしていないパスワード

前述の記事では、漏洩したパスワードデータのすべてを判明できたわけではなく、あくまでもパスワード探しを行い、見つけられたものだけです。でもどれだけ見つけられたのでしょうか？

報告によれば45万件の漏洩したパスワードのうち13万7,559件、つまり全体の約30％が判明したそうです。そして、そのうちの10万6,873件がGmailやHotmailに使いまわされていたそうです。

今、この原稿を書いている最中に「ディノスに111万件の不正アクセス、1万5000件の不正ログイン」[注3]というニュースが入ってきました。このディノスの大量の不正ログインも、先ほどと同様に、ほかのサイトから漏れたパスワードを使ってログインを試していると考えられます。

最近は、Webサービスのアカウント名をメールアドレスにするものが多く、それで同じパスワードを使いまわしていると、1つのサイトでパスワードが判明されれば、芋づる式に次々にほかのサイトもアウトになります。

パスワードの本来の目的はアカウントを使うユーザを正しく認識するためのものです。ですが、システムの全アカウントの3割がその役目を果たしていない、しかも、まったく情報を漏らしていないはずのサイトでさえも影響がおよびます。

ディノスの例では、割合は1.35％と低い率と考えるのではなく、ディノスに大きな落ち度もないのに、不正に利用できるアカウントが1万5,000件も手に入るという見方をすべきです。これだけ手に入れば何をするにしても十分過ぎるほどの数でしょう。

「パスワードがきちんとしていれば安心だ」と言うかもしれませんが、これだけ悲惨な数字を見ていると、これまでのパスワード認証は期待どおりに役にたっているとは到底言える状況ではありません。

システムにおける適切なパスワードの管理とは

ここでもう一度、パスワード管理の方法とパスワードを不正に割り出す方法を整理してみましょう。そのほうが議論の全体像を整理できてわかりやすいと思います。

ですが、その前に「123456」、「password」のようなパスワードや、アカウント名が「admin」でパスワードも「admin」といったものでは、さしたる事前の準備も必要なく、いくつか試せばいいので、これから述べるようなパスワードの割り出しの作業すら必要ありません。これはさすがに、迂闊で愚かなパスワードとしか呼びようがありません。しかし、これはユーザだけが悪いわけではなく、システムがこのようなパスワードを許容すること自体が、システムの欠陥だと筆者は思います。

それでは本題に入りましょう。ある程度の複雑さを持ったパスワードに関しては、パスワード管理ファイルをサイトから流出させ、専用のコンピュータを用意してパスワードを見つける処理を行うというのが前提となります。サイトへの侵入方法やデータファイルの流出に関しては別途回をあらためて話をしたいと思います。

さて、以下に3つのパスワードの管理パターンを説明し、次に、どのような方法でパスワードを見つけていくかの説明を加えます。

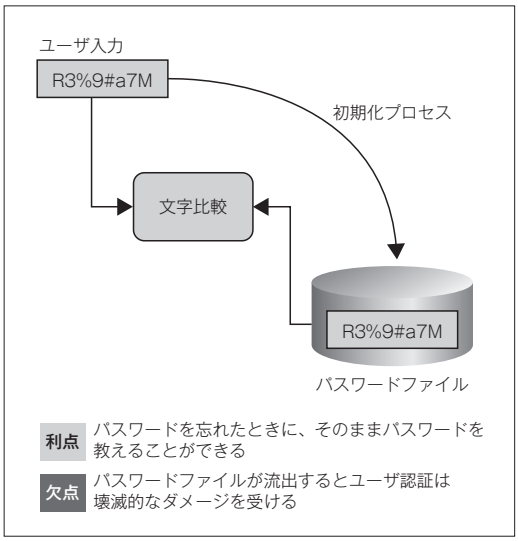
パスワードを防御なしに管理

パスワード認証というと、ユーザからの入力の文字列と、すでに登録しているパスワードの文字列を付き合わせる（**図1**）ことだと思っている人が意外と多いのではないかと思います。

これは簡単に実装できますが、パスワードファイルが外部に流出した時点で、サービスの存続に関わるレベルで、システム全体のユーザ認証が崩壊します。なぜならば、そのままの形ですべてのパスワー

注3） http://itpro.nikkeibp.co.jp/article/NEWS/20130510/475982/

◆図1　保存されているパスワードは保護されていない



ドを利用できるからです。

中には、「これはファイルへのアクセスをコントロールすることで安全性を保っている」と理解している人がいるでしょうが、逆をいえば、それだけしか保護をしておらず、情報流出する可能性については考慮されていません。1つのエラーがシステム全体をカタストロフィーな状態に導く可能性がある脆弱な状態におかれています。

このシステムが、大きな問題をはらんでいるのは、あらためて強調しなくとも多くの読者のみなさんは理解されているだろうと思います。その一方で、このようなシステムは、それほど多くはないだろうと楽観的に考えるかもしれません。でも、パスワードを忘れたときに親切にももともとのパスワードを教えてくれるタイプのシステムは、周りを見渡せば結構あるはずです。

一方向ハッシュ関数を導入する

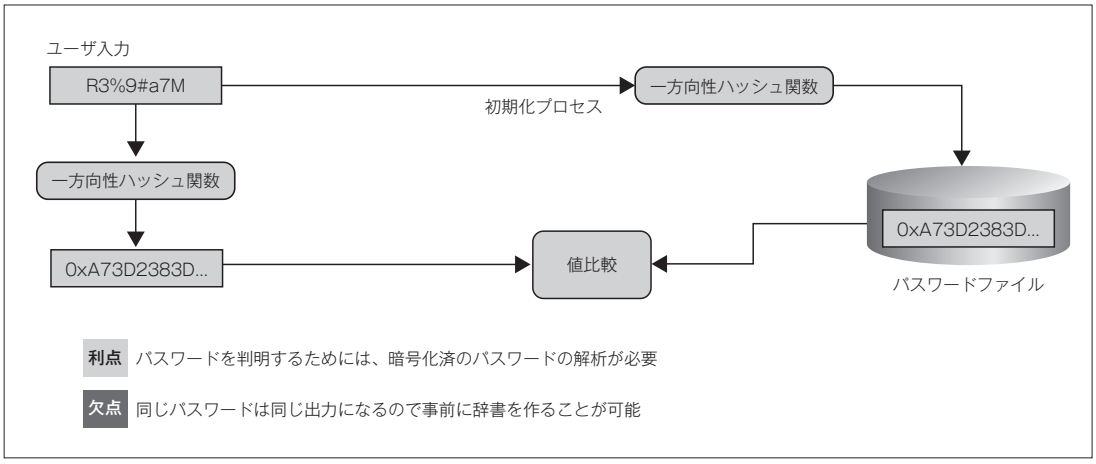
一方向ハッシュ関数とは、値xに対しハッシュ関数Hを用いて計算した値H(x)から、逆をたどってxを見つけることは極めて困難であるという性質を持つ関数です。一方向ハッシュ関数としては、MD5、SHA-1、SHA256といったものだけではなく、暗号化関数を使ったメッセージ認証コードなども同様に使えます。たとえば古典的UNIXのパスワード生成にはDES暗号を使っているDES-CBC-MACというメッセージ認証コード法が使われています。

よく「パスワードを暗号化」するという表現を使うので、暗号化するなら復号もできるだろうと類推しそうですが、これは一方向にしか計算できません。ですので一方向ハッシュ関数なのです。

パスワードを入力し、それを一方向ハッシュ関数で計算した値を、事前に登録しておいた一方向ハッシュ関数で計算した値と比較します(**図2**)。たとえばWindows XPなどで使っていたLMハッシュ(LAN Manager hash)は、この方式です。

生の文字列を持っている**図1**の方法よりは格段にマシになりましたが、たとえば同じ入力のものは同じ出力値を持ってしまう弱点を持っています。です

◆図2　一方向性ハッシュ関数を導入する



ので辞書に載っているようなパスワードであれば、事前に処理してハッシュ化済み辞書を作成できます。つまり、辞書攻撃には極めて脆弱なパスワードシステムだといえます（コラム「パスワードの強さ」も参照）。

ソルトを加えたパスワード管理

一方向性ハッシュ関数だけだと逆引きの攻撃辞書が使えるので、それを困難にするようにソルト（salt）と呼ばれる、ユーザごとに異なるランダムな値を加えたのちに一方向性ハッシュ関数に入力する方法をとります（**図3**）。

このソルトは隠さなくてもかまいません。長ければ長いほど逆引きの攻撃辞書のサイズが巨大になります。ソルトはランダムデータですが、それほど大きなものは必要なくここ数年のパスワードの安全性でかまわなければ32ビット（4バイト）程度でも十分に役目は果たします。70年代では12ビット程度が使われていました。将来的にも使うと考えるならば、80ビット（10バイト）程度あればかなり確実性をもって安全と言えるでしょう[注4]。

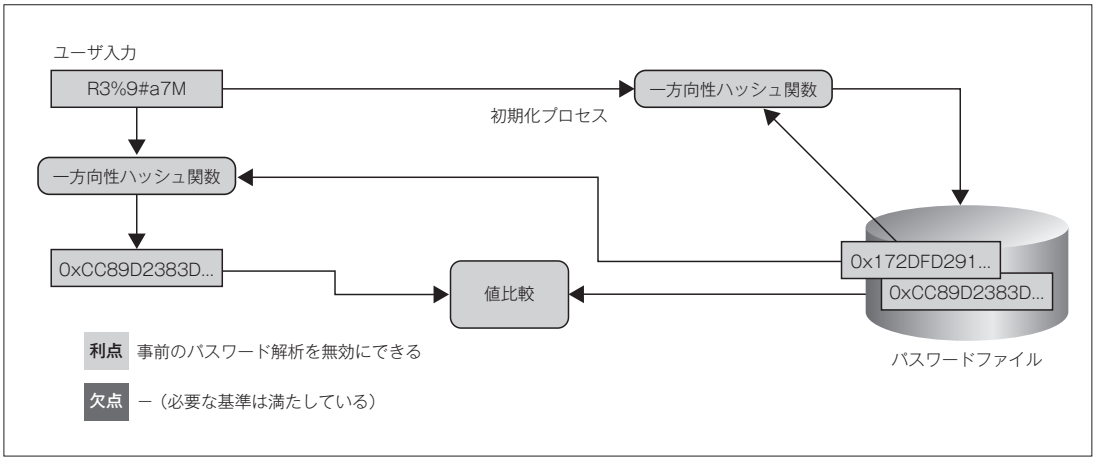
UNIXのパスワード処理では、さらに一方向性ハッシュ関数を複数回かけて計算時間をより多くかけさせるという手法を使っています。ただし、一度に多数のユーザの対応をしなければならないWebサイトで行うのは、認証サーバ側に負荷がかかり過ぎる可能性があるので、本当にこの手法を選択すべきかどうかは、考える必要があります。基本的には、複雑さを増すにはパスワードに使える文字種類を多くしたり、文字数の数を増やすことが重要です。これでやっと普通のパスワード管理です[注5]。

安全なパスワードだけれども

よくある「安全なパスワードを運用しましょう」という説明には「難しい並びのパスワード」「長いパスワード」「頻繁に変更するパスワード」という条件が推奨されています。しかし、次のポイントを指摘したいと思います。

- **難しい並びのパスワード**
  人が考えるとバイアスが入るので安全なランダム性を確保できない
- **長いパスワード**
  人間の記憶能力を過大評価している。結果として思い出しやすいパスワードを選択するバイアスがかかる

◆**図3　ソルト（salt）を加える**



注4）ただし、パスワードが12文字～16文字分のランダム文字列といった安全性の非常に高いものでなければ、ソルトだけビット数を増やしても意味がありません。

注5）ここでの「普通」とは、ただ単純に「統計的に多くあるパターン」という意味ではなく、「理想に近い」という意味での普通です（参考：「Dream Fighter」by Y.Nakata）。

● 頻繁に変更するパスワード

1番目と2番目の問題を何度も繰り返すことになり、結果として、覚えやすいパスワードか、同じパスワードを繰り返し使われることになる可能性が大きい

たまに「“i love you”ならば“1 l0v3 y0u”といった具合に規則性をもってほかの文字列に置き換えをすると英語の辞書に載っていないので辞書攻撃は避けられる」「“i want to eat cake”といった好きな文章の頭文字をとって“iwtec”とすると良い」と解説するWebサイトを見かけますが、この程度の乱雑さは、コンピュータの前では乱雑さにすらならず無力です。

ネットワーク経由で自由にアクセスできるサイトサービスにおいて、パスワードのみで抵抗するならば、すべてのパスワードを十分に長いランダム文字列を機械で生成するしか方法はありません。当然、数十とか百を越えるランダム文字列を人間は覚えき

◉パスワードの強さ

辞書に載っていない完全にランダムな文字列をパスワードとしていたら、どれほど安心できるのでしょうか。条件として文字の複雑さは英数字大文字小文字の区別あり（0-9A-Za-z）の62文字が使えるとしましょう。Webサービスだと標準的なパターンだと思います。8文字で約218兆3401億通り、10文字で約84京通りです。

一方向ハッシュ関数はSHA-256を使っているということにします。次に使えるパソコンは一台で、秋葉原で調達できる機材の範囲とします。ただし、お金は多少のことは目をつぶりましょう。ヘビーゲーマーレベルのNVIDIAGeForce GTX-690を4枚挿して、CPUは4コア以上搭載したパソコンを使うことにします。

あくまでも筆者のざっくりな試算[注A]ですが、SHA-256は1秒間に22億回程度の計算ができると見積もっています。ちなみにSHA-1ですと約44億回程度、MD5だと約140億回程度の計算ができるでしょう。

平均を求めるわけですから、8文字の組み合わせは約218兆通りですので半分の109兆までの処理となり、それには49622秒、つまりおおよそ14時間あれば見つかるということになります。10文字では6年強となります。

ソルトを持たない一方向ハッシュ関数の場合、逆引き辞書が使えます。その逆引き辞書を作るのは8文字パスワードであれば計算する時間は障害とならないのは上記のとおりです。しかし、そのデータを保持するには、今回の想定する範囲では無理です。何も考えないで辞書を作ろうとすると、大量のパスワードとハッシュ値の関係リストが必要になります。ざっくり見積もって10PB（ペタバイト）クラスの容量が必要になります。これはデーターセンターで使う最大級の外部記憶装置に匹敵します。

現在、ハードディスクは大容量で安くなっているといっても、せいぜい4TB程度です。これからも進化するでしょうが、それでもペタクラスの容量を秋葉原から調達するのはまだまだ先の話でしょう。

ですが、暗号学的な手法[注B]を使ってすべての対応表をもたず、探すための補助テーブルを作る方法を使うと、引きかえにいくばくかの検索時間はかかりますが、劇的に保持するデータ量を削減できます。しかも、その実装は公開されている[注C]ので、そのコードを使い、どこかに実際にあるシステムに適応させるのは難しいことではありません。

ちなみにWindows XPやVistaのための辞書テーブルがすでに用意されており[注D]、Vistaで8文字ランダム英数字（大文字小文字区別あり）の条件のパスワードだと、テーブルのデータサイズは134.6GB（検索成功率99%）だそうです。これはすでにノートパソコンに載る容量サイズです。ノートパソコンでも数秒から数十秒の検索時間の範囲で99%の確率で正しいパスワードを見つけることができるようです。

注A）Ivan Golubev's Password Recovery Suiteのデータをもとに推定。 http://www.golubev.com/igprs/
注B）Making a Faster Cryptanalytic Time-Memory Trade-Off http://lasecwww.epfl.ch/~oechslin/publications/crypto03.pdf
注C）OPHCRACKサイト http://lasecwww.epfl.ch/~oechslin/projects/ophcrack/
注D）http://ophcrack.sourceforge.net/tables.php

れませんので、パスワード管理のためのツールを使うことになります。計算機が簡単に推定可能なパスワードを使うくらいなら、十分に長いランダム文字列を紙にかいて、その紙を物理的にしっかり管理したほうがよっぽど安全です。

あるいは、もう諦めて、自分も次回はログインできないかわりに誰もログインできないくらい複雑なパスワードを考えて、使うときはパスワードを再設定するかです。

これくらい割り切って使わないとパスワードが本来与えるはずであろうユーザ認証の能力を引き出せません。

まとめ

さて、現実を眺めると、多くの場面で導入されているこれまでのパスワード管理／運用方法は、すでに寿命がつきかけていると言わざるを得ません。しかし一方で、いまだにパスワードは全盛です。

筆者は公開鍵方式を利用したユーザ認証方法を積極的に取り入れるべきだと思っています。この技術は、UNIXユーザにとってはSSHの公開鍵認証などで身近に使われていますが、一般にはなかなか普及できていないのが実状です。

この現状ですが、先端企業は徐々に変わりつつあります。すでに、Googleのアカウントは二重認証を取り入れています。筆者はすでに使っているのですが、それほど煩雑というほどではありません。銀行などのように想定される被害金額が大きい場合、ワンタイムパスワードのセキュリティトークンを使うのも合理的な方法だと思います。

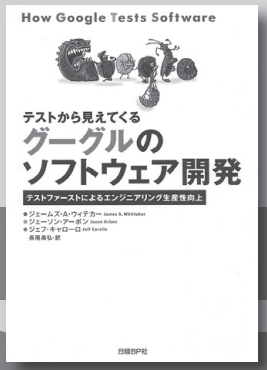
今は、これまでの古典的なパスワード認証から次の世代の認証へと変化する端境期（はざかいき）だと言えるのかもしれません。ですが、まだしばらくは、これまでのパスワードとの付き合いは続きそうです。

ユーザ認証に関しては、まだまだ書き足りないことがたくさんありますが、それはまた別の機会に書くとして、今回はこのへんで。SD

BOOK no.1

テストから見えてくる グーグルのソフトウェア開発

ジェームズ・A・ウィテカー、 ジェーソン・アーボン、 ジェフ・キャロー口 【著】 ／長尾 高弘 【訳】
A5判、 432ページ／価格＝2,730円 （税込） ／発行＝日経BP社
ISBN＝978-4-8222-8512-8



ソフトウェア開発におけるテストの存在は、品質を保つために不可欠な工程でありながら、納期を圧迫するやっかいごとのように思える。しかしそれは、リリーススパンの長いデスクトップ向けソフトウェア開発に倣ったテスト手法を、スピードが問われる現代の開発に用いているからではないだろうか。Googleで生産性向上（テストと品質管理以上の役割）を担うチームメンバーによって書かれた本書は、従来とは異なる同社のテストに対する考え方、テストのやり方、人材の見つけ方、組織のあり方などが紹介されている。デスクトップからクラウドへの移行に直面しているソフトウェア開発現場にとって、テストを見直す指針として大いに参考になるはずだ。

BOOK no.2

次世代ネットワーク制御技術 OpenFlow入門

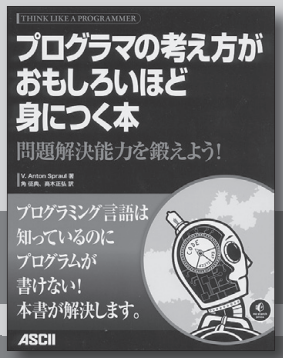
石井 秀治、 大山 裕泰、 河合 栄治 【著】
B5変形判、 192ページ／価格＝2,100円 （税込） ／発行＝アスキー ・ メディアワークス
ISBN＝978-4-04-886953-9

SDN/OpenFlowの書籍も増えてきた。そろそろ本格的な運用に向けて、各ベンダが開発に鎬を削るようになるだろう。6月に開催されたInteropがその分水嶺になりそうだ。本誌短期連載でSDNの落とし穴について伊勢幸一氏が解説しているように、ネットワークの本質的な理解なしにこの技術の利用は困難である。本書が優れているのは、ネットワーク機器の解説にかなり力をいれていることだ。ソフトウェアでネットワークを抽象表現するために、ハードウェアのしくみを知らねばならない。その構成のため、OpenFlowで何を実現したいのか理解しやすいものになっているのだが、従来のネットワーク技術も復習できてお得な本である。

BOOK no.3

プログラマの考え方がおもしろいほど身につく本
問題解決能力を鍛えよう！

V.Anton Spraul 【著】 ／角 征典、 髙木 正弘 【訳】
B5変形判、 256ページ／価格＝2,310円 （税込） ／発行＝アスキー ・ メディアワークス
ISBN＝978-4-04-886955-3



本書は副題にある「問題解決能力」を養うための本である。人の書いたプログラムの修正やデバッグはできても、自分で一から起こすとなると発想力を問われる。そんなときの考え方について解説されている。最初に問題解決の戦略の例として「川渡り問題」や「スライドパズル」、「数独」など、やさしげな題材から入っている。その後にパズルのプログラミング的な解決方法を学び、配列／ポインタ／クラス／再帰などで実際にC++でのプログラミングを考えながら学習していく。最後にプログラマの考え方として、弱点をふまえた計画の立て方や問題に立ち向かう心がけ、新しいスキルの学び方などを解説している。翻訳書のせいか1段落が長く読みにくい感はあるが、解説は丁寧で、プログラミングを学ぶ本とは別なベクトルなので有意義であろう。

BOOK no.4

ソフト・エッジ
ソフトウェア開発の科学を求めて

中島 震、 みわよしこ 【著】
新書判、 272ページ／価格＝798円 （税込） ／発行＝丸善出版
ISBN＝978-4-621-05383-6

本書はソフトウェアに関わる問題をさまざまな視点から概観する。かつて社会問題となった銀行や証券システムの大規模障害などを題材に、なぜ、どういうしくみで起こったのかを解説しつつ、根本的な原因を探る。本書の後半では、さらに視野を広げ、ソフトウェアを成り立たせている理論や原理の側面から、あるいは設計やテストといった開発の側面から、ソフトウェアの特徴をつかもうと試みる。システム障害が起こると、原因はソフトウェアの特殊性に求められることが多いが、それは正しいのか？ ソフトウェアにかかわらず多くの人が携わって作ったものには同じ問題が潜んでいるのではないか？ そんなことを考えさせられる1冊だ。

プログラム知識ゼロからはじめる
iPhoneブックアプリ開発

第3回

アプリ開発に必要な画像を準備する

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。今回はアプリ制作に必要な画像データの作り方について解説します。

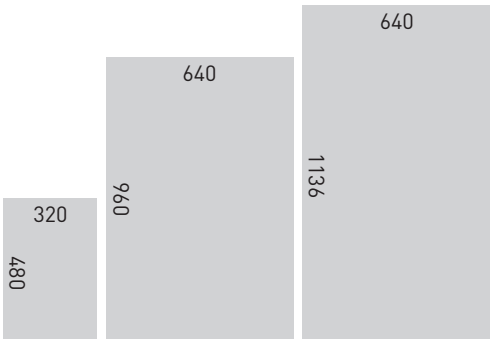
今回のテーマ

iPhoneアプリを作るためにはさまざまな準備が必要ですが、そのうちの1つが画像の用意です。これはプログラム同様、アプリを完成させるための重要なステップの1つになります。今回はまず、アプリを完成させるために絶対に必要な画像の種類を把握し、それらの画像を準備するまでのプロセスを解説します。

なぜいろいろな種類の画像を用意する必要があるのか？

画像の種類を多く必要とする理由は、機種によって使用する画像が異なるからです。現在、iPhoneとiPod touchだけで見ると次の3つのタイプの機種に別れます（**図1**）。

▼図1　3タイプのディスプレイ解像度



①初代iPhoneから3GSまでの3.5インチ画面の機種（画素数が320×480px[注1]）
②iPhone 4から4Sまでの3.5インチRetinaディスプレイの機種（画素数がそれまでの倍の640×960px）
③iPhone 5からの4インチRetinaディスプレイの機種（画素数が縦だけ伸びて640×1136px）

iPhone 5が出たばかりのころは、4インチ画面用の画像は推奨ではあったものの、必須ではありませんでした。しかし今後は4インチ画面に対応していないアプリは審査で落ちてしまうため、必ず対応しなければいけなくなりました。本連載でも第1回目と第2回目ではアプリ開発の基本をわかりやすく解説するためにあえて3.5インチ画面にのみ対応しましたが、今回からは4インチ画面にも対応させていきます。このため、最低でも3タイプの機種用に画像を用意しなければいけないということになります。

画像を作る

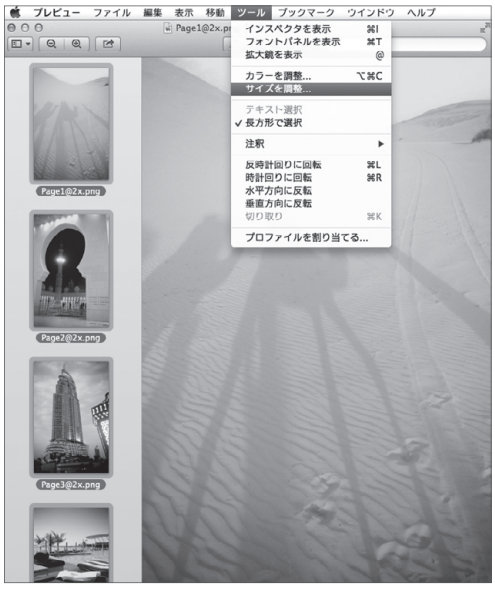
前述のとおり、いくつかのパターンで画像を制作することになりますが、シンプルに考えれば実際には3.5インチと4インチという2つの画面サイズの違いを意識すればいいだけの話です。旧機種の3.5インチ（低画素）用の画像は、高画

注1） px＝ピクセル

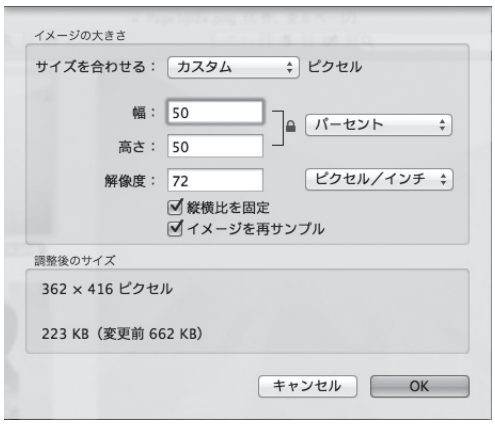
素の画像を元に50%（半分）に縮小すればいいのです。これはMacの付属アプリ「プレビュー」（**図2**）を使えば簡単に行えます。

▼図2　プレビューのアイコン

▼図3　プレビューで画像を表示



▼図4　プレビューでサイズを50%に変換



まず、縮小したい画像をまとめて「プレビュー」にドラッグ＆ドロップして開きましょう。すると、左側にすべての画像のサムネイルが縦に並びます（1枚の場合は出ません）。サムネイルをどれか1つクリックし、command A ですべての画像を選択し、メニューバーの［ツール］から［サイズを調整...］を選んでください（**図3**）。現れたダイアログボックスの幅と高さ欄の右にあるプルダウンメニューを「パーセント」にし、両方とも「100」を「50」に変更します（**図4**）。あとは［OK］を押せばすべての画像が半分のサイズに縮小されます。

画像のファイル形式

iPhoneではさまざまな画像形式の読み込みに対応していますが、Appleの推奨はPNG形式の画像なので、できる限りPNGを使用しましょう。PNGは透明（半透明）の透過画像にもできるため、アプリ開発では扱いやすいということもあります。たとえば丸みのあるボタンの場合、PNGであれば余白の部分を透明のまま使用できるので、簡単に背景と同化させることができます（**図5**）。

画像のピクセルサイズは偶数で

画像を作るときのピクセル数は偶数で作ることを心がけましょう。ピクセルは1ピクセル単位で区切られ、半ピクセルというものは存在しません。なのでRetina用の画像サイズに奇数の

▼図5　透過画像の効果

ピクセル数を使用した場合、そのちょうど半分のサイズは存在しないので、低画素機種用画像を作るときには一番近いピクセル数に自動的に変換されてしまいます。こうなると、レイアウト上のズレが発生する原因となりえるので注意が必要です。

画像の種類

iPhone開発に絶対に欠かせない画像は次のとおりです。

[アイコン画像(3種)図6]

・icon.png(57×57px)
・icon@2x.png(114×114px)
・App Store用高画素アイコン(1024×1024px)

[デフォルト画像(最低1つ)]

・Default-568h@2x.png(640×1136pxの画像)

[アプリ内で使用するその他の画像(3パターン)]

・3.5インチ画面用非Retina画像(320×480pxまたは480×320pxの画面に合わせたもの)
・3.5インチ画面用Retina画像(640×960pxまたは960×640pxの画面に合わせたもの)
・4インチ画面用Retina画像(640×1136pxまたは1136×640pxの画面に合わせたもの)

▼図6　3種類のアイコン画像例

それでは1つ1つの画像について細かく説明していきます。

@2x画像について

まず最初に説明しなければいけないのが、ファイル名と拡張子の間に「@2x」が入った画像についてです。これは高画素の「Retinaディスプレイ用」の画像であることを区別するために使用されています。ファイル名のあとに@2xと付けるだけでプログラムがそれをRetina用の画像と認識し、高画素機種を使用しているときは自動的に切り替えてくれるというわけです。ですから、低画素と高画素の機種の画像を切り替えるために特別なプログラムを組む必要はありません。

反面、3.5インチと4インチを切り替えるためにはプログラムで行うしかありません。その代わり、ファイル名の付け方に特別なルールはありません。Default画像と同じようにファイル名のあとに高さ568ポイントを意味する「-568h」を付けるか、「-4inch」を付けるなどして区別すると管理しやすいでしょう。

ピクセル数とポイント数の違い

今回はたくさんの数字が出てくるため、混乱してしまわないように覚えておいてほしいことがあります。それはピクセル数とポイント数の違いです。プログラムは物を描写したり動かしたりするとき、ピクセル数ではなく“ポイント数”を使って計算されます。

このポイント数は標準解像度(非Retina)画面では1ポイント＝1px(縦・横1×1ピクセル)で換算されているため、ポイント数とピクセル数は比例した関係にありました。しかし、解像度が倍になったRetinaディスプレイでもこのポイント数の扱いは統一させなければいけないため、Retinaディスプレイ端末の場合は縦・横2pxにつき1ポイントと換算されます。つまり、Retinaディスプレイの場合では、ピクセル数の半分がポイント数になるということです。新しい4インチ画面の縦幅はピクセルだと1136pxあります

が、ポイントに変換すると半分の568ポイントとなります。4インチ画像のファイル名の最後に付く“568h”はまさにこのポイント数を表しているのです。

実際にはそこまで難しいことではありませんが、扱い慣れていない数字だとちゃんと理解できるまで少し時間がかかるかもしれません。プログラムを書き出すようになれば自然と理解できるようになることなので、今はそこまで深く考えなくても大丈夫です。

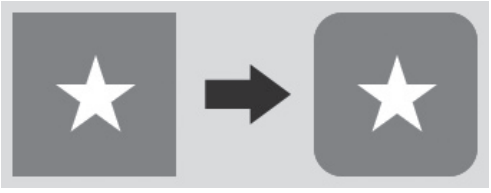
アイコン画像

ホーム画面に並ぶアイコン画像を2通り(低画素と高画素)、そしてApp Store上ではより高画素な画像が必要になるため、全部で3パターンのアイコン画像を用意する必要があります。

App Store用の高画素アイコン(1024×1024px)はアプリ内では使用しないため、アプリに入れる必要はありません。しかしアプリ提出時にアップロードすることになるので、最初にまとめて作っておくと良いでしょう。一番楽なのは、最初に1024×1024pxのサイズでアイコンを作り、それを縮小して114×114pxと57×57pxの2つのアイコンを作る方法です。ただし縮小率が高いため、小さくしたときに思ったとおりの仕上がりにならないこともあります。その場合は、それぞれのサイズに合わせて微調整、またはそのサイズ用に一から作り直す必要性があるかもしれません。とくに57×57pxは画素数が低いので、あまりディテールの細かい絵は崩れてしまう恐れがあります。

もう1つの注意点は、アイコン画像では半透明背景を使えないことです。アイコンは正方形に作りましょう。角の丸みはホーム画面に並ぶときに自動的に処理されます(**図7**)。

▼図7　正方形で作れば自動で角丸に処理される



デフォルト画像

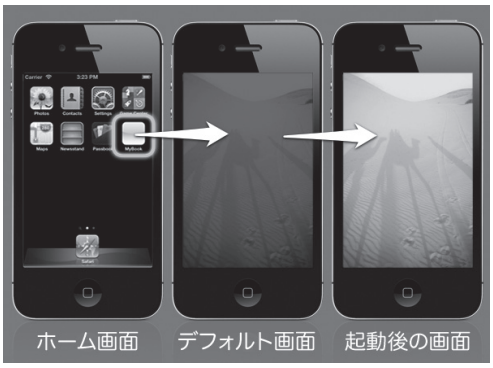
デフォルト画像とは、アプリのアイコンを押した瞬間から、アプリが起動(読み込みが完了)するまでのほんの短い時間をつなぐための画像です。スプラッシュ画像とも呼びます。最近のiPhoneの性能だと、よほど起動に時間がかかるアプリでない限りはそんなに待たされることもないので、ただの真っ黒の画像でも良いです。ただ、ブランドロゴを見せたり、“読み込み中”の表示を見せたりといった工夫も凝らせます。

工夫の1つとして、起動後に最初に出現する画面のスクリーンショットをデフォルト画像に設定することで瞬時に起動したように見せるという手法もありますが、これには賛否両論あり、ユーザによっては起動しているのに触っても反応がないことにストレスを感じる方もいます。たとえば半透明の黒の画像(50〜70%くらい)をかぶせることで、まだ完全には起動しきっていないように見せると混乱が避けられるかもしれません(**図8**)。

アプリ内で使用するその他の画像

アプリ内で使用する画像とは、ボタン画像や背景画像、写真集アプリなら写真の画像など、使用するすべての画像を指します。ボタン画像

▼図8　デフォルト画像の表示例



など、ものによっては低解像度と高解像度の2種類だけで済む画像も多いですが、画面全体を埋め尽くす背景画像やフルスクリーンで表示する写真や絵の場合は、3.5インチ用の低解像度&高解像度画像、4インチ用の高解像度画像の計3パターンを用意することになります。

◇

ここまでの説明を読んで、たくさんの画像を作成しなければいけないように感じるかもしれませんが、用意する画像のうちの約半分は、元となる高画素の画像を半分(50%)に縮小しているだけなので、実際にはそこまで大がかりな作業というわけではありません。どのような画像を用意しなければいけないのかさえしっかり把握できれば、画像の制作過程をスムーズに効率よく進められるでしょう。

第2回に作った写真集アプリで具体的に考えてみましょう。必要な画像は次のとおりです。

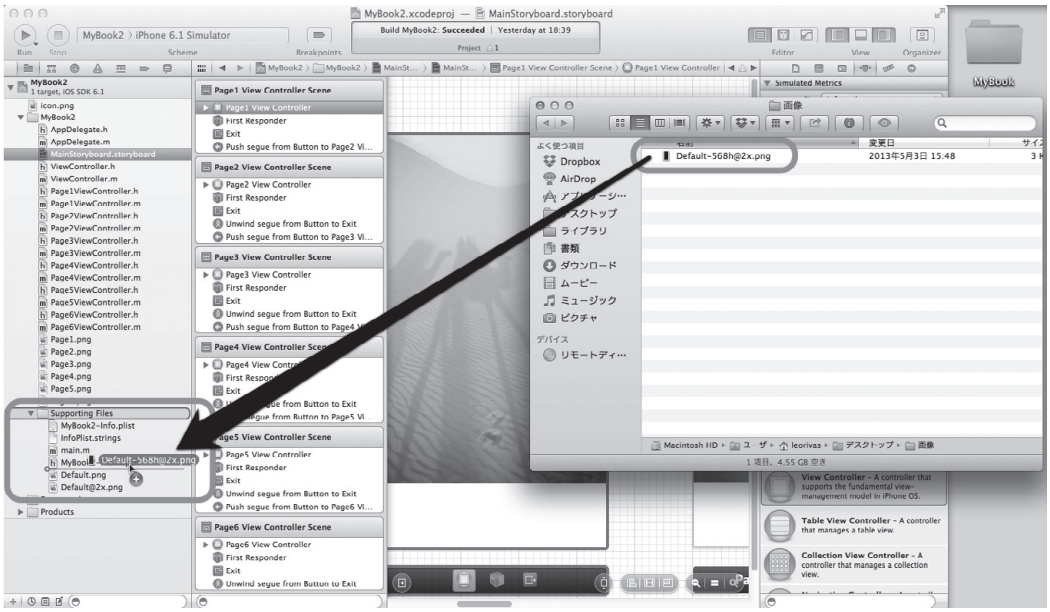
(a)ホーム画面に並ぶアイコン2種類
(b)デフォルト画像
(c)各ページの写真画像3種類ずつ

(a)のアイコンについては、現状ではApp Storeへの公開はしないので、当面114×114pxと57×57pxの2つのアイコンを用意すれば大丈夫です。ただ、前述のとおり大きな画像から小さな画像に縮小するという作り方を考えれば、App Store用のアイコンから作るというのもありでしょう。本稿の例では、2つのアイコンのファイル名は「icon.png」「icon@2x.png」としました。

(b)のデフォルト画像は、本来ならば新プロジェクトを作った段階で自動的に含まれているファイルです(Default.png、Default@2x.png、Default-568h@2x.png)。ですが、本連載では第1回目のステップ9であえてDefault-568h@2x.pngを削除して4インチ非対応にしたため、この画像のみもう一度追加しなおすことになります(これから新しくプロジェクトを作成する方はすでに同名のファイルが含まれているはずなので、さらに追加する必要はありません)。画像サイズが640×1136pxであれば、ただの真っ黒の画像で構いません。

(c)の各ページの写真画像(Page1.png〜

▼図9　Default-568h@2x.pngをSupporting Files内にコピー



Page6.png。皆さんが作った写真集アプリでは画像の枚数などが違うと思いますので適宜読み替えてください)は、これまで3.5インチ画面用非Retina画像のみ作っていました。そこで残りの2種類の写真画像について、それぞれ「4インチ画面用Retina画像」と「3.5インチ画面用Retina画像」を作り、前者のファイル名を「PageX-568h@2x.png(Xは各ページの数字)」、後者のファイル名を「PageX@2x.png(Xは各ページの数字)」としてください。これで追加する写真画像のファイルは、Page1-568h@2x.png～Page6-568h@2x.png と Page1@2x.png～Page6@2x.pngが用意できたはずです。

アプリを4インチ画面に対応させる方法

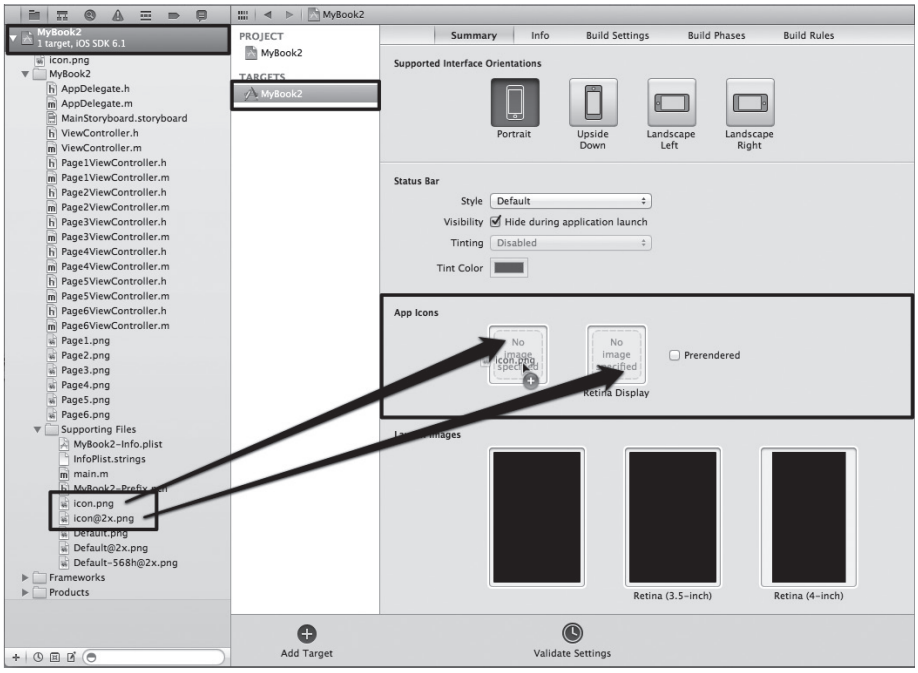
すべての画像の準備が終わったら、いよいよXcodeでアプリの4インチ対応をします。ここからは第2回で作成したアプリに対する処理として説明しますので、作成済みのアプリのプロジェクト(記事例ではMyBook2.xcodeproj)をXcodeで開いておいてください。

まず最初に、Default-568h@2x.pngをXcodeの「Supporting Files」の中にドラッグ&ドロップしてください(**図9**)。同じ要領でアイコン画像2つを「Supporting Files」の中に、追加の写真画像は元の写真画像と同じフォルダ(例では「MyBook2」)の中に入れます。

アイコンを設定するためには、左カラムの一番上のプロジェクトファイルをクリックし、「TARGETS」の下のプロジェクト名(例ではMyBook2)を選択しましょう(**図10**)。スクロールして「App Icons」エリアを探し、さきほど「Supporting Files」に追加したアイコンファイルを**図11**のように1つずつ四角い箱の中にドラッグ&ドロップしてください。左が低画素アイコン、右がRetinaディスプレイ用アイコンです。これでホーム画面に並ぶアイコンとして反映されるようになります。初期設定では、アイコンには自動的に光沢が付くようになっていますが、光沢を付けたくない場合は、「Prerendered」にチェックを入れましょう。

さて、Default-568h@2x.pngが追加された時

▼図10　アイコン画像をそれぞれセット



点で一応は4インチ画面に対応したことになりますが、内部の画像が4インチ用に調整されていなければ余白は残ったままなので、本当の意味での対応とは言えません。

Default-568h@2x.pngを入れていない状態(4インチ未対応)でアプリを起動すると、画像を中心に、上下の余ったスペースに黒帯がかかった状態になります(**図11の左**)。一方、Default-568h@2x.pngを入れてから(4インチ対応の状態で)シミュレーターで確認してみると、画像が上に寄せられ、下の部分に白い大きな余白ができます(**図11の右**)。見栄えはよくありませんが、とりあえずこの状態であれば4インチに対応したことになります。

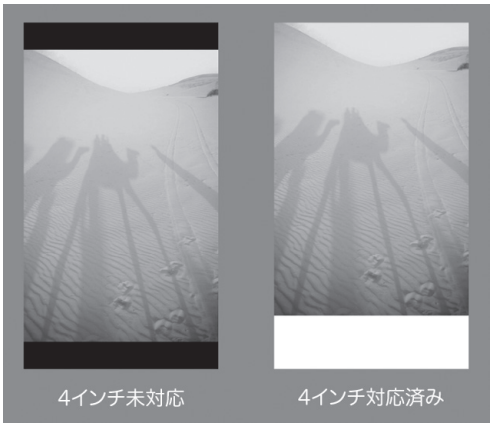
この先は3.5インチと4インチの画面の切り替えをどのようにしたいかによって対処法は変わってきます。最もシンプルでコードを1行も書かずに済ませる方法は、下の余白部分(88ポイント分)になんらかのデザインを加え、4インチのときだけは見えて、3.5インチのときは隠れて見えないようにする手法でしょう(**図12**)。

しかし、3.5インチのときと4インチのときにそれぞれ独自の画像を表示させたい場合は、コードによる対処が必要になります。

まとめ

以上がiPhoneアプリ開発に必要な画像の準備の説明です。次号の第4回では実際にコードを使って、3.5インチ画面用の画像と4インチ画面用の画像を機種に応じて切り替える方法を解説します。SD

▼図11　4インチ用デフォルト画像の有無



▼図12　余白部分にデザインを追加した例

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。個人ではスタジオルーペとして数字を指でドラッグ&ドロップ保存できる「フュージョン計算機(FusionCalc)」が代表作。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。現在、iPhoneアプリ「ヘル・ベースボール」にて漫画を不定期連載中。

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「Crop It Cam!」(おしゃれな切り抜き写真カメラ)、「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。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38回 安心安全なアプリケーションを公開するために

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！

谷口 岳　TANIGUCHI Gaku
タオソフトウェア㈱

普及するスマホとセキュリティ事情

携帯電話はスマートフォンと名を変え、急速に普及をしました。スマートフォンは従来の携帯電話と異なり、自分でソフトウェアを入れられる非常に便利なものです。しかしセキュリティ的にはどうでしょうか？　肌身離さず持ち歩く携帯電話は利用者がどこへ行ったのか、誰と電話したのか、電話帳にどのような友人がいるのか、といったプライベートな情報が蓄えられています。しかも、これらの情報にはアプリケーション（以下、アプリ）がアクセス可能です。

悪意を持って作られたアプリであるマルウェアも、もちろんこれらの情報にアクセス可能で、中には電話帳を盗み取ることを目的としたものもあります。しかし電話帳からの情報取得は、電話帳アプリやSNSアプリとしては必要なものですので、電話帳から情報取得するアプリがすべて悪いアプリとは言えません。正しい使い方のために取得されたのか、盗み取られているのかわからないものが数多くあり、利用者に不安を与えています。

このような現状で、Androidアプリ開発者としては何ができるでしょうか？「安心安全なアプリケーション」をユーザに届けるにはどうしたら良いでしょう。本稿では、「安全なアプリケーションを作る方法」、「安心できるアプリケーションを公開する方法」について解説をしたいと思います。

安全なアプリケーションを作る方法

アプリに脆弱性があると、他の悪意のあるアプリに利用者の情報を抜き取られたり、破壊されたりします。残念ながらAndroidの公式のドキュメントでは、開発者が押さえておくべきセキュリティに関するまとまった資料がありません。日本語で参考にできる資料としては現在次の2つが存在します。

- 「Android Security～安全なアプリケーションを作成するために[注1]」（図1）
- 「Androidアプリのセキュア設計・セキュアコーディングガイド[注2]」（図2）

これらしかありませんが、逆に言えばこの2冊を読破してしまえば、Androidのセキュリティに関する事項はだいたい押さえられます。ぜひ読破してくださいと言いたいのですが、参考資料の紹介で終わってしまいますので、ここでは良く見かける脆弱性を3点ご紹介したいと思います。

注1）弊社著作。インプレスジャパン刊。ISBN 978-4844331346 URL http://www.taosoftware.co.jp/android/android_security/

注2）日本スマートフォンセキュリティ協会（JSSEC）の公開資料。URL http://www.jssec.org/report/securecoding.html

▼図1　Android Security～安全なアプリケーションを作成するために

▼図2　Androidアプリのセキュア設計・セキュアコーディングガイド

成するときは、自分のアプリしか読み書きできないようにしてください（APIのデフォルトの動作です）。

注意する点は外部記憶装置（SDカードなど）を利用する場合で、他のアプリから読み書きできないファイルパーミッションのファイルを作成しても、外部記憶装置の特性上、他のアプリから読み書き可能になります。重要なデータは外部記憶装置に置かないようにしてください。

コンポーネントの脆弱性

Androidアプリは複数のコンポーネント（Activity、Service、BroadcastReceiver、Content Provider）が集まって1つのアプリを構成します。コンポーネント単位で他のコンポーネントと情報のやり取りをしますが、他のアプリのコンポーネントからアクセス可能になっている例が多く見られます。

たとえば、アプリ内だけで使用するContent Providerが外部アプリからアクセス可能になっている場合は、他のアプリからデータを参照することが可能です。また電話番号を渡すと電話をかけるActivityを作った場合、このActivityに外部からのアクセスが可能だと、パーミッションを持ってないアプリが電話をかけられるようになってしまいます。このようなことがないように、外部からのアクセスが必要のないコンポーネントは、AndroidManifest.xmlで「export=false」を指定してください。

ファイルの脆弱性

重要なデータを書いたファイルを他のアプリから読み込み可能にすると、重要なデータが漏洩（ろうえい）することになります。Androidでファイルを作

WebViewの脆弱性

アプリでWebページを表示するために、WebViewが良く使用されます。WebViewにURLを指定するだけで簡単にブラウザもどきが作成できます。非常に簡単で便利なしくみですが、このWebViewに悪意のあるJavaScriptが含まれるHTMLをロードし、実行させてしまうと情報を抜き取られます。このため次の事項に注意してください。

1. WebViewを使用するときは必要がなければJavaScriptを実行させない設定にする（デフォルトの動作です）
2. JavaScriptを使用するときは、WebViewにロードするHTMLは他のアプリから改変可能な場所に置かない（外部記憶装置等）
3. JavaScriptを使用するときは、WebViewにロードするURLはアクセス先を限定し、HTTPS通信をする

ここで述べた脆弱性はよく見かけますし、発見しやすい脆弱性です。設計ミスはもちろんですが、ケアレスミスとしても発生しがちなので注意してください。

安心できるアプリケーションを公開する方法

スマートフォンには行動履歴や通信履歴などのさまざまな利用者情報が蓄積されており、アプリがそれらに対してアクセスを行い、外部へ送信している場合があります。一方、利用者情報の利用目的が不明瞭で、十分な説明がないまま取得・活用するアプリも増加しており、利用者の不安が高まっているとされています。このような社会情勢を受けて、2012年8月に、総務省から「スマートフォン プライバシー イニシアティブ」ドキュメントが公開されました[注3]。

スマートフォン プライバシーイニシアティブへの準拠

この指針では「アプリケーションごとにプライバシーポリシードキュメントを策定すると共に、一定の情報の取得については、個別の情報の取得について同意取得を求める」こととされています。一読するのをお勧めしますが、このドキュメントは複数のスマートフォンOSを対象としており、また対象読者がアプリ提供者だけでなく、移動体通信事業者、OS事業者、アプリ紹介サイト、情報収集モジュール提供者と非常に広くなっています。そこで弊社では、Androidアプリの開発者に絞ったガイドライン「Androidスマートフォンプライバシーガイドライン by タオソフトウェア[注4]」を無料で公開しているので参照いただければと思います(**図3**)。

スマートフォン プライバシー イニシアティブでは、利用者情報を取得し、外部通信や蓄積を行うアプリは次のことをするように推奨しています。

1. アプリケーションプライバシーポリシーを作成
2. 重要な情報を取得するときにダイアログでユーザに告知

注3）http://www.soumu.go.jp/menu_news/s-news/01kiban08_02000087.html

注4）http://www.taosoftware.co.jp/android/android_privacy_policy/

▼図3　Androidスマートフォンプライバシーガイドライン by タオソフトウェア

このあとから上記2つに関して解説をしていきますが、その前に、ここで出てきた「利用者情報」という言葉について補足しておきます。「利用者情報」とは、利用者の識別にかかわる情報、電話帳などの第三者に関する情報、利用者の通信サービス上の行動履歴、利用者の状態に関する情報などのスマートフォンの利用者に関する情報すべてを指します。個人情報保護法における個人情報とは異なることに注意してください。利用者情報は個人情報保護法と関連しますが、わざわざ新しい言葉である「利用者情報」という言葉を定義して使用しています。これは現在、個人情報の解釈が人によって分かれるような状況になってしまい、個人情報という言葉を使用すると非常にややこしくなるからです。

アプリケーションプライバシーポリシー

アプリを作成したらアプリのプライバシーポリシードキュメントを作成します。現在ほとんどの会社のWebページではプライバシーポリシードキュメントが存在しますが、このドキュメントとは異なるので注意してください。また複数のアプリで使用できるプライバシーポリシーを作成すると、取得していない情報の取り扱いが含まれたりしてあいまいなものとなりますの

で、必ずアプリごとに作成してください。またアプリに利用規約がある場合、利用規約と混ぜて記載を行うと、これも非常にわかりにくくなります。分離して作成してください。

プライバシーポリシーには次の8つの内容を記載します。

1.情報を収集するアプリ提供者などの氏名または名称

アプリ提供者の名称や連絡先などを記載します。

2.取得される情報の項目

これはパーミッション主体での記載や電話帳の情報といった大まかな情報ではなく、メールアドレス、電話番号など具体的に記載をします。

3.取得方法

自動的にアプリが取得するのか、利用者が入力するかを記載します。

4.利用目的の特定・明示

アプリ自体の目的に使用するのか、アプリ以外の目的(広告等)に使用するのかを記載します。また取得したデータを第三者提供する場合はその旨記載します。

5.通知・公表または「同意取得」の方法、利用者関与の方法

「同意取得の対象、タイミング」はアプリ内で同意取得ダイアログを出すタイミング(アプリ起動時か、データ取得時か)、またダイアログを出すか出さないかを記載します。

「利用者関与の方法」とは、利用者がアプリにより利用者情報の利用や取得の中止を希望する場合に、その方法を記載します。方法としては、アプリのアンインストール、サーバにログインしてアカウント削除するなどがあげられます。注意事項としては、利用者取得の停止ができないものは作らないということです。

6.外部送信・第三者提供・情報モジュールの有無

「第三者」とは、アプリ提供者以外、あるいはサービスを提供している会社以外の人を言います。たとえば、アプリに含まれている広告会社であったり、情報収集会社であったりします。第三者に情報を提供している場合は、その記載を行います。ほとんどは広告会社でしょうが、①組み込んでいる情報収集モジュールの名称、②情報収集モジュール提供者の名称、③取得される情報の項目、④利用目的、⑤第三者提供の有無などです。しかし、これらの情報が開発者に提供されておらずわからない場合があります。この件は問題になっており、現在広告会社の業界団体により取り組みがなされています。しばらくすると日本における広告モジュールのプライバシーポリシーが公開されることになると思いますので、それを参照してください。1点注意することは、海外の広告モジュールを使用する場合記載が難しくなることです。後述する第三者モジュールの注意点を参考にしてください。

7.問い合わせ窓口

問い合わせ窓口の連絡先などを記載してください。

8.プライバシーポリシーの変更を行う場合の手順

「プライバシーポリシーの変更」については、Webサイトでは、「弊社サイトで告知します」という表現をよく使用しますが、アプリではWebサイトで変更されてもユーザは気が付きませんので、この手法は使用できません。重要なプライバシーポリシーの変更に関しては、アプリ内でポップアップして告知するしくみが必要となります。

また、総務省のドキュメントにはありませんが、Androidではパーミッション情報はアプリの特性を判断する重要な情報なので、使用して

いるパーミッションとその利用目的を書いておくと良いでしょう。弊社で書きました、具体的なサンプルが次のURLにありますので、参考にしていただけたらと思います。

[プライバシーポリシー例]

- tPacketCapture
  http://www.taosoftware.co.jp/android/packetcapture/
- tSpyChecker
  http://www.taosoftware.co.jp/android/spychecker/

プライバシーポリシーの掲示

作成したプライバシーポリシーの置き場所も非常に大事です。

- アプリをインストールする前にプライバシーポリシー概要が参照可能であること
- アプリをインストールする前にプライバシーポリシーが参照可能であること
- アプリからプライバシーポリシーの参照が可能であること

アプリケーションプライバシーポリシーはドキュメントが長くなりがちです。利用規約が実質ユーザに読まれていないように、プライバシーポリシーも読まれない可能性もあります。このためGoogle Playに公開する場合は、概要を作成してアプリ説明欄に記載するのが良いでしょう(**図4**)。

Google PlayにはアプリのプライバシーポリシーのURLを記載する場所があります(**図5**)。プライバシーポリシードキュメントをGoogle Playとは異なるアプリ説明ページに置き、そのURLを記載します。アプリの詳細画面で「プライバシーポリシー」のリンクが表示され(**図6**)、ユーザがタップするとプライバシーポリシードキュメントが表示されるようになります(**図7**)。

また、アプリの使用中にプライバシーポリシーを参照したいこともあります。このようなときのために、アプリ内にプライバシーポリシードキュメントを埋め込んだり、先ほどのプライバ

▼図4　プライバシーポリシー概要案

プライバシーポリシー概要

本アプリケーションは、電話帳のバックアップ機能のため電話帳に含まれるすべてを取得し弊社サーバに送信します。

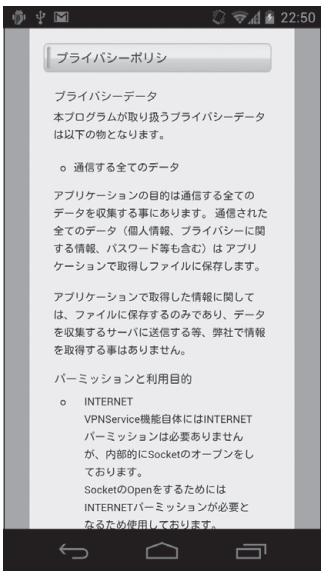
これらのデータは本アプリケーション以外の目的には使用しません。
アプリケーションには広告が含まれますが、広告会社には電話帳データは送信されません。

プライバシーポリシーの詳細につきましては、
http://www.taosoftware.co.jp/android/packetcapture/#privacy_policy
を参照ください。上記URLへは、デベロッパ情報のプライバシーポリシーリンクから移動可能です。

▼図5　Google Play Developer Console上の入力画面

▼図6　アプリ詳細画面のプライバシーポリシーリンク

▼図7　クリックしたときの表示



▼図8　パーミッション確認画面

▼図9　ポップアップ例

シーポリシーリンクを表示できるような、メニューやアバウトボックスなどを作成することをお勧めします。

同意取得ダイアログ

重要なデータを取得する場合は、ユーザに同意を求める必要があります。Androidでは、アプリインストール時にパーミッション確認画面が表示されます(**図8**)。

ここで、「同意してダウンロード」ボタンをユーザが押したことによって、利用者の情報を自由に扱う同意を得たことにはならないので注意してください。なぜなら、具体的な利用目的や外部送信の有無、第三者提供の有無などが書かれておらず、何に同意したら良いのかわからないからです。ユーザに同意を求める場合は、**図9**のようにアプリ内でポップアップダイアログなどを出して詳しく説明したうえで同意を取得してください。

ポップアップダイアログによる個別同意取得が必要な情報かは、次の2つから判断します。

1. 利用者情報の性質と種類
2. 利用者情報の利用目的

1では、個人情報および個人識別性が高い利用者情報を用いるときは、個別同意取得を求めます。かみくだくと個人情報そのものや、個人情報になりうる、氏名、年齢、電話帳などのデータ、また利用者による変更が困難な、IMEIやAndroid IDなども該当します。

2では、電話帳アプリが電話帳データを取得するなど、アプリ自体の機能のために情報を用いるときは利用者にわかりやすいですが、広告や利用状況把握などのために利用者情報を用いるのは利用者にわかりにくいです。したがって、アプリ自体の機能以外に利用者情報を用いるときは個別同意取得を求めます。

IMEIやAndroid IDを利用するには個別同意取得が必要な点と、どのような些細な情報でもアプリ本来の目的以外の利用には個別同意取得が必要な点に注意してください。

パーミッションを最低限にする

プライバシーポリシーや同意取得ダイアログの表示は非常に大事ですが、Androidではパーミッションによりアプリができることが異なってきます。電話帳を取得するパーミッションがあるアプリとないアプリでどちらが安心できるかと言えば、パーミッションがないアプリのほうが安心できます。

Google Playのアプリを見ていると、必要がないのにパーミッションを付けているアプリが多く見られます。昔存在した機能で必要だったパーミッションであったり、開発時に参考にした間違った情報をそのまま記載していたり、使用しているライブラリのドキュメントをよく読まずに記載したりといった例が見受けられます。パーミッションは必要最小限にするようにしてください。

第三者モジュールに注意

最後になりますが、アプリに自分が作成したもの以外を入れるときは注意してください。以前FaceBook SDKに脆弱性があり、そのFaceBook SDKを組み込んだアプリすべてが脆弱性を持ってしまうということがありました。ソースコードがある場合は必ずソースコードを見るようにしてください。また、広告モジュールではソースコードが存在しないものがほとんどですが、マルウェアの動作をする広告モジュールが世の中に存在します。マルウェアと認定されている広告モジュールを入れて配布したら、そのアプリはマルウェアとなります。普通に考えて、PC用に買ったパッケージアプリがウィルスに感染していた場合、制作会社にクレームをつけるでしょう。同じようにマルウェアが入ったアプリを公開した場合は、アプリを制作したあなたの責任となり、多くのユーザに迷惑をかけることになります。

利用者情報を収集する広告アプリは多く存在し、多くの利用者情報を吸い上げるアプリほど、広告収入で取得できる金額が多かったりします。広告収入は重要な収入源ではありますが、わけのわからない広告モジュールは入れないように注意してください。

最後に

「安心、安全なアプリケーションを公開」するには、結構手間がかかります。自分はアプリを作りたいだけだと思っている方もいるかもしれません。また、どのような情報を取得して利用するかを記載すると、怖がって使ってもらえないかもしれないという声をときどき聞きます。どちらにせよ、物を作って人に使ってもらうということは、対価をもらっているにせよ、もらっていないにせよ、責任が生じます。自分の作成したものが人に迷惑をかけないように脆弱性を持つアプリを作らないようにしてください。また、自分が不利になるかもしれないことを書くのは確かに嫌なものですが、自分の作ったものが大切なら、ごまかしはせずに堂々と自信を持って公開をしましょう。SD

Column

Androidアプリケーション脆弱性診断Webサービス「Tao RiskFinder」

第三者モジュールの取り扱いで述べたように、現在Androidアプリは開発用のライブラリも含め多くの外部モジュールを使用するようになりました。このような状況では、故意にではないにせよ、ライセンスに問題があるものを入れてしまったり、危険なモジュールを入れてしまったりする可能性があります。また、Androidアプリの脆弱性も大きな問題です。アプリを発注する会社では、不正モジュールの発見や脆弱性の問題を見つけるのは非常に困難です。そこで弊社ではアプリの脆弱性や問題を検査できるツール「Tao RiskFinder」を開発しました。現在法人利用のみとなっておりますが、ご興味のある方はぜひご利用ください。将来は個人の開発者の方が利用できるような形態にして、世の中のアプリが少しでも良いアプリになってもらえたらと思っております。

Tao RiskFinder
http://www.taosoftware.co.jp/services/riskfinder/index.html

谷口 岳（たにぐち がく）タオソフトウェア㈱ 代表取締役

Android OSの発表直後からAndroid開発を開始、ブログや講演などAndroidを広める活動を行っている。現在Android専業会社に転換し、受託開発、コンサルタントなどを手がける。

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10回

Intel VT-xを用いたハイパーバイザの実装 その6「ユーザランドでのI/Oエミュレーション」

浅田 拓也（ASADA Takuya）Twitter @syuu1228

はじめに

前回は、VMX non root modeからvmm.koへVMExitしてきたときの処理を解説しました。今回はI/O命令によるVMExitを受けて行われるユーザランドでのエミュレーション処理を解説します。

解説対象のソースコードについて

本連載では、FreeBSD-CURRENTに実装されているBHyVeのソースコードを解説しています。このソースコードは、FreeBSDのSubversionリポジトリから取得できます。リビジョンはr245673を用いています。

お手持ちのPCにSubversionをインストールし、次のようなコマンドでソースコードを取得してください。

```
svn co -r245673 svn://svn.freebsd.org/base/head src
```

/usr/sbin/bhyve による仮想CPUの実行処理のおさらい

/usr/sbin/bhyveは仮想CPUの数だけスレッドを起動し、それぞれのスレッドが/dev/vmm/${name}に対してVM_RUN ioctlを発行します（**図1**）。vmm.koはioctlを受けてCPUをVMX non root modeへ切り替えゲストOSを実行します（VMEntry）。

VMX non root modeでハイパーバイザの介入が必要な何らかのイベントが発生すると制御がvmm.koへ戻され、イベントがトラップされます（VMExit）。

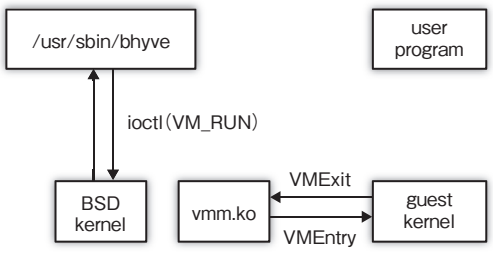
イベントの種類が/usr/sbin/bhyveでハンドルされる必要のあるものだった場合、ioctlはリターンされ、制御が/usr/sbin/bhyveへ移ります。/usr/sbin/bhyveはイベントの種類やレジスタの値などを参照し、デバイスエミュレーションなどの処理を行います。

今回は、この/usr/sbin/bhyveでのデバイスエミュレーション処理の部分を見ていきます。

/usr/sbin/bhyve での I/O 命令ハンドリング

前回の記事に引き続き、I/O命令でVMExitした場合について見ていきます。VMExitに関する情報はVM_RUN ioctlの引数であるstruct vm_runのvm_exitメンバ（struct vm_exit）に書き込まれ、ioctl return時

▼図1　VM_RUN ioctlによる仮想CPUの実行イメージ



にユーザランドへコピーされます。/usr/sbin/bhyveはこれを受け取り、vmexit->exitcodeを参照してどのようなVMExit要因だったか判定し、VMExit要因ごとの処理を呼び出します。I/O命令でVMExitした場合のexitcodeはVM_EXIT_INOUTです。

VM_EXIT_INOUTの場合、I/Oの命令のエミュレーションに必要な情報（ポート番号、アクセス幅、書き込み値（読み込み時は不要）、I/O方向（in/out））がstruct vm_exitを介してvmm.koから渡されます。

/usr/sbin/bhyveはこの値をI/Oポートエミュレーションハンドラに渡し、I/Oポート番号からどのデバイスへのアクセスなのかを判定し、デバイスのハンドラを呼び出します。

ハンドラの実行が終わったら、/usr/sbin/bhyveはふたたびVM_RUN ioctlを発行して、ゲストマシンの実行を再開します。

では、以上のことを踏まえてソースコードの詳細を見ていきましょう。**リスト1**、**リスト2**、**リスト3**、**リスト4**にソースコードを示します。キャプションの丸数字で読む順番を示しています。

vmmapi.c と bhyverun.c の解説

libvmmapiはvmm.koへのioctl・sysctlを抽象化したライブラリで、/usr/sbin/bhyve・/usr/sbin/bhyvectlはこれを呼び出すことによりvmm.koへアクセスします（**リスト1**）。

リスト2 bhyverun.cは/usr/sbin/bhyveの中心になるコードです。

▼リスト1　lib/libvmmapi/vmmapi.c

```
…… (省略) ……
280 int
281 vm_run(struct vmctx *ctx, int vcpu, uint64_t rip, struct vm_exit *vmexit)
282  {
283    int error;
284    struct vm_run vmrun;
285
286    bzero(&vmrun, sizeof(vmrun));
287    vmrun.cpuid = vcpu;
288    vmrun.rip = rip;
289
290    error = ioctl(ctx->fd, VM_RUN, &vmrun);
291    bcopy(&vmrun.vm_exit, vmexit, sizeof(struct vm_exit));
292    return (error);
293 }
```

①前回の記事の最後でユーザランドへreturnされたioctlはここに戻ってくる

②vmm.koから渡されたvmexit情報をコピーしてコール元へ渡す

▼リスト2　usr.sbin/bhyve/bhyverun.c

```
…… (省略) ……
294 static int
295 vmexit_inout(struct vmctx *ctx, struct vm_exit *vme, int *pvcpu)
296 {
297  int error;
298  int bytes, port, in, out;
299  uint32_t eax;
300  int vcpu;
301
302  vcpu = *pvcpu;
303
304  port = vme->u.inout.port;
305  bytes = vme->u.inout.bytes;
306  eax = vme->u.inout.eax;
```

⑥VMExit時にvmm.koが取得した、in/out命令のエミュレーションに必要な情報（ポート番号、アクセス幅、書き込み値（読み込み時は不要）、I/O方向（in/out））を展開する

```
307   in = vme->u.inout.in;
308   out = !in;
309
……(省略)……
322   error = emulate_inout(ctx, vcpu, in, port, bytes, &eax, strictio);
323   if (error == 0 && in)
324     error = vm_set_register(ctx, vcpu, VM_REG_GUEST_RAX, eax);
325
326   if (error == 0)
327     return (VMEXIT_CONTINUE);
328   else {
329     fprintf(stderr, "Unhandled %s%c 0x%04x\n",
330       in ? "in" : "out",
331       bytes == 1 ? 'b' : (bytes == 2 ? 'w' : 'l'), port);
332     return (vmexit_catch_inout());
333   }
334 }
……(省略)……
508 static vmexit_handler_t handler[VM_EXITCODE_MAX] = {
509   [VM_EXITCODE_INOUT] = vmexit_inout,
510   [VM_EXITCODE_VMX] = vmexit_vmx,
511   [VM_EXITCODE_BOGUS] = vmexit_bogus,
512   [VM_EXITCODE_RDMSR] = vmexit_rdmsr,
513   [VM_EXITCODE_WRMSR] = vmexit_wrmsr,
514   [VM_EXITCODE_MTRAP] = vmexit_mtrap,
515   [VM_EXITCODE_PAGING] = vmexit_paging,
516   [VM_EXITCODE_SPINUP_AP] = vmexit_spinup_ap,
517 };
518
519 static void
520 vm_loop(struct vmctx *ctx, int vcpu, uint64_t rip)
521 {
……(省略)……
532   while (1) {
533     error = vm_run(ctx, vcpu, rip, &vmexit[vcpu]);
534     if (error != 0) {
535       /*
536       * It is possible that 'vmmctl' or some other process
537       * has transitioned the vcpu to CANNOT_RUN state right
538       * before we tried to transition it to RUNNING.
539       *
540       * This is expected to be temporary so just retry.
541       */
542       if (errno == EBUSY)
543         continue;
544       else
545         break;
546     }
547
548     prevcpu = vcpu;
549     rc = (*handler[vmexit[vcpu].exitcode])(ctx, &vmexit[vcpu],
550               &vcpu);
551     switch (rc) {
552     case VMEXIT_SWITCH:
553       assert(guest_vcpu_mux);
554       if (vcpu == -1) {
555         stats.cpu_switch_rotate++;
556         vcpu = fbsdrun_get_next_cpu(prevcpu);
557       } else {
558         stats.cpu_switch_direct++;
559       }
560       /* fall through */
561     case VMEXIT_CONTINUE:
```

⑦デバイスエミュレータを呼び出す

⑯in命令だった場合は読み込んだ結果がゲストのraxレジスタにセットされる。今回はoutなのでここを通らない

⑰VMEXIT_CONTINUEがreturnされる

⑤VM_EXITCODE_INOUTでVMExitしてきているのでvmexit_inout()が呼ばれる

⑲whileループで再びvm_run()が実行され、ゲストマシンが再開される

③ioctlから抜け、ここに戻ってくる

④EXITCODEに対応したハンドラを呼び出す。ここではin/out命令の実行でVMExitしてきたものとして解説を進める

```
562     rip = vmexit[vcpu].rip + vmexit[vcpu].inst_length;   ⑱ゲストのripを1命令先に進める
563     break;
564   case VMEXIT_RESTART:
565     rip = vmexit[vcpu].rip;
566     break;
567   case VMEXIT_RESET:
568     exit(0);
569   default:
570     exit(1);
571   }
572 }
573 fprintf(stderr, "vm_run error %d, errno %d\n", error, errno);
574 }
…… (省略) ……
757: }
```

inout.c

inout.cはI/O命令エミュレーションを行うコードです。実際にはI/Oポートごとの各デバイスエミュレータのハンドラを管理する役割を担っており、要求を受けるとデバイスエミュレータのハンドラを呼び出します。呼び出されたハンドラが実際のエミュレーション処理を行います。

▼リスト3　usr.sbin/bhyve/inout.c

```
…… (省略) ……
72 int
73 emulate_inout(struct vmctx *ctx, int vcpu, int in, int port, int bytes,
74     uint32_t *eax, int strict)
75 {
76   int flags;
77   uint32_t mask;
78   inout_func_t handler;
79   void *arg;
80
81   assert(port < MAX_IOPORTS);
82
83   handler = inout_handlers[port].handler;   ⑧ポート番号ごとに登録されているI/Oポートハンドラを取り出す
84
85   if (strict && handler == default_inout)
86     return (-1);
87
88   if (!in) {
89     switch (bytes) {
90     case 1:
91       mask = 0xff;
92       break;
93     case 2:
94       mask = 0xffff;
95       break;
96     default:
97       mask = 0xffffffff;
98       break;
99     }
100    *eax = *eax & mask;
101  }
102
```

```
103  flags = inout_handlers[port].flags;
104  arg = inout_handlers[port].arg;
105
106  if ((in && (flags & IOPORT_F_IN)) || (!in && (flags & IOPORT_F_OUT)) )
107    return ((*handler)(ctx, vcpu, in, port, bytes, eax, arg));
108  else
109    return (-1);
110 }
…… (省略) ……
141 int
142 register_inout(struct inout_port *iop)
143 {
144  assert(iop->port < MAX_IOPORTS);
145  inout_handlers[iop->port].name = iop->name;
146  inout_handlers[iop->port].flags = iop->flags;
147  inout_handlers[iop->port].handler = iop->handler;
148  inout_handlers[iop->port].arg = iop->arg;
149
150  return (0);
151 }
```

⑨ポート番号ごとに登録されているハンドラを取り出す

⑩I/Oポートハンドラはregister_inout()で登録されている

consport.c

consport.cはBHyVe専用の準仮想化コンソールドライバです。現在はUART（Universal Asynchronous Receiver Transmitter）エミュレータが導入されたので必ずしも使う必要がなくなったのですが、デバイスエミュレータとしては最も単純な構造をしているので、デバイスエミュレータの例として取り上げました。

▼リスト4　usr.sbin/bhyve/inout.c

```
…… (省略) ……
95 static void
96 ttywrite(unsigned char wb)
97 {
98  (void) write(STDOUT_FILENO, &wb, 1);
99 }
100
101 static int
102 console_handler(struct vmctx *ctx, int vcpu, int in, int port, int bytes,
103    uint32_t *eax, void *arg)
104 {
105  static int opened;
106
107  if (bytes == 2 && in) {
108    *eax = BVM_CONS_SIG;
109    return (0);
110  }
111
112  if (bytes != 4)
113    return (-1);
114
115  if (!opened) {
116    ttyopen();
117    opened = 1;
118  }
119
120  if (in)
```

⑮ttywrite()はwrite()で標準出力に文字を書き込む

⑬console_handler()ではI/O方向がinならttyread()、outならttywrite()を実行し、標準入出力に対してI/Oを行う

```
121    *eax = ttyread();
122   else
123    ttywrite(*eax);
124
125   return (0);
126 }
127
128 static struct inout_port consport = {
129   "bvmcons",
130   BVM_CONSOLE_PORT,
131   IOPORT_F_INOUT,
132   console_handler
133 };
134
135 void
136 init_bvmcons(void)
137 {
138
139   register_inout(&consport);
140 }
```

⑭今回はoutが実行された場合を見ていく。eaxで指定された書き込み値をttywrite()に渡している

⑫登録するハンドラ関数としてconsole_handler()が指定されている

⑪consportデバイスは起動時にここでハンドラを登録している

まとめ

I/O命令によるVMExitを受けて行われるユーザランドでのエミュレーション処理について、ソースコードを解説しました。今回までで、ハイパーバイザの実行サイクルに関するソースコードの解説を一通り行ったので、次回はvirtioのしくみについて見ていきます。SD

テキストデータならお手のもの

開眼シェルスクリプト

㈲ユニバーサル・シェル・プログラミング研究所　http://www.usp-lab.com
上田 隆一　UEDA Ryuichi　Twitter @uecinfo

第19回 CGIスクリプトを作る(1)――Webサーバへのデータは標準出力で渡す

シェルスクリプトでCGIスクリプトを作れるのか

今回から何回かは、連載当初からいつかやることになると考えていたCGIスクリプトを作るというお題を扱います。

CGIというのはCommon Gateway Interfaceの略で、単純に言うとブラウザからWebサーバに置いてあるプログラムを起動するための仕様です。CGIという言葉はInterfaceを指すので、CGIで動く(動かされる)プログラムのことは、CGIプログラムやCGIスクリプトというほうが丁寧です。スクリプト言語で書いた場合はCGIスクリプトというのが良いでしょう。

CGIプログラムは、どんな言語で作ってもかまいません。C言語で書いても良いわけですが、この領域ではLightweight Language(LL言語)で書かれることがほとんどであり、伝統的にはPerl、PHPが多く、最近ではRuby、Pythonもよく使われます。

そこにシェルスクリプトを加えてやろうというのが今回から数回の内容です。シェルスクリプトは簡単にOSのコマンドが使えることから、(セキュリティ的に)大丈夫かいな、とよく言われます。確かにWebでデータをやりとりするという目的に比べると、シェルスクリプトでできることはそれをはるかに超越しており、しかも少ない文字数で邪悪なことができてしまいます。`rm -Rf /`(8文字!)とか。

しかし、「インジェクションを食らいやすいかどうか」という観点においては、気をつけていれば言語レベルではほかの言語と大差なく、むしろ食らいにくいんじゃないかな、と筆者は考えています。世間での書き方のガイドラインが未成熟なだけで、シェルスクリプトでCGIをやると危ないというのは短絡的な判断ですし、相手を知らなさ過ぎます。食わず嫌いはいけません。

女をよくいう人は、女を十分知らないものであり、女をいつも悪くいう人は、女をまったく知らないものである。――モーリス・ルブラン

この格言を`sed 's/女/シェルスクリプト/g'`して音読してから、先にお進みください。

Apacheを準備

今回、想定する環境はbash、Apacheが動くUNIX系の環境です。筆者は手元で動かしたいので今回もMacを使います。Linuxで動かす場合については情報が大量にWeb上にあるので、ここで説明しなくても大丈夫でしょう。

筆者もこれを執筆中に初めて知ったのですが、OS Xには最初からApacheがバンドルされていて、すぐ使えるようになっています。**図1**のようにコマンドを打つと、Apacheが起動します。

本連載の読者ならば、動作確認はブラウザではなく、**図2**のようにcurlでやりましょう。curlは、オプション指定されたURLからHTMLなどを受け取って標準出力に出力します。curlはたいていのLinuxディストリビューションのパッケージ管理ツールでインストールできます。

同様のツールにはwgetやfetchもあります。

次に**図3**のように、CGIを置くディレクトリを確認します。CGIプログラムはcgi-binというところに置くことが多いので、cgi-binで設定ファイル(httpd.conf)を検索します。

検索はエディタを開いて、そのエディタの機能で行ってもかまいません。ただ、こういった記事や説明手順を書くときは、シェルの操作を行ったような体裁のほうがわかりやすく書けます。さっきのcurlも、ブラウザのスクリーンショットを掲載するより楽です。きっとコミュニケーションのコストに違いがあるのでしょう。案外大事な余談でした。

確認の結果、/Library/WebServer/CGI-Executables/という汚い名前のディレクトリで動くことがわかりました。今回はたいへん遺憾ですが、ここにCGIスクリプトを置くことにします。いちいちこのディレクトリを覚えておくのは面倒ですので、**図4**のように自分のホームの下にシンボリックリンクをはりましょう。どうせ自分しか使わないので、所有者も変えておきます。

CGIプログラムとは「ただのプログラム」

さあ作業開始です。最初にやるのはCGIプログラムを動かすことです。CGIプログラムと聞くと何か特別なものだと考えている人が多いので、その誤解を解いておきましょう。ちょっとした実験をします。

まず、/tmp/の下にhogeというファイルを作り、所有者をApacheの実行ユーザに変えておきます。Apacheの実行ユーザ、そしてグループは次のように調査できます。

```
$ grep ^User /private/etc/apache2/httpd.conf
User _www
$ grep ^Group /private/etc/apache2/httpd.conf
Group _www
```

ユーザとグループがわかったら、次のようにhogeを置き、所有者を変更しましょう。

```
$ touch /tmp/hoge
$ sudo chown _www:_www /tmp/hoge
```

次にrmコマンドをcgi-binの下に置きます。拡張子は.cgiにしておきます。

```
$ cp /bin/rm ~/cgi-bin/rm.cgi
```

このrm.cgiをブラウザで呼び出してみます。これはcurlを使うと雰囲気が出ないので、ブラ

▼図1　Apacheを立ち上げる

```
$ sudo -s
# apachectl start
# ps cax | grep httpd
16023   ??  Ss     0:00.15 httpd
16024   ??  S      0:00.00 httpd
```

▼図2　curlで動作確認

```
$ curl http://localhost
<html><body><h1>It works!</h1></body></html>
```

▼図3　cgi-binの場所を調査

```
$ apachectl -V | grep conf
 -D SERVER_CONFIG_FILE="/private/etc/apache2/httpd.conf"

$ cat /private/etc/apache2/httpd.conf | grep cgi-bin
    ScriptAliasMatch ^/cgi-bin/((?!(?i:webobjects)).*$) "/Library/WebServer/ ➪
CGI-Executables/$1"
#ErrorDocument 404 "/cgi-bin/missing_handler.pl"
```

▼図4　ホームから簡単にアクセスできるようにする

```
$ ln -s /Library/WebServer/CGI-Executables/ ./cgi-bin
$ cd cgi-bin
$ sudo chown ueda:wheel ./
```

ウザで試します。アドレスの欄には、http://localhost/cgi-bin/rm.cgi?/tmp/hogeと入力します。

ブラウザに表示されるのは、残念ながら**図5**のようなInternal Server Errorです。

しかし、/tmp/hogeは、次に示すように消えています。

```
$ ls /tmp/hoge
ls: /tmp/hoge: No such file or directory
```

びっくりしましたか？

結局、何をやったかというと、ブラウザにhttp://localhost/cgi-bin/rm.cgi?/tmp/hoge を指定することで、サーバ(この例では自分のMac)のcgi-binの下のrm.cgiのオプションに、/tmp/hogeを渡して/tmp/hogeを消したということになります。sshでリモートのサーバに対し、

```
$ ssh <ホスト> '~/cgi-bin/rm.cgi /tmp/hoge'
```

とやることと何ら変わりがありません。違うのは22番ポートでなく、80番ポートを使用したくらいです。ただし、rmコマンドをインターネット上から不特定多数の人に実行されたらたまったものではないので、Apacheでは、

- UserやGroupで実行するユーザを限定
- 実行できるプログラムを特定のディレクトリの下のものに制限
- 拡張子を登録したものだけに制限

するなど、一定の制約を設けてなるべく安全にしてあります。

逆に、~/cgi-bin/の下に置いて実行可能なようにパーミッションを設定すれば、プログラムはなんでもCGIで起動できるようになります。rm.cgiのようにC言語で書いてあっても、伝統的なPerlで書いても動きます。ということは、シェルスクリプトでも動くということになります。

▼図5　rm.cgiを実行した結果

CGIシェルスクリプトを書く

では、シェルスクリプトでCGIスクリプトを書いてみましょう。まず、ブラウザに字を表示するための最小限のCGIスクリプトを**図6**に示します。このシェルスクリプトは何の変哲もないものですので、普通に端末から実行できます。

▼図6　最小限のCGIスクリプト

```
$ cat smallest.cgi
#!/bin/bash -xv

echo "Content-Type: text/html"
echo ""
echo 魚眼perlスクリプト
   書いたら実行できるようにしておきましょう
$ chmod +x smallest.cgi
```

```
$ ./smallest.cgi 2> /dev/null
Content-Type: text/html

魚眼perlスクリプト
```

ブラウザから呼び出すと**図7**のように見えます。この例のポイントはいくつかあります。まず、`Content-Type-type: text/html`ですが、これはHTTPプロトコルで定められたHTTPヘッダです。先ほどのrm.cgiでブラウザにエラーが出たのは、HTTPヘッダをrm.cgiが出さないからです。ブラウザとApacheはHTTPプロトコルでしゃべっているので、Apache(が動かしているCGIプログラム)がHTTPヘッダを返さず、ブラウザが怒ったのでした。

▼図7　ブラウザからsmallest.cgiを実行した結果

ヘッダの次のecho ""は、ヘッダと中身を区切る空白行を出すためにあります。ヘッダの前には余計なものを出してはいけないので、たとえば図8のようなCGIスクリプトをブラウザから呼び出すと、やはりブラウザにエラーが表示されます。

図6ではContent-Type-type: text/htmlと「テキストのHTML」を送ると言っておいて、実際には単なる1行のテキストしか送っていませんが、これは今のところこだわらないでおきましょう。

次に着目すべきは、シェルスクリプトはただ標準出力に字を出しているだけで、ブラウザやWebサーバに何か特別なことをしているわけではないということです。これはApacheがシェルスクリプトの出力を受け取ってブラウザに投げるからです。シェルスクリプトの側ですべきことは、正確なHTTPヘッダの出力だけということになります。いかにもUNIXらしい動きです。

シバン(#!/bin/bash)の行にログを出力する-xvというオプションを付けましたが、このログはどこに行くのでしょうか。実は図9のように、Apacheのエラーログに行きます。

今、挙げたポイントは、別のLL言語でもまったく同じことです。違うのは、LL言語には便利なライブラリが存在していて、Webサーバとのダイレクトなやりとりがちょっとだけ隠蔽されていることです。でもまあ、何を使おうが普通のCGIの場合、最終的にはHTTPでHTMLやJavaScriptを出力することになります。

出力系のCGIスクリプトを作る

ブラウザにテキストデータを渡して処理させる

さて、シェルスクリプトでCGIスクリプトが作れるとわかったので、さっそく何か作ってみましょう。実用的なものは次回以降にまわすとして、今回は何かおもしろいものを作ってみましょう。

まずは、ブラウザから呼ばれたときに次のような動きをするCGIスクリプトを作ります。

- ブラウザ側(クライアント側)を待たせる
- サーバ側でCGIスクリプトに文字列を渡すと、ブラウザにその文字が表示される

▼図8　HTTPヘッダの前に何か出力するとエラーになる

```
$ cat dame.cgi
#!/bin/bash -xv

echo huh?
echo "Content-Type: text/html"
echo ""
echo 湾岸pythonスクリプト
$ curl http://localhost/cgi-bin/dame.cgi 2> /dev/null | head -n 3
<!DOCTYPE HTML PUBLIC "-//IETF//DTD HTML 2.0//EN">
<html><head>
<title>500 Internal Server Error</title>
```

▼図9　error_logにCGIスクリプトの標準エラー出力がたまる

```
$ cat /private/var/log/apache2/error_log
 (略)
[Tue Apr 23 21:46:14 2013] [error] [client ::1] #!/bin/bash -xv
[Tue Apr 23 21:46:14 2013] [error] [client ::1]
[Tue Apr 23 21:46:14 2013] [error] [client ::1] echo "Content-Type: text/html"
[Tue Apr 23 21:46:14 2013] [error] [client ::1] + echo 'Content-Type: text/html'
[Tue Apr 23 21:46:14 2013] [error] [client ::1] echo ""
[Tue Apr 23 21:46:14 2013] [error] [client ::1] + echo ''
[Tue Apr 23 21:46:14 2013] [error] [client ::1] echo ??? (略)
```

とくに何か用途があるわけではないのですが、CGIスクリプトが一瞬で終わらずに途中で一度止まるので、動いていることを実感できると思います。では、**リスト1**のようなシェルスクリプトを作ります。

3行目のmkfifoというコマンドは、「名前つきパイプ」という特別なファイルを作るコマンドです。「名前つきパイプ」は、その名のとおりパイプでして、片方から字を突っ込むと、もう片方から字が出てきます。たとえば、

```
$ echo hoge | cat
```

という処理を名前付きパイプで書くと次のようになります。

▼リスト1　notify.cgi

```
01 #!/bin/bash
02
03 mkfifo /tmp/pipe
04 chmod a+w /tmp/pipe
05
06 echo "Content-Type: text/html"
07 echo ""
08 cat /tmp/pipe
09 rm /tmp/pipe
```

▼リスト2　notify2.cgi

```
01 #!/bin/bash
02
03 mkfifo /tmp/pipe
04 chmod a+w /tmp/pipe
05
06 echo "Content-Type: text/plain"
07 echo ""
08 cat /tmp/pipe
09 rm /tmp/pipe
```

▼図11　ブラウザでアラートが表示される

▼図10　送り込む文字列

```
$ echo '<script>alert("no more XSS!!")</script>' > /tmp/pipe
```

```
端末1
$ cat /tmp/pipe
端末2
$ echo hoge > /tmp/pipe
```

こうすると、端末1のcatは/tmp/pipeに何か字が流れてくるまで止まった状態になり、端末2でecho hogeが実行されたらhogeと出力します。echo hogeが終わると、catも終わります。よくよく考えると、この動作は普通のパイプのものと同じです。ただし、/tmp/pipeはrmで消さない限り残ります。

4行目のchmodは、/tmp/pipeの所有者以外でも書き込めるようにするためのパーミッション変更です。

さて、notify.cgiをブラウザから呼び出してみましょう。CGIスクリプトは**cat /tmp/pipe**でいったん止まるので、ブラウザは待ちの状態になります。

次に、おもむろに端末から**図10**のように打ってみてください[注1]。**図11**のようにアラートが出たら成功です。何の役にも立たないですが、多分、おもしろいと思っていただけたかと。

ちなみに、HTTPヘッダがちゃんと意味があるということを示すために、notify.cgiを**リスト2**のように書き換えて(Content-Typeをtext/plainにしました)、もう一度やってみます。

今度は、ブラウザに「<script>alert("no more XSS")</script>」と文字列が表示されたと思います。まともなブラウザならば……。

ファイルをダウンロードさせる

HTTPヘッダの話が出たので、最後にファイルのダウンロードでもやってみましょう。たとえばみんな大好きExcelファイルのダウンロードを行うCGIスクリプトは、**リスト3**のように

注1) /tmp/pipeのないときにやってしまうと、/tmp/pipeという普通のファイルができてしまうので注意してください。

書けます。

6行目のapplication/octet-streamは、「バイナリを送り込むぞ」という宣言、7行目は「hoge.xlsxという名前で保存してくれ」、8行目は変数LENGTHに書いてあるサイズのデータを出力するぞ、という意味になります。

そして、実際にファイルをブラウザに向けて発射するには、10行目のようにお馴染みのcatを使います。catはテキストもバイナリも区別しません。区別してしまうとほかのコマンドと連携して使えなくなってしまいます。

ファイルはありとあらゆるものがダウンロードさせられますが、ヘッダは微妙に変化させます。たとえば、mpegファイルをブラウザで直接見せたいのなら**リスト4**のように書きます。

筆者の普段使っているブラウザ(MacのGoogle ChromeとFirefox)では、**図12**のようにブラウザのプラグインが立ち上がり、画面内でムービーが再生されます。

ヘッダに`Content-Disposition: attachment; filename="hoge.mpeg"`を加えると、ファイルを再生するかファイルに保存するか聞いてきたり、再生されずにファイルに保存されたりします。

筆者のHTTPヘッダについての知識はこの程度ですが、もし別の言語でHTTPヘッダを間接的にいじったことのある人は、シェルスクリプトでも細かい制御ができることでしょう。

終わりに

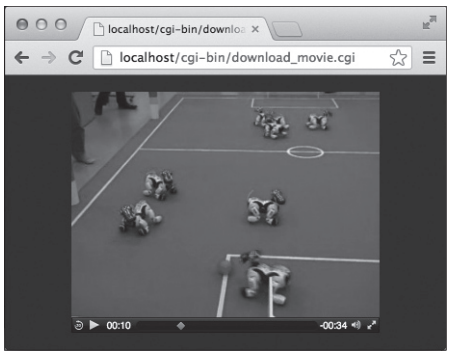
今回はシェルスクリプトでCGIスクリプトを書きました。とくに出力について扱いました。おそらく今回の内容で一番重要なのは、Apacheを経由してブラウザにコンテンツを送るときには、標準出力を使うということでしょうか。ここらあたりにも、インターネットがUNIXとともに発展してきた名残があります。いや、名残というよりも必然かもしれません。標準入出力は、これ以上ないくらい抽象化されたインターフェースであり、まず最初に使用を検討すべきものでしょう。

次回はCGIスクリプトでのPOST、GETも絡めて何かを作ってみようと考えています。SD

▼リスト4　mpegファイルを見せるためのCGIスクリプト(download_movie.cgi)

```
#!/bin/bash

FILE=/tmp/japanopen2006_keeper.mpeg
LENGTH=$(wc -c $FILE | awk '{print $1}')

echo "Content-Type: video/mpeg"
echo "Content-Length: $LENGTH"
echo
cat $FILE
```

▼図12　ヘッダを適切に書くとブラウザでよしなに取りはからってくれる



▼リスト3　ファイルをダウンロードさせるCGIスクリプト

```
01 #!/bin/bash -xv
02
03 FILE=/tmp/book1.xlsx
04 LENGTH=$(wc -c $FILE | awk '{print $1}')
05
06 echo "Content-Type: application/octet-stream"
07 echo 'Content-Disposition: attachment; filename="hoge.xlsx"'
08 echo "Content-Length: $LENGTH"
09 echo
10 cat $FILE
```

短期集中連載

SDNは使えるのか、使うべきなのか？

仮想ネットワークの落とし穴

第2回

株式会社データホテル
伊勢幸一（いせこういち）
Twitter @ ibucho

エンドポイントモデルの検証
――VXLAN、NVGRE、STT、独自拡張問題

ファブリックからエンドポイントモデルへ

本来、クラウド基盤等においてネットワークを仮想化する目的は2つ挙げられます。数多くのテナントを1つのネットワークシステム上で展開するとき、IEEE802.1Qにおける最大VLAN数（4,094）以上のテナントセグメントを作成することと、テナントの仮想マシン（以後VM：Virtual Machine）がそのIPアドレスやMACアドレスを変更することなく、IaaS基盤上で自由に移動、起動、停止できるネットワークを形成することです。

前回取り上げたファブリックモデルによるネットワークの仮想化は、データリンク層における多重化や仮想セグメントを提供しますが、ネットワーク層以上に対する仮想化までには対応していません。ネットワーク層以上を仮想化し、VLAN数の上限問題と柔軟なネットワークセグメントを形成する仮想化方式がエンドポイントモデルです。

エンドポイントモデルは既存の物理ネットワークをそのまま活用しつつ、その物理トポロジーに依存しない論理的なネットワークセグメントを形成する技術であり、多くの場合、従来のIPネットワーク網上にトンネル技術によって仮想的なEthernet L2セグメントをオーバーレイすることを指します。また、単にトンネリングによってオーバーレイネットワークを形成するだけではなく、テナントセグメントを識別するIDをトンネルヘッダ内で新たに定義し、4094以上のテナントセグメントを作成することも可能にしています。

一般的には物理サーバ上の仮想ハイパーバイザがエンドポイントの役割を担う形式と、物理スイッチがトンネルエンドポイントとなる形式があります（**図1**）。物理スイッチをすべてのトンネルのエンドポイントとして利用するにはコストパフォーマンスに問題があり、比較的に非仮想ネットワークと仮想ネットワークとのゲートウェイとして利用されるケースに限られます。もう1つのエンドポイントモデルとして仮想スイッチの機能をNICか、もしくは隣接するスイッチに処理をオフロードするVEB（VIirtual Ethernet Bridge）と VEPA（Virtual Ethernet Port Aggregator）がありますが、これらはVLAN数上限の拡張やL2オーバーレイを形成するものではないためここでは割愛します。

現在、代表的なエンドポイントモデルのプロトコルにはVXLAN（Virtual eXtensible Local Area Network）、NVGRE（Network Virtualization using Generic Routing Encapsulation）、STT（Stateless Transport Tunneling）があります。前回同様、これら3種類のプロトコルにつ

いて、それぞれの概要とデメリット、致命的欠陥について検討してみましょう。

VXLAN

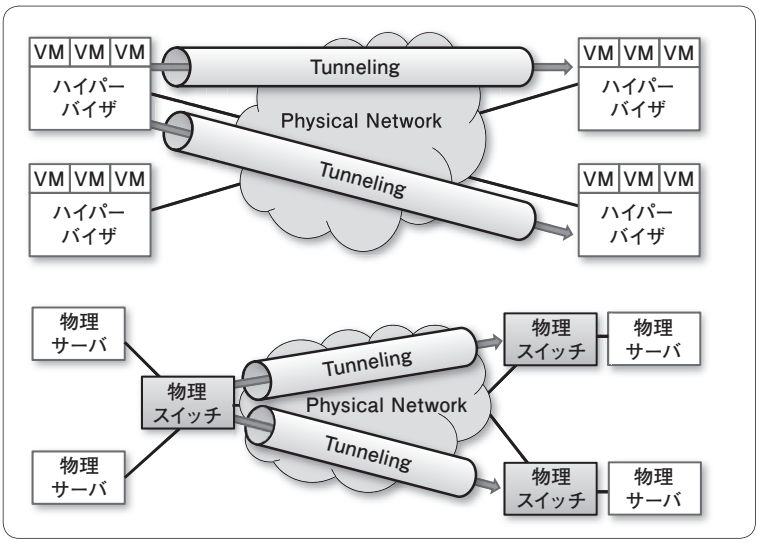
VXLANはVMware、Cisco Systems、Alista、Broadcom、Citrix Systems、Red Hatの6社によって提案されている仮想ネットワークプロトコルです。2011年8月に最初のIETFドラフトが公開されました。**図2**に示すようにVXLANはオリジナルのEthernetフレームをUDPパケットにカプセリングして既存のIP網上を搬送し、同時にVXLANの独自ヘッダにある24ビット長のセグメントID（VNI：VXLAN Network ID）によって論理的VLAN数を約1,600万以上に拡大しています。

VXLANによるカプセリング機能はカーネルのUDPサービスが利用できるため、カーネル内に組み込むこともユーザランドアプリケーションとして実装することもできます。パフォーマンスを考慮するとカーネルへの実装が望まれますが、UDPサービスのオーバーヘッドはほとんどないため、実装と開発工数の軽快さを考えるとユーザランドでの実装でも問題はないでしょう。VXLANヘッダも実質フラグとVNIしか持たないので、トンネル処理によるオーバーヘッドも非常に少ないプロトコルです。またVXLANヘッダは新たに定義された独自フォーマットとは言え、UDPペイロードの内側にあるため、インターネットプロトコルスタック上では、アプリケーションデータとみなされます。したがって、VXLANフレームは途中経路にあるネットワーク機器のフォワーディング処理に影響を与えることはなく、物理サーバのNICや既存の物理ネットワーク機器、NATやファイアウォールなどのネットワーク構成要素を従来のままで運用することができます。

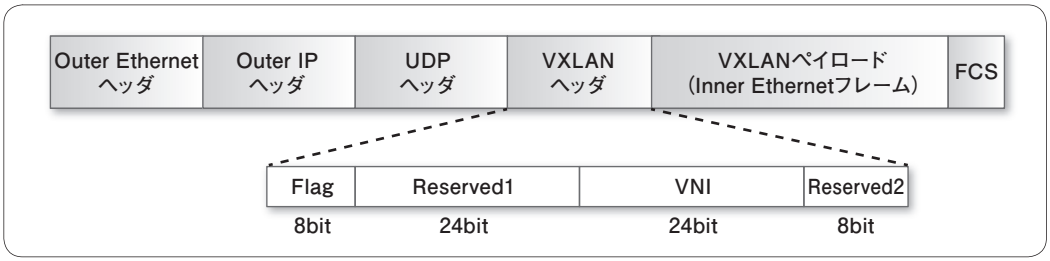
MTU問題

トンネルプロトコル全般に言えることですが、

▼図1　エンドポイントモデル

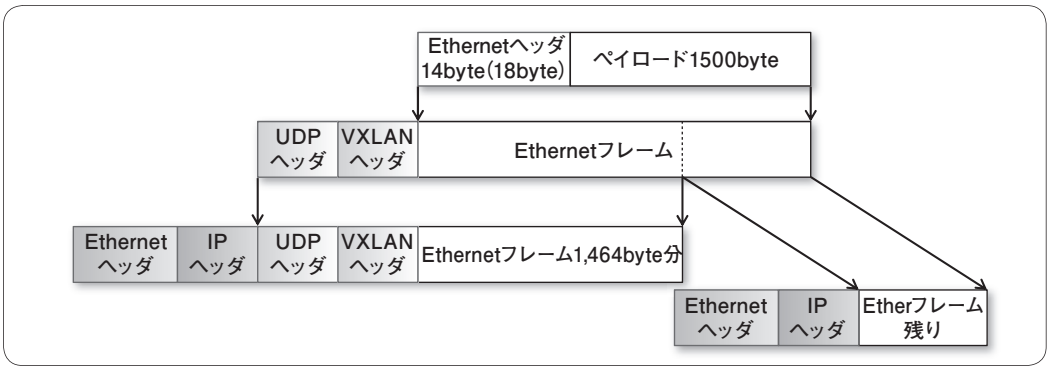


▼図2　VXLANフレームとヘッダ



仮想ネットワークの落とし穴

▼図3　IPフラグメンテーションによる分割搬送



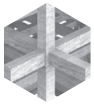
搬送用Ethernetフレームのペイロードに UDPやVXLANというトンネルヘッダを差し込むため、元のEthernetフレームのペイロードが1,500バイトである場合、1つの搬送用Ethernetのペイロードに格納することはできません。したがって最大ペイロード長を持つ元のEthernetフレームは、複数のフレームに分割されて搬送されることになります(**図3**)。

したがって元のフレームの復元に関してはIPレベルでの再組立とUDPレベルでのチェックサムによってある程度保証はされますが、クラウド基盤上で交換されるフレームのほとんどがペイロード1,500バイトのフルフレームである場合、ネットワークの性能に著しい影響を及ぼす可能性があります。ネットワーク機器のポート性能は送受信するデータ量で示されますが、フォワーディング性能はパケット量に依存します。つまり、機器の内部で処理されるフォワーディングは各プロトコル層のヘッダを処理し、転送先インターフェースを決定するのであり、元のフレームが常に2分割で搬送されるとすると、単純にこれら機器のフォワーディング性能は半分になります。

VXLANを実装しているベンダでは、このIPフラッグメンテーション問題に対し、物理ネットワークのMTUを拡大してVMが認識するMTUサイズを1,600バイト以上にするか、もしくは各VMのMTUを1,400バイトにするかを推奨しています。Ethernet MTUサイズの拡大をサポートする機器はありますが、すべてのスイッチやルータがサポートしているわけではありません。

物理ネットワークがサポートするMTUを拡大するということはデータリンクプロトコルの互換性と設備投資負担の問題があるため、現実的にはVMのMTUを縮小する対策が選択されるでしょう。その場合、ゲストOSイメージのすべてを縮小したMTUに設定するか、Chefなどの管理ツールを用いてVMが立ち上がるときにMTUを調整するしくみを施すことになります。いずれにしても安定運用まで最適化するには相当な労力と負担が強いられることでしょう。

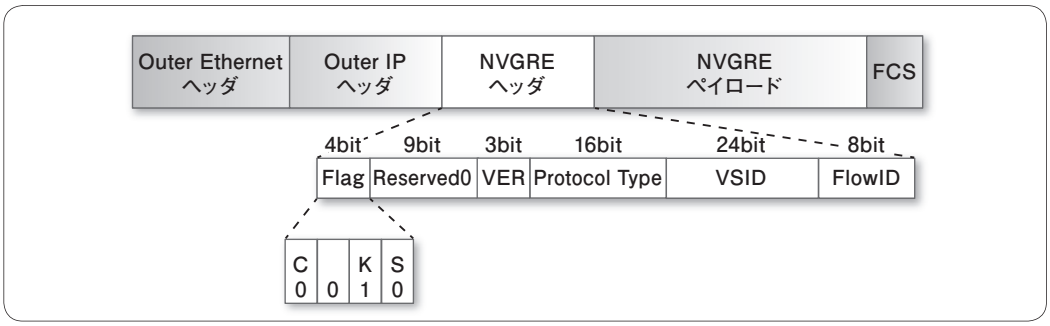
NVGRE



NVGREはMicrosoft、Alista、Intel、Dell、HP(ヒューレットパッカード)、Broadcom、Emulexの7社によって提案されているプロトコルであり、VXLANと同時期にIETFドラフトが公開されました。NVGREはRFC2784、RFC2890で標準化されているGRE(General Routing Encapsulation)プロトコルのキーオプションフィールドを拡大解釈することによって仮想ネットワークを実現する技術です。NVGREはVMが送出するEthernetフレームをNVGREヘッダと搬送用IPヘッダでカプセリングして物理ネットワーク上に転送します。

前述したように、NVGREヘッダフォーマッ

▼図4　NVGREフレームとヘッダ



トはGREヘッダフォーマットとまったく同じです。ヘッダの先頭にある4ビットフラグはMSBからLSBに向かって、チェックサムビット、未使用、キービット、シーケンスビットとしてそれらオプションの有無を示しますが、NVGREではキーオプションだけ1にセットされ、それ以外はすべて0にセットされることになります(**図4**)。

また、32ビットのキーオプションフィールドは上位24ビットをNVGREセグメントを示すVSID(Virtual Subnet ID)として利用されます。これはVXLANにおけるVNIに相当します。また下位8ビットは物理ネットワークでのECMPで利用されるフローIDが定義されます。NVGREはキーオプションの解釈が異なるだけで、ヘッダフォーマットもプロトコル番号もGREとまったく同じであるため、ネットワーク機器やOSにとってGREパケットとNVGREパケットの区別はつきません。したがって既存のネットワーク機器やサーバのNIC、OSのEthernetやIPのドライバーなどへの変更はいっさい必要ありません。

ただし、GREは通常サーバOSのカーネル内で処理されるため、NVGREをサポートするにはカーネルもしくはハイパーバイザのソースコードに手を加える必要があり、実装とメンテナンスの負担はVXLANに比べて大きくなります。

チェックサム問題

NVGREにも当然MTUにまつわる問題がありますが、この場合、ほかのプロトコルに比べて状況はかなり深刻です。アウターIPのヘッダ長が基本の20バイトであるとすると、GREヘッダを含むIPペイロードが1,481バイト以上ある場合、当然のことながら送信側でIPフラッグメンテーションが発生します。受信側では分割されたIPパケットを再組立するわけですが、NVGREヘッダはチェックサムオプションを含まないため、復元されたIPペイロードの正当性を確認する術がありません。

通常、Ethernetフレームには最後にフレームの正当性を確認するためのFCS(Frame Check Sequence)が添付されていますが、このFCSは(ほとんどの場合)NIC上でハードウェア的に処理されるため、NICはFCSを除去した状態で上位ドライバに渡し、また上位ドライバーはFCSのないフレームをNICに渡すため、ハイパーバイザやVMではFCSを利用して元のフレームの正当性をチェックすることができないのです。したがって、復元されたNVGREの正当性は保証されず、万が一、伝送途中でフレームがノイズを拾って壊れてしまった場合、その崩れたフレームをそのままVMに送り付けるとVMに障害を発生させる恐れがあります。

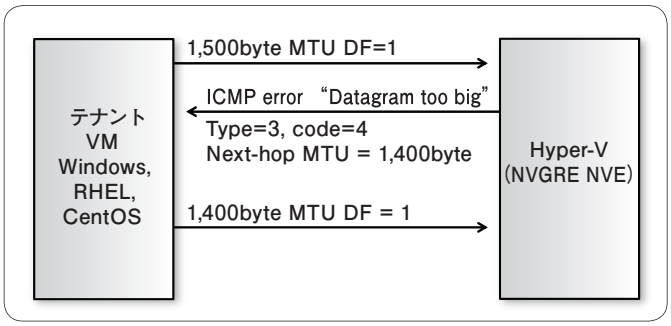
現在NVGREを実装しているMicrosoftのHyper-VではVMからペイロード1,500バイトのフレームが送出されてきた場合、ICMP Datagram too big エラーメッセージを返すと同時に、伝送可能なMTUサイズ(たとえば1,400バイト)をVMに通知し、NVGREパケットが伝

仮想ネットワークの落とし穴

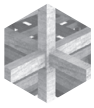
送途中でフラッグメンテーションされないようにしています(**図5**)。

このMTU問題に関してはVXLAN同様、すべてのVMイメージのMTUを最適化するか管理ツールによって調整するか、もしくはすでにNVGREとMTU調整が実装されているMicrosoft製品で統一するかという選択になり、いずれにしても運用負担の増加かベンダロックインを招く恐れがあります。

▼図5　ハイパーバイザによるMTU調整



STT

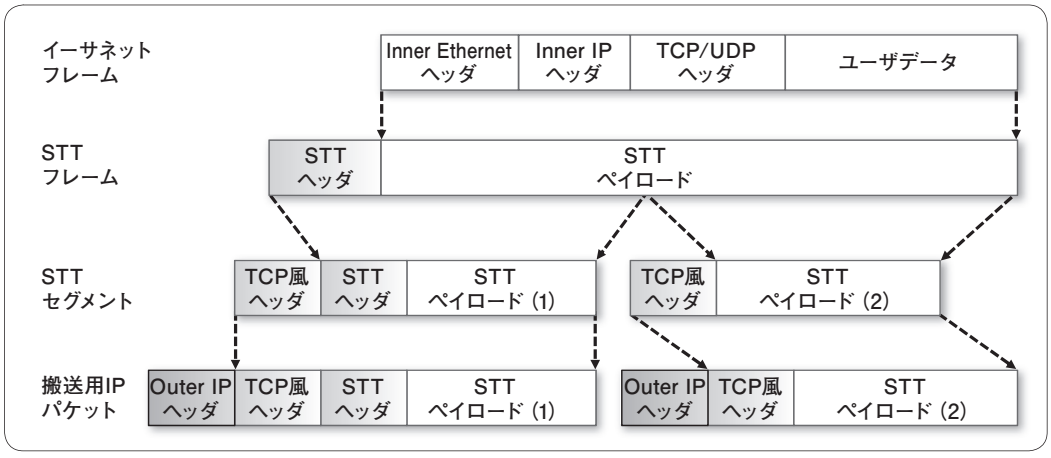


STTは現在VMwareに買収された旧Niciraによって同社製品のNVP(Network Virtualization Platform)に実装されていた独自のトンネルプロトコルですが、2012年1月にIETFドラフトとして公開されオープンプロトコルとなりました。STTは最大64KBのペイロードをTCPパケットに搭載したトンネル化によってL3ネットワーク上を搬送し、仮想オーバーレイネットワークを構築しています。

ほかのエンドポイントモデルは元のEthernetフレームをペイロードとすることを前提としていますが、STTはEthernetだけではなく、フレームリレーやATMなど、より大きいサイズのペイロードをサポートするデータリンクフレームも対象として設計されたようです。しかしクラウド基盤やデータセンター内では基本的にEthernetによるネットワークが利用されていることから64KBという大ペイロード長にそれほど意味はありません。この最大64KBという値は後述するSTTセグメントオフセット値が16ビットで表現されるためであり、とくに通信性能や各プロトコルの最大MTUサイズなどを考慮して決定されたものではありません。

STTではハイパーバイザが上位のVMから受け取ったフレームを、まずSTTヘッダによってカプセリングし、STTフレームを作成します。次にSTTフレームを物理ネットワーク上で搬送可能な断片に分割し、個々のセグメントを拡張されたTCPヘッダでカプセリングします(**図6**)。この断片化されたセグメントをSTTセグメント

▼図6　STTカプセリングシーケンス



と呼びます。STTセグメントはさらに物理ネットワークに依存したIPパケットによってカプセリングされ、送信元物理マシンから宛先仮想マシンが稼動している物理マシンへと運ばれるのです。

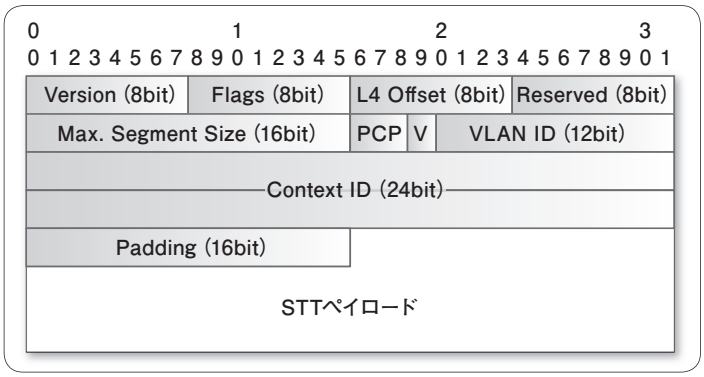
STTヘッダはVXLANやNVGREと比較すると複雑な構造をしており、**図7**で示すようにVersion、Flags、L4Offset、MaxSegumentSize、PCP、Vビット、VLAN ID、そしてContext IDという情報を持っています。64ビットのContext IDが一般的にテナントセグメントの識別子として利用されますが、その用途はテナントセグメントに限定されず、STTフレームを受け取ったエンドポイントがなすべき処理を示す汎用IDとして定義されています。VXLANのVNIやNVGREのVSIDに相当しますが、仮想ネットワークセグメントに対して1対1対応で割り当てることを規定していません。つまり、複数のセグメントに1つのIDを割り当てることも可能であり、逆に1つのセグメントに対して複数のContext IDを割り当て、同じテナントへのフレームに対して異なる処理を施すことも可能としています。

STTセグメントをカプセリングする拡張TCPヘッダは通常のTCPヘッダとまったく同じフォーマットをしていますが、RFC793で定義されているTCPヘッダのシーケンス番号と応答確認番号フィールドをそれぞれSTTフレーム長、STTセグメントオフセットとSTTフレーム識別子として利用し、独自の処理を行います(**図8**)。STTフレーム識別子とはフレーム単位に割り当てられ、アンダーレイネットワークで搬送可能な単位まで分割されたSTTセグメントを再組立する際に、どのフレームのセグメントであるかを判別するために利用されます。

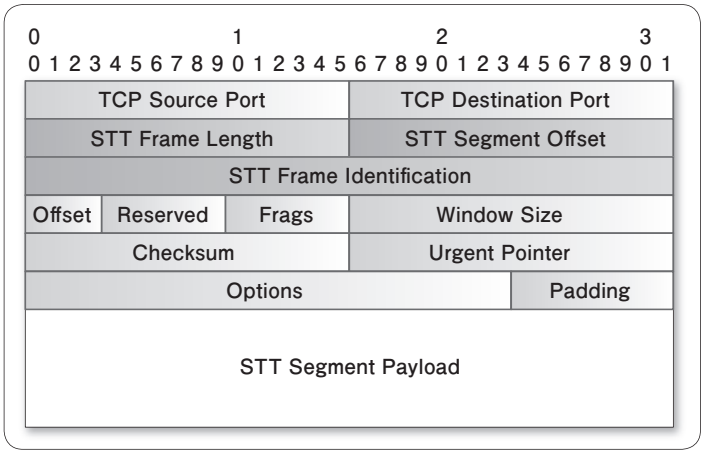
拡張TCPヘッダにある6ビットのフラグフィールドはRFC793での定義と同じですが、応答番号が有効であることを示すACKビットと、受け取ったデータを上位アプリケーションに直ちに送り出すことを指示するPSHビットは常に1にセットし、そのほかのビットは0にするべきとされています。

STTはペイロード長が64KBもあり、最大1,514バイトのIEEE802.3フレームや最大1,518バイトのIEEE802.1Qフレームは十分格納されるので、一見、MTU問題がないかのように思えますが、STTフレームは結局TCPでカプセリングされる際、アンダーレイネットワー

▼図7　STTヘッダフォーマット



▼図8　拡張TCPヘッダフォーマット



仮想ネットワークの落とし穴

クのMSS（Maximun Segment Size）以内に分割されるため、アンダーレイネットワークがEthernetIPであるならば結局IPのレイヤで分割されてしまいます。したがって、VXLANやNVGREと同様にフレームフラッグメンテーションによるネットワークパフォーマンスの低下を避けることはできません。

しかし、STTが持つ致命的欠陥は別の部分にあります。VXLANのトンネル処理は、実質フラグとVNIしかないVXLANヘッダとUDPヘッダだけです。NVGREのヘッダではVXLANに加えてヴァージョン、プロトコルタイプ、フローIDが加えられていますが、もともとGREヘッダを流用しているのでVXLANにおけるUDPのようなトランスポート層ヘッダ部分がなく、処理も軽減されています。それらをトレードオフするとNVGREもVXLANと同程度の処理ステップとなるでしょう。

図7で示したようにSTTヘッダフォーマットの内容はVXLANやNVGREに比べて非常に複雑な構成であり、VXLANと比較するとヴァージョン、L4オフセット、MSS、PCPとVビット、VLAN IDフィールド分が余計にあり、仮想ネットワークインスタンスが処理するフィールドは2倍以上です。さらにSTTセグメントはTCPヘッダによってカプセリングされるので、このTCPヘッダの処理負荷はUDPヘッダの比ではありません。

そこで、STTドラフトではこのTCPヘッダの処理をNICにオフロードすることによってパフォーマンスの低下を回避する対処を提案しています。つまり、プロトコルオーバーヘッドが大きいため、そのデメリットをハードウェアによって解決することを前提として設計されているのです。

ここでさらに問題が生まれます。STTセグメントは拡張TCPヘッダを利用してカプセリングされるとはいえ、その内容はRFC793に準拠した完全なTCPではなく、TCPヘッダの形式を利用した独自ヘッダとなっています。NVGREの場合、標準化されたGREのフォーマットをそのまま利用しているため途中経路にGREヘッダを解釈するノードがあったとしても通り抜けるフレームに施される処理に矛盾は発生しません。しかし、拡張TCPヘッダを持つSTTセグメントのプロトコル番号は通常のTCPと同じく6番を利用しているにもかかわらず、標準的なTCPとは完全互換ではないので、途中経路にTCPヘッダを解釈するノードがあった場合、そのヘッダ内容がTCPに完全準拠していないという理由で破棄される可能性があります。すると分割されたSTTフレームは受信先で再組み立てできなかったり、フレーム全体が受信先まで到達しなくなる可能性があります。

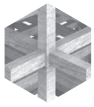
STTのドラフトでもこの問題を認識しており、将来的にはSTTセグメントのプロトコル番号をTCPとは別に割り当てて利用する対処を暗示しています。しかし、STTセグメントのプロトコル番号をTCPとは異なる番号にすると、今度はNICによるTCPヘッダ処理のオフロードができなくなってしまいます。STTプロトコルはもともとNICへのTCP処理オフロードによってパフォーマンスの低下を回避することを前提として設計されているため、別のプロトコル番号を導入することはネットワークのパフォーマンス低下を許容しなければなりません。もしくは利用される可能性のあるNICすべてにSTTプロトコルオフロードを実装しなければならないことになりますが、各NICベンダがその実装を受諾するとは思えません。

また、NICへのTCP処理オフロードにも問題があります。NICのハードウェアはCPUのような割り込みの捕捉と処理を精密かつ敏速にできるわけではありません。したがって、一度OSやハイパーバイザがプロトコル処理をNICに移譲してしまうと、NICの処理が終了してレディ状態になるで割り込みを受け付けなくなる場合があります。そのため、CPU、ハイパーバイザ、ゲストOSなどの割り込みやコンテキストスイッチとNICコントロールのタイミングにズレが生

じてネットワークオペレーションがスタックする可能性があります。従来、この問題に対しては意図的にNICオフロードを無効化するか、より最適化されたドライバを導入するか、NICを入れ替えるかの選択になりますが、数千台から数万台によって構成されるクラウドIaaS基盤のすべてのNICとハイパーバイザ、ゲストOS、アプリケーション間の動作検証を実施することは気の遠くなる作業です。

つまり、STTによるネットワークの仮想化はVXLANやNVGRE以上のネットワークパフォーマンスの低下を招く可能性が高く、そのパフォーマンス低下と引き換えに得られるものは64ビットのContext IDによる無限に近いテナントセグメンテーションでしかないことになります。

まとめ



各エンドポイントモデルに共通する致命的欠陥は、VM間通信を開始する前にあります。IP網上にオーバーレイネットワークを構成するエンドポイントモデルには、VMのMACアドレスとIPアドレスをマッピングするアドレス解決手段がありません。IPv4のARPはブロードキャストによってアドレス解決を実施しますが、オーバーレイネットワークでは基本的にP2Pのトンネルで形成されるため、ブロードキャストが構成できない場合があります。そこで、VXLAN、NVGRE、STTの各ドラフトではそのアドレス解決にマルチキャストグループを利用することを示唆しています。

本来テナント間のトラフィックを安全に分離分割することを目的とした仮想ネットワークにおいて、1つのマルチキャストグループですべてのテナントのアドレス解決を賄うことは本末転倒であり、通常はテナント単位にマルチキャストグループを割り当てるでしょう。つまり、エンドポイントモデルを導入するということは、4094以上のマルチキャストグループを運用しなければならないのです。4,000以上のマルチキャストグループの運用を経験したことのあるエンジニアがこの世に存在するのでしょうか。少なくとも筆者には想像もつきません。

すでに出荷されているVMwareやMicrosoftの実装では、IaaS基盤全体を制御するオーケストレイターやフェデレータ、もしくはSDNコントローラによってIPとMACのアドレス解決を行う実装が多いようです。

また、エンドポイントモデルは物理ネットワーク上にトンネリングによってL2リンクオーバーレイを形成するネットワーク仮想化技術ですので、どのような方式であっても付加的なプロトコルヘッダとフラッグメンテーション処理により基本的には通信オーバーヘッドが発生し、ネットワークのパフォーマンスを低下させます。エンドポイントモデルの仮想化技術はネットワークのパフォーマンスを向上させることはできません。

パフォーマンスを伝送速度や伝送量だけではなく、柔軟かつ敏速なトポロジー形成を含めて論じるのであればこの限りではないでしょう。サーバ間、アプリケーション間の通信速度や伝送データ量よりも柔軟かつ敏速なテナントセグメントの形成によってシステム全体をより効果的効率的に運用することができるのであれば、十分有効な技術となりえます。

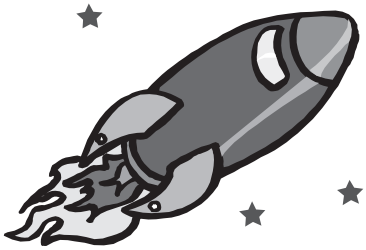
つまりエンドポイントモデルによるネットワークの仮想化はネットワーク伝送速度やスループットがそれほど重要ではなく、テナント単位のL2セグメントの追加、削除、変更を敏速に実行することが最も優先されるような環境に適しています。

ネットワークを仮想化するとすべての問題が解決されるかのような錯覚に陥りがちですが、それぞれの技術の特徴と実装の動向、そして自分たちの環境にとって最も重要なファクターは何であるかを十分に検討し精査したうえで検証、導入を判断すると良いでしょう。SD

5

Debian Developer　やまねひでき　henrich@debian.org

Debian 7.0 “Wheezy”の変更点



Debian Hot Topics

Debian 7.0リリース!

みなさんお待ちかねのDebian 7.0“Wheezy”が、5月4日に無事にリリースとなりました。インストール時のポイントはインストールガイド[注1]を、アップグレードの注意などはリリースノート[注2]を読んでいただくとして、今回はWheezyの更新内容とその背景を解説してみたいと思います。

アーキテクチャの追加

Debian名物、サポートアーキテクチャの追加が今回も行われています。対象はarmhfとs390xで、汎用機であるs390xのほうは読者のみなさんにはあまり縁がないと思いますが、armhfのほうは本誌でも連載があったOpenBlocksに採用されていたり、人気のシングルボードコンピュータであるRaspberry Piにも利用されており、armhf利用者はarmelと比較して速度アップの恩恵を受けられることになります。

パッケージ数の推移

Wheezyでのパッケージ数(概算)は、

更新：20,160
削除：4,125
追加：12,800

です。Wheezyでのパッケージ総数は37,493ですので、400個ほどを除いた37,000個のパッケージには何らかの手が加えられたということになります。この変更には、バージョンのアップデートだけではなく次に述べるようなシステム上の変更点も含まれています。

セキュリティの強化

セキュリティ強化(hardening)が一部のパッケージで実施されています。これはGCCのコンパイル時にさまざまなセキュリティフラグを有効にすることによって、攻撃の脅威を緩和するものです。方法としては、パッケージ側でdpkg-buildflagsをソースパッケージ中のdebian/rulesファイル中で明示的に利用する、あるいはdebhelperバージョン9以降を利用すること(debian/compatファイルで9を記入)で実現しています。開発段階の遅い時期に開始の号令がかけられたこともあり、すべてのパッケージに適用されるまでにはなっていませんが、必ずインストールされる「base」パッケージ、過去にセキュリティ問題が起こったパッケージやネットワークから攻撃を受ける可能性が高いデーモンを含むパッケージなど、3割程度は何らかのセキュリティフラグを有効にした状態となっています[注3]。また次の開発サイクル(コードネーム“Jessie”)では、この辺のチェックを系統立てて行って[注4]大幅にカバー率が増えるのが期待されるところです。

注1) URL http://www.debian.org/releases/wheezy/installmanual
注2) URL http://www.debian.org/releases/wheezy/releasenotes
注3) URL http://outflux.net/debian/hardening/
注4) URL http://buildd.debian.org/~brlink/

Multi-Archサポート

Wheezyで導入された「異なるアーキテクチャのライブラリを併存させてインストールできるしくみ」がMulti-Archです。今では主流の64bit環境(amd64)を使いつつ、32bitアプリケーション(i386)を利用する……などが想定される主な利用方法でしょう[注5]。このしくみを利用すれば、ライブラリはMulti-Archで導入し、カーネルにはqemuを使うことでx86環境でARMを動かすなどの芸当も手軽にできます[注6]。デフォルト状態ではそのような環境にはなっていませんので、適宜設定をして利用することになります。

具体的にSkype(i386)を利用する場合を見てみましょう。Skypeのサイトからパッケージ(Debian 7.0用)をダウンロードしてamd64アーキテクチャのマシンで**dpkg -i**でインストールしようとすると**図1**のようにエラーが出ます。

ここでまず、

```
$ sudo dpkg --add-architecture i386
```

として追加したいアークテクチャを指定します。

それから/etc/apt/sources.listの

```
deb http://ftp.jp.debian.org/debian/ ➥
wheezy main contrib non-free
```

という行を

```
deb [arch=amd64,i386] http://ftp.jp. ➥
debian.org/debian/ wheezy main contrib ➥
non-free
```

▼図1　amd64のマシンにSkype(i386)を導入する(その1)

```
$ sudo dpkg -i skype-debian_4.1.0.20-1_i386.deb
dpkg: /home/user/skype-debian_4.1.0.20-1_i386. ➥
deb の処理中にエラーが発生しました (--install):
パッケージアーキテクチャ (i386) がシステム (amd64) ➥
と一致しません
処理中にエラーが発生しました:
/home/user/skype-debian_4.1.0.20-1_i386.deb
```

のように、**[arch=<対象アーキテクチャ1>,<対象アーキテクチャ2>]**を追加するのがポイントです。追加後には**apt-get update**の実行をお忘れなく。すると、先ほどのエラーが消えて次の処理ができるようになります。ここまできたら管理者ユーザ権限で**apt-get install -f**を実行すれば依存関係のエラーが消えてSkypeが導入できます(**図2**)。

カーネル、X.org

WheezyではLinuxカーネルのパッケージ名がlinux-2.6パッケージからlinuxパッケージに名称が変更になり、合わせてバージョンも3.2となっています。なお、このバージョンのカーネルは

▼図2　amd64のマシンにSkype(i386)を導入する(その2)

```
$ sudo dpkg -i skype-debian_4.2.0.11-1_i386.deb
以前に未選択のパッケージ skype を選択しています。
(データベースを読み込んでいます ... 現在 154035 個 ➥
のファイルとディレクトリがインストールされています。)
(.../skype-debian_4.2.0.11-1_i386.deb から) ➥
skype を展開しています...
dpkg: 依存関係の問題により skype の設定ができません:
skype は以下に依存 (depends) します: libasound2 ➥
(>= 1.0.16).

(中略)

処理中にエラーが発生しました:
skype

$ sudo apt-get -f install
パッケージリストを読み込んでいます... 完了
依存関係ツリーを作成しています
状態情報を読み取っています... 完了
依存関係を解決しています ... 完了
以下のパッケージが自動でインストールされましたが、➥
もう必要とされていません:
   liblapack3gf
これを削除するには 'apt-get autoremove' を利用して ➥
ください。
以下の特別パッケージがインストールされます:
   gcc-4.7-base:i386 libasound2:i386 libaudio2: ➥
i386 libavahi-client3:i386

(中略)

この操作後に追加で 101 MB のディスク容量が消費されま ➥
す。
続行しますか [Y/n]? y
取得:1 http://ftp.jp.debian.org/debian/ wheezy/ ➥
main gcc-4.7-base i386 4.7.2-5 [143 kB]
  :
```

注5) 過去にはia32-libsパッケージを導入してこのような場合の対応を行っていました。

注6) URL http://gihyo.jp/admin/serial/01/ubuntu-recipe/0226?page=2

メンテナのBen Hutchingsさんがupstreamのメンテナも務めています。ですので、kernel.orgでリリースされているカーネルとDebianパッケージでのカーネルにはあまり乖離がありません。しかも、そのバージョンのLinuxカーネルに世界でもっとも精通した人物がパッケージのメンテナンスを行う、ということになります。もっとも複雑なソフトウェアの1つであるLinuxカーネルのパッケージを扱う人物としてこれ以上ふさわしい人材はいないと言えるでしょう。

しかし、読者の中には「今は3.9が出てるだろ、なんでもっと新しいカーネルにしないんだ？」と言われる方もいるでしょうから、背景を分析してみます。まず、Debianの安定版としてリリースされるということは最低でも3年はサポートが必要になるので、リリースされている安定版カーネルのうち、LTS（Long Term Support）カーネルである必要があります。現状、LTSカーネルのバージョンは4つ、3.4、3.2、3.0、2.6.34となっており、3.4.1が2012年6月1日リリースです。Wheezyの開発フリーズが行われたのは2012年6月30日ですから、ほぼ同時期です。ちょっとこのタイミングでは対応が難しい、と考えても無理はないでしょう。また、この時期の3.4はLTSと言いつつも、あまり「枯れて」いない状態で、一方3.2はそれなりにバグ潰しができている、と見なすこともできます。こう見ていくと、フリーズ時点で3.2以外を選択する理由はほぼなかったと言えるでしょう[注7]。また、3.2でも新しいバージョンのドライバは随時バックポートを実施して導入しているようですので、Wheezyリリース時に使えないデバイスについても、のちのポイントリリースでカーネルが更新されるときに使えるようになる可能性は高そうです。なお、もっと新しいカーネルを使いたいんだ！という人はDebian unstableに新しめのカーネルが入ってきますのでそちらからどうぞ。

X.orgは現時点での最新リリース7.7となっていますので、ビデオドライバ周りはほぼ万全という状態です。トラブルがあるとすれば、Radeonなどの一部で、KMS（Kernel Mode Setting）とファームウェアが絡んだ問題が考えられます[注8]が、その場合はfirmware-linux-nonfreeなどの該当するファームウェアを別途導入するか、起動時にパラメータで「radeon.modeset=0」などと指定をしてKMSを無効にする必要があります。

ツールチェイン

開発者にとって重要なツールチェイン周りは順当にバージョンアップをしています。コンパイラについては、リリースがごく最近のGCC4.8はさすがに入っていませんが、GCC4.7.2が採用されているので現時点では十分でしょう。一方でLLVM（clang）は3.0が採用されています。最新のclangがほしいんだ、という方は別途LLVMプロジェクトから提供されている「LLVM Debian/Ubuntu nightly packages」[注9]などを利用すると良いでしょう。

Wheezyリリースパーティ

そのほか、デスクトップ周りの解説については次回にまわすとして、イベントの話を若干しましょう。Wheezyのリリースを祝って世界各地でパーティが開かれました。日本ではちょうどゴールデンウィークに当ってしまったので、日付を1週間ほどずらして5月11日に東京・渋谷でパーティを開催しました[注10]。なお、この月はちょうど東京エリアDebian勉強会が100回を迎えたということもあり、前半はアンカンファレンス形式での勉強会（**写真1**）、後半はパーティ

注7） 筆者も「3.4とかにしないの？」と聞いたことがありますが、「3.2がもっとも自信をもってリリースできるバージョンだ」と返されました。

注8） 筆者の場合はThinkpad x121eのAMD CPUモデルだったのですが、Radeon HD 6250がこの問題に該当して一苦労させられました。

注9） URL http://llvm.org/apt/

注10） 会場をご提供いただいたVOYAGE GROUP様、ありがとうございました。

という形で開催されました。また次回Debian 8.0 “Jessie”のときも同様の催しが開けるといいですね。

大統一Debian勉強会2013

今月、6月29日(土)に東京・日本大学駿河台キャンパスにて、イベント「大統一Debian勉強会2013」が開催されます。名前は仰々しいですが、日ごろ、関東、関西、福岡と行われている勉強会が大集合して開くお祭りだととらえてください。当日は「善きDebian者は堅牢なるGnuk Tokenを使う」というGPGハードウェアトークンについてのセッションや、業務面でのDebianパッケージシステム利用についてのセッション「とあるWeb企業でのDebianシステムの使い方。」、Microsoftのクラウドプラットフォームからの視点で語るセッション「Azure de Debian」、パッケージングツールdebhelperについてのセッション「debhelperにおけるプログラミング言語対応のしくみについて」など、Debianに関する各種プレゼンテーションのほか、1日作業可能な「ハッククラボ」部屋の用意やアンカンファレンス方式の参加者による討論やワークショップ、そしてライトニングトークなどを予定しています。また前回と違い、今回は各種スポンサー様もブース参加して賑やかになることが予想されます。楽しい1日になると思いますので、読者の方々もぜひ参加を検討いただければと思います。詳しい内容とタイムテーブルなどについては、gum.debian.or.jp(図3)をご覧ください。SD

▼写真1　東京エリアDebian勉強会の様子

▼図3　大統一Debian勉強会2013の公式サイト

Red Hatと富士通のつながり

小崎 資広
KOSAKI Motohiro
富士通 Linuxソフトウェア開発統括部 所属

オンサイトエンジニアとは？

はじめまして。富士通でカーネルエンジニアとして働いている小崎と申します。

筆者は現在、ボストン郊外にあるRed Hatのウェストフォードオフィスというところで働いています。今回は番外編として、「ボストン通信」をお送りしたいと思います。

なぜ富士通のエンジニアがRed Hatのオフィスにいるのかというと話せば長くなるのですが、10年ほど前、Red Hat LinuxからRHELへと移行する黎明期に共同開発推進室をRed Hat社内に設置する合意がRed Hat・富士通の二社間で成立、メンバは交代しつつも以後ずっと共同開発パートナーシップが続いています。開発内容はカーネルダンプ機能強化やトレース機能、CPU・メモリのホットリムーブの実装、大規模マシンでの性能強化などいろいろ。富士通は東証や銀行などのミッションクリティカル系の経験が豊富なことから信頼性強化などの分野が多いですね。

なぜ西海岸ではなくボストンなの？　という質問もよく受けるのですが、Red Hatはもともと東海岸の企業なのでいまだに主要開発拠点が東にあるんですね。ただ西海岸に引っ越すという話は定期的に浮上してはいるようで、つい2年ほど前もシリコンバレーにオフィスを移すかどうか地元を巻き込んで大騒ぎし、結局ウェストフォード当局が折れて固定資産税50％OFFのスペシャル減税をオファー、残留が決まるとかいうニュースが地元新聞を賑わせたりしました。

ウェストフォードのオフィスは車で10分ほどのところにナショバというスキー場があり、またちょっと車をとばせば、日帰り圏内に本格的なスキー場がいくつもあるのでウィンタースポーツ好きな人には楽しいエリアです。LKML（Linux kernel mailing list）のメンテナは西海岸企業に雇われてる人が多いので「本物のOS開発者はTahoe（タホ、西海岸から日帰りでいけるスキー場。4.3BSD-Tahoeの由来として有名）でスキーを楽しむ」などとジョークを飛ばされますが、なに、こちらもなかなかいい所ですよ。

日々の開発形態とか

Linuxの企業なので当たり前という意見もありますが、カーネル開発はメンバがアップストリームコミュニティと大幅にオーバーラップしていることもあり雰囲気は驚くほどLKMLに似ています。みんなで思い思いにパッチをメーリングリストにポストし、ほかの開発者がレビュー、結果を返信していきます。ちゃんと確認をとったわけではありませんが、このへんのLKML風文化は初期にRHカーネルのメンテナをしていたAlan Coxの影響によるんじゃないかなぁ、と推測しています。あの人は普通じゃないから、いい意味で。

さて、恵比寿通信連載第3回にて詳しく解説されていたように、RHELには「upstream first」というポリシーがあり、原則としてアップストリームカーネルにマージされているパッチしかマージされません。ではバックポートだけだか

らアップストリームと違って簡単かと言えばそんなことはなくて、結構たいへん。ベースのカーネルバージョンが違うので素直にバックポートできないこともありますし、周辺コードが違うためにバックポートすると逆にエンバグしてしまうパッチもあります、しかたがないので別の修正方法について議論しているとアップストリームのメンテナが出てきて「なんで、おめーアップストリームで直さねーんだ？」などと瞬殺されたり。なに、よく訓練された開発者にはこれぐらいはご褒美です。

アップストリームの最新はkABI[注1]互換が崩れているのでパッチの一部分だけをバックポートしなくちゃいけないなんてときは（とくにアップストリームの変更が大規模なときは）たいへんで、「だから俺はアップストリームで議論したとき、このパッチやだって言ったんだ」などと愚痴がとびだしたり。それでもアップストリームで変更をした当人が登場したりして、なんとかなってしまうのですが。

カンファレンスでこんにちは

Red Hatは世界的な企業ですので社内メーリングリストでしょっちゅうやりとりしている相手でもオフィスが違う[注2]ので会ったことがない、なんてこともあるわけです。カンファレンスで知り合いが話し込んでいるので声を掛けてみたら同じチームの同僚とはじめて会ったので話し込んでたなどと言われたり。Linuxは結構いろいろなカンファレンスが世界中で開催されているので、みんなそういった機会を利用して飲みニケーションを図っているようです。筆者も先月サンフランシスコで行われたLinux Storage, Filesystem & MM Summit では何人かの開発者とご挨拶をさせていただきました。

注1）kABI（kernel Applicatioon Binary Interface）：カーネルインターフェースのバイナリ互換性。これが崩れるとサードパーティードライバが動かなくなる。
注2）そもそも在宅勤務の人がたくさんいます。

東海岸食事事情

ボストンといえばロブスター。旅行ガイドには絶対載ってる定番ですが、意外にも地元の人はそんなにしょっちゅう食べません。よく考えたら当然で、三重の人間が毎日伊勢エビ食べているわけではないみたいな話ですね。筆者はだいたいハンバーガー、サンドイッチ、メキシカン・ブリトー、タイヌードルあたりをローテーションしてます。中華は使ってる油がよくないのか次の日に堪えるので最近は敬遠気味。日本食は中国人や韓国人経営のところばかりなので、ほとんど行かないですね。こっちでは社食のハンバーガーでも、ちゃんと肉をその場で焼いてくれるのが新鮮でしたね。ああ、こっちでは本当に国民食なのね。と。誌面の関係であまり詳しく書けませんでしたが、本家Red Hatの開発の雰囲気を少しでも伝えることができていれば幸いです。SD

写真1は前述のMM Summitで一番議論が白熱したLinux VMの課題リスト持ち寄りと倒す順序の議論。pagevecを最初に倒そう、direct reclaimのレイテンシは重要な問題だけどデータが少ないからまずはtracepoint増やすぜとか、reclaimの延長でmmapされたページの処理でtry_to_unmap()が1ページずつIPI飛ばしてinvalidateしてるのは許せん、バッチ処理せいとか思い思いにみんなで書きなぐりました。○がついてる数字が優先度というか難易度順というか倒す順序です。

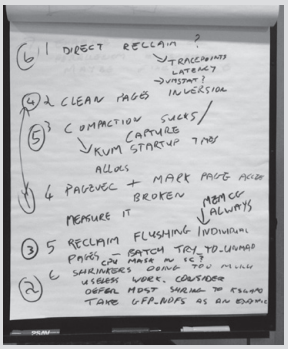


写真1　Linux VMの課題リスト

第39回

Ubuntu Monthly Report

Chromium OSをビルドする

Ubuntu Japanese Team
あわしろいくや　AWASHIRO Ikuya　ikuya@fruitsbasket.info

今回はUbuntu 12.04が推奨ビルド環境のChromium OSをビルドする方法と、簡単な使用方法を解説します。

なぜChromium OS?

みなさんはもう先月号の記事を参考に、Ubuntu 13.04をお使いのことだと思います[注1]。サポート期間が短縮されたこと以外は、13.04は非常に良いリリースだと筆者も考えており、この記事の執筆も13.04で行っています。ただし、今回使用するのはUbuntu 12.04 LTSですので、もしない場合は環境をご用意いただく必要があります。それを最初にお断りしておきます。

Chromium OSは、Googleが開発しているChrome OSのオープンソース版ですが、日本国内ではChrome OSプリインストールPCは販売されていないので、Chromium OSをビルドして手持ちのPCで使用するのが正攻法です。もちろん日本語でもバッチリ使用できるといいますか、むしろかなり注力しているようにすら思えます。Chrome OSプリインストールPCが発売されていないのが残念です。

Chromium OSをビルドする意義はいくつかあります。まずはUbuntuとは発想がかなり異なり、普通に触るだけでもおもしろいこと。Ubuntuが推奨ビルド環境であり、本稿で取り上げるのにふさわしいこと。Ubuntuを使用しているとOSのフルビルドを行う機会はほぼないので貴重な体験であること。Chromium OS自体開発が盛んに行われていること。低スペックのPCでも動作するので、初期型ネットブックなどWindows XPのサポートが切れたPCにちょうどいいことなどです。実は本連載と同時にChromium OSを活用する記事の連載もスタートしたのですが、そのころ[注2]とはビルド方法も大幅に異なっています。気になる方は『Software Design総集編【2001～2012】』でお読みいただければと思います。

ビルドする

ビルド環境ですが、OSをまるごとビルドすることになるので、それなりのスペックが望ましいです。筆者が実際に使用したのはクアッドコアCPUのAMD A8-3820です。メモリは8GBです。HDDの空き容量は40GBもあれば足りるはずです。Ubuntuは12.04のAMD64版にしてください。日本語Remixなどからインストールしたi386版は避けるのが無難です[注3]。あとはsudo権限が必要です。これはおおむね問題ないと思います。

まずは開発ツールの準備から開始します。**図1**のコマンドを実行してください。

注1） そうであってほしいものです。

注2） 回数を見て明らかなようにほぼ3年前です。

注3） i386版であってもビルドは通るかもしれませんが、遅くなるようです。それよりも、あまりテストされていないのであれば大きな落とし穴にハマる可能性もあるので、そちらのほうが重大です。

Chromium OSのビルドは、原則として専用のラッパーツールを使用しますので、そのインストールを行いました。次にsudoに細工をします。**リスト1**の内容のスクリプトを作成してください。

これをsudo_editor.shというファイル名で作成し、次のコマンドで実行します。

```
$ sudo bash  ./sudo_editor.sh
```

これでビルドの準備は整いました。次に、どのアーキテクチャでビルドするかを決定する必要があります。Chromium OSの用語では“board”と呼んでいます。boardは次の3種類です。

x86-generic ……i386用
amd64-generic ……AMD64用
arm-generic ……ARM32ビット用

この中では、x86-genericが妥当でしょう。というわけで、今回はx86-genericを使用します。次のコマンドを実行してください。

```
$ export BOARD=x86-generic
```

ソースコードを取得します。次のコマンドを実行してください。

```
$ repo init -u https://git.chromium.org/
chromiumos/manifest.git
```

最後にユーザ名とメールアドレスを質問されるので、入力してください。その後実際にダウンロードを開始します。

```
$ repo sync
```

何せOSまるごとのダウンロードですので、回線速度にもよりますが、かなり時間がかかります。本誌のほかの記事を読んでまったりお待ちください。“Your sources have been sync'd successfully.”と表示されると完了です。

いよいよビルドを実行します。次のコマンドを実行してください。

```
$ cros_sdk -- ./build_packages ?board=${BOARD}
```

完了すると“INFO : Elapsed time (build_packages): 16m59s”など、かかった時間が表示されます。この環境では17分ほどでビルドが完了したということです。

ビルドが完了したら、今度はイメージの作成です。次のコマンドを実行してください。

```
$ cros_sdk -- ./build_image ?board=${BOARD}
```

イメージの作成後にもかかった時間が表示されます。今回実行した環境では“INFO : Elapsed time (build_image): 10m3s”でした。

その下の行には、作成したイメージをUSBメモリに転送する方法と、仮想マシンのイメージに変換する方法が表示されています。**図2**は今回実行した環境での表示例です。

図1　開発ツールの準備

```
$ sudo apt-get install git-core gitk git-gui subversion curl
$ mkdir chromiumos; cd chromiumos/
$ git clone https://chromium.googlesource.com/chromium/tools/depot_tools.git
$ export PATH="$PATH":`pwd`/depot_tools
```

リスト1　sudo_editor.sh

```
#!/bin/sh
cat > ./sudo_editor <<EOF
#!/bin/sh
echo Defaults  !tty_tickets >  $1
echo Defaults timestamp_timeout=180 >>  $1
EOF
chmod +x ./sudo_editor
sudo EDITOR=./sudo_editor visudo -f /etc/sudoers.d/relax_requirements
```

USBメモリに転送する場合は、中身を削除してもかまわない4GB以上[注4]のUSBメモリを用意し、PCに挿してください。そして、**図3**のコマンドを実行します。

2番めに実行したコマンドは、ここに書かれているのを実行するのではなく、端末に表示されているものを実行してください。fromオプションのパスが異なるので、実行に失敗します。間違いなく入力し、Enterキーを押すと**図4**のような注意が表示されます。

これもUSBメモリの型番によって異なった表示になりますが、間違いがなければ"y"を入力して、Enterキーを押してください。もちろんダブルクォーテーションマークは不要です。これもしばらく時間がかかりますが、いかにもddの実行結果のようなメッセージを出力して終了します。

ついでに仮想マシン用のイメージも作成しましょう。**図5**のコマンドを実行します。

これも誌面にあるものをそのままではなく、端末に表示されているものを実行してください。説明を読むとVMware用のイメージが作成させるように見えますが、実際にはQemu/KVM用のイメージが作成されます。イメージが作成された場所が若干わかりにくいのですが、fromオプションのフォルダーに"**chromiumos_qemu_image.bin**"という名前で作成されています。筆者はVirtualBoxを使用しているため、このイメージを変換して使用したかったのですが、試してみたところマウスが動作しませんでした。ですので、Qemu/KVMを使用するのが無難です。

実行する

Qemu/KVMがインストールされていない場合、インストールしてください。

```
$ sudo apt-get install qemu-kvm
```

実行は**図6**のコマンドを実行してください。

image_to_vm.sh実行後に表示されるメッセージとほぼ同じですが、パスの指定が若干異なりますので気を付けてください。起動するとウィンドウが小さく感じますが、マウスで任意の大きさに変更できます。

注4）手元で作成したイメージは2.4GBのため、2GB以下だと明確に足りません。USB 3.0に対応した4GB USBメモリが600円とかで買える時代ですので、専用品を用意してもいいでしょう。

図2　USBに転送する方法と仮想マシンのイメージに変換する方法が表示される

```
Done. Image(s) created in /mnt/host/source/src/build/images/x86-generic/R29-4145.0.2013_05_18_1238-a1
Developer image created as chromiumos_image.bin
INFO    : Elapsed time (build_image): 10m3s
To copy the image to a USB key, use:
  ./image_to_usb.sh --from=../build/images/x86-generic/R29-4145.0.2013_05_18_1238-a1
To convert it to a VMWare image, use:
  ./image_to_vm.sh --from=../build/images/x86-generic/R29-4145.0.2013_05_18_1238-a1 ?board=x86-generic
```

図3　USBメモリに転送する

```
$ cd src/scripts/
$ ./image_to_usb.sh ?from=../build/images/x86-generic/R29-4145.0.2013_05_18_1238-a1
```

図4　注意が表示される

```
WARNING : this will erase all data on /dev/sdb: BUFFALO USB Flash Disk,
```

図5　仮想マシンのイメージを作成する

```
$ ./image_to_vm.sh --from=../build/images/x86-generic/R29-4145.0.2013_05_18_1238-a1 ?board=x86-generic
```

図6　実行する

```
$ sudo kvm -m 1024 -vga cirrus -pidfile /tmp/kvm.pid -net nic,model=virt | io -net
user,hostfwd=tcp::9222-:22 -hda ../build/images/x86-generic/R29-4145.0.2013_ | 05_18_1238-a1/
chromiumos_qemu_image.bin
```

仮想マシンの場合、遅い以外にさしたる問題はないのですが、USBメモリからの起動だと、手持ちのPCだと起動しないのが多かったです。筆者が実験した動作状況を**表1**に示します。

筆者が本命にしていたEee PC 901で動かないのは困ったもので、たとえば中古のノートPCを購入してChromium OSを使用するような場合は、事前に検証しないと動作しないということもありそうです。

設定

起動時の設定は、使用言語などとGoogleアカウントとアカウントに表示する画像だけというシンプルさです。起動直後は壁紙も何もない画面で面食らうかもしれませんが、右下の時計をクリックすると[設定]があるので、これをクリックします。一番下までスクロールし、[詳細設定を表示]をクリックしてください。まずはここで[日時]の[タイムゾーン]を[東京標準時]にします。次に[言語と入力の設定]をクリックしてください。[Mozc(for Japanese Keyboard)]にチェックを入れます。ATOKキーバインドに変更したい場合は、さらに[設定]をクリックし、[キー設定の選択]を[ATOK]にしてください。Mozcを起動する方法は、設定画面に書かれているとおり Alt + Shift キーです。

壁紙とChromiumのテーマも変更できますが、壁紙はどうも用意されていなさそうですので、各自で用意する必要がありそうです[注5]。

設定とはちょっと違いますが、Ctrl + Alt + F2 キーで仮想コンソールが表示できます。戻る場合は Ctrl + Alt + F1 キーです。ユーザ名などは仮想コンソールに書いてあります。

2回目以降のビルド

2回目以降はもっと簡単にビルドできます。詳細な解説は省略しますが、おおむね**図7**のとおりに実行すればいいでしょう。

あとはimage_to_usb.shやimage_to_vm.shを実行してください。

注意

Chromium OSは活発に開発されているため、ビルドのタイミングによっては失敗することや、起動しなくなるといった問題が発生する可能性は十分にあります。そればかりか、ビルドの方法が変わることもありえます。もっとも、これ以上シンプルにする方法は思いつきませんが……。そのような場合には、英語ですがQuick Start Guide[注6]を確認してください。ビルドできない場合は、日を置いて試してみればビルドができるようになるでしょう。SD

注5) Google+にアップロードしている写真などを選択できてもよさそうですが……。

注6) http://dev.chromium.org/chromium-os/quick-start-guide

図7　2回目以降のビルド

```
$ cd chromiumos/depot_tools
$ git pull
$ cd ../
$ export PATH="$PATH":`pwd`/depot_tools
$ repo sync
$ export BOARD=x86-generic
$ cros_sdk -- ./build_packages ?board=${BOARD}
$ cros_sdk -- ./build_image ?board=${BOARD}
$ cd src/scripts/
```

表1　動作状況

発売年	メーカー	型番	動作可否	備考
2008	ASUS	Eee PC 901-16G	×	画面が表示されず
2009	ACER	Aspire Timeline AS1410	○	
2012	ASUS	U24A-PX3210	×	起動はするがタッチパッドが動作せず
2012	ASUS	X202E-CT987G	×	起動しなかった

Linux カーネル観光ガイド

第16回 Linux 3.10で予定される新機能～pvpanic～

Text：青田 直大　AOTA Naohiro

3.9カーネルがリリースされ、3.10カーネルの開発が始まっています。3.0系になったのがついこの前のようなのにもう2桁まで来てしまいました。Linuxカーネルの開発が活発に進んでいることがうかがえますね。今月はLinux 3.10に入る機能の中からpvpanicとAndroidのSynchronization frameworkを中心に紹介していきたいと思います。

pvpanic

pvpanicは仮想マシンのゲスト側のカーネルがパニックしたときに、ホストにそのことを通知するための機能です。この機能を試すには、

- Linuxカーネル 3.10 以降
- qemu 1.5.0以降
- seabios のgit版

が必要になります。Linuxカーネルの設定部分は次の場所にあります。

```
Device Drivers --->
  [*] X86 Platform Specific Device Drivers  --->
    <M>   pvpanic device support
```

上記3つをインストールして（カーネルモジュールはちゃんとゲスト側のファイルシステムにインストールしましょう）、ゲストのLinuxを起動します。まずQEmuによってISAのパニック通知インターフェースが、そしてseabiosによってACPIのパニック通知インターフェースがQEMU0001というIDで追加されます。ACPIの場合にはたとえば**図1**のようにしてインターフェースが追加されていることを確かめることができます。続いてモジュールpvpanicをカーネルに読み込ませれば準備は完了です。sysrq-triggerを使ってカーネルをわざとパニックさせてみましょう（**図2**）。

カーネルパニックしたらQEmuのモニターに切り替えます。グラフィカルなウィンドウを使っている場合は Ctrl + Alt + 2 で、qemuを-nographic引数で起動していてる場合は Ctrl + a c でQEmuモニターに切り替わります。モニターで仮想マシンの状態を見てみるとゲストがパニックしたことを検出しているのがわかります（**図3**）。

このとき同時にQMP（QEMU Monitor Protocol）というプロトコルでパニックしたことを知らせるイベントが送信されています。qemuを"-qmptcp:127.0.0.1:4444,server"という引数を付けて起動し、qmp-shell[注1]をこのアドレスを指定して接続します。これはQEmuのモニターと同じように仮想マシンの状態を見たり、動作中にディスクを追加したり、仮想マシンの電源を

注1）ここから入手できます。http://git.qemu.org/?p=qemu.git;a=tree;f=QMP;hb=HEAD

▼図1　ACPIの確認とモジュールの読み込み

```
# strings /sys/firmware/acpi/tables/SSDT|grep QEMU
QEMU0001
# modprobe pvpanic
# lsmod
Module                  Size  Used by
pvpanic                 2017  0
```

▼図2　カーネルをパニックさせる

```
# echo c > /proc/sysrq-trigger
[  256.428775] SysRq : Trigger a crash
[  256.429307] BUG: unable to handle kernel NULL pointer dereference at           (null)
[  256.429460] IP: [<ffffffff814581b6>] sysrq_handle_crash+0x16/0x20
[  256.429460] PGD 1c122067 PUD 1cbd5067 PMD 0
[  256.429460] Oops: 0002 [#1] SMP
[  256.429460] Modules linked in: pvpanic
[  256.429460] CPU: 0 PID: 776 Comm: bash Not tainted 3.10.0-rc2+ #250
[  256.429460] Hardware name: Bochs Bochs, BIOS Bochs 01/01/2011
[  256.429460] task: ffff88001c090000 ti: ffff88001c054000 task.ti: ffff88001c054000
[  256.429460] RIP: 0010:[<ffffffff814581b6>]  [<ffffffff814581b6>] sysrq_handle_
crash+0x16/0x20
[  256.429460] RSP: 0018:ffff88001c055e68  EFLAGS: 00010092
[  256.429460] RAX: 000000000000000f RBX: 0000000000000063 RCX: 00000000000005da
[  256.429460] RDX: 0000000000001099 RSI: 0000000000000046 RDI: 0000000000000063
[  256.429460] RBP: ffff88001c055e68 R08: 0000000000000003 R09: ffffffff81ea8048
[  256.429460] R10: 00000000000001ab R11: 00000000000001aa R12: ffffffff81c93c40
[  256.429460] R13: 0000000000000286 R14: 0000000000000001 R15: 0000000000000000
[  256.429460] FS:  00007f9df51ab700(0000) GS:ffff88001fc00000(0000) knlGS:0000000000000000
[  256.429460] CS:  0010 DS: 0000 ES: 0000 CR0: 0000000080050033
[  256.429460] CR2: 0000000000000000 CR3: 000000001cf1f000 CR4: 00000000000006f0
[  256.429460] DR0: 0000000000000000 DR1: 0000000000000000 DR2: 0000000000000000
[  256.429460] DR3: 0000000000000000 DR6: 00000000ffff0ff0 DR7: 0000000000000400
[  256.429460] Stack:
[  256.429460]  ffff88001c055ea8 ffffffff814589a1 ffff88001c055ee8 0000000000000002
[  256.429460]  fffffffffffffffb 00007f9df51b2000 ffff88001c055f50 0000000000000001
[  256.429460]  ffff88001c055ec8 ffffffff81458a43 ffff88001c055ee8 ffff88001ddb4300
[  256.429460] Call Trace:
[  256.429460]  [<ffffffff814589a1>] __handle_sysrq+0x121/0x190
[  256.429460]  [<ffffffff81458a43>] write_sysrq_trigger+0x33/0x40
[  256.429460]  [<ffffffff812102c3>] proc_reg_write+0x43/0x70
[  256.429460]  [<ffffffff811b207e>] vfs_write+0xce/0x1e0
[  256.429460]  [<ffffffff813b105a>] ? trace_hardirqs_off_thunk+0x3a/0x6c
[  256.429460]  [<ffffffff811b2562>] SyS_write+0x52/0xa0
[  256.429460]  [<ffffffff81721006>] system_call_fastpath+0x1a/0x1f
[  256.429460] Code: 45 01 f4 45 39 65 34 75 e5 4c 89 ef e8 24 f8 ff ff eb db 66 90 0f 1f 44
00 00 55 c7 05 14 fa a4 00 01 00 00 00 48 89 e5 0f ae f8 <c6> 04 25 00 00 00 00 01 5d c3 0f
1f 44 00 00 55 48 89 e5 53 48
[  256.429460] RIP  [<ffffffff814581b6>] sysrq_handle_crash+0x16/0x20
[  256.429460]  RSP <ffff88001c055e68>
[  256.429460] CR2: 0000000000000000
[  256.429460] ---[ end trace 1afcd15abc8fc3e7 ]---
[  256.429460] Kernel panic - not syncing: Fatal exception
```

▼図3　パニック検出の確認

```
QEMU 1.5.50 monitor - type 'help' for more information
(qemu) info status
VM status: paused (guest-panicked)
```

切ったりするためのものです。pvpanicモジュールを読み込んでいて、カーネルがパニックするとqmp-shellでイベントが送られてきているのを見ることができます。

図4のようにパニックしたことを示すイベントである“GUEST_PANICKED”と、仮想マシンが停止したことを示すイベントである“STOP”とが送られてきています。ここで仮想マシンが停止しているのもpvpanicによるものです。

pvpanicのしくみ

pvpacniのしくみを、まずは初期化の部分から見ていきましょう。

QEmuにhas_pvpanicという変数が追加され、これがtrueの場合にpvpanic_init()が呼び出されます。has_pvpanicは1.5の仮装マシンではデフォルトでtrueになっていて、それ以外ではfalseとなります。pvpanic_init()でisa_create_simple()によって、pvpanicのISAデバイスを登録します[注2]。

仮想マシンの起動時に、pvpanicのISAデバイスの初期化関数であるpvpanic_isa_initfn()が呼び出されます[注3]。この関数ではISAのI/Oポートを準備し、さらにこのI/Oポートの位置を“/machine/fw_cfg”というオブジェクトに、“etc/pvpanic-port”というファイル名で追加します[注4]。

SeabiosはACPIのSSDT(Secondary System Description Table)というデバイス情報を記載したテーブルを構築するときに、先ほどの“etc/pvpanic-port”というファイルがあるかどうかをチェックします[注5]。もしこのファイルがあればそこに書かれているpvpanicのI/Oポート番号を、なければ0をSSDTテーブルの“QEMU0001”というIDを持つ部分に書きこみます。これで仮想マシン側からACPIを使ってpvpanicのI/Oポートがどこにあるのかを、ACPIのSSDTの“QEMU0001”から取得できるようになったわけです。

ゲスト側でpvpanicのカーネルモジュールが読み込まれると、まずACPIのSSDTを参照して、“QEMU0001”からpvpanicのI/Oポートを取得します[注6]。さらに、カーネルパニック時に呼び出されるコールバックとしてpvpanic_panic_notify()を登録します。

カーネルパニックが起きるとpvpanic_send_event()からoutb()を使って、I/Oポートに“1”が送信されます[注7]。ここからQEmuのhandle_event()に伝えられます(**リスト1**)。handle_event()で、QEmuからQMPにてGUEST_PANICKEDイベントが送出され、仮想マシンが“RUN_STATE_GUEST_PANICKED”状態になって停止されます(**図5**)。

注2) http://git.qemu.org/?p=qemu.git;a=commitdiff;h=3ab135f3462af4c523a4b5969f9d6c67b2ac427a

注3) http://git.qemu.org/?p=qemu.git;a=commitdiff;h=eec3d2adc98dd9ef7352823ce6597f88a51cf7cb

注4) http://git.qemu.org/?p=qemu.git;a=commitdiff;h=10a584b2875a391d1036adac18955a892e56f5e3

注5) http://code.coreboot.org/p/seabios/source/tree/master/src/acpi.c

注6) http://git.kernel.org/cgit/linux/kernel/git/torvalds/linux.git/tree/drivers/platform/x86/pvpanic.c#n93

注7) http://git.kernel.org/cgit/linux/kernel/git/torvalds/linux.git/tree/drivers/platform/x86/pvpanic.c#n59

▼図4　qmp-shellへのパニックの通知

```
# qemu -qmp tcp:127.0.0.1:4444,server ...
# qmp-shell  127.0.0.1:4444
Welcome to the QMP low-level shell!
Connected to QEMU 1.5.50

(QEMU)
{u'timestamp': {u'seconds': 1369325618, u'microseconds': 719474}, u'data': {u'action':
u'pause'}, u'event': u'GUEST_PANICKED'}
{u'timestamp': {u'seconds': 1369325618, u'microseconds': 953920}, u'event': u'STOP'}
(QEMU) quit
{u'return': {}}
```

このようにQEmu、Seabios、Linuxカーネルがうまく組み合わされてカーネルパニックを通知しています。

Android sync driver

以前からAndroidのカーネルに入っている機能がLinux本体へとマージされていましたが、3.10でも新しくAndroid出自の機能Synchronization frameworkと、それを使ったSoftware synchronization objectsという実装がマージされています。

Synchronization frameworkは、その名前から想像できるとおり、各ドライバ間で同期をとるためのものです。しかし、ただ同期をとるだけであれば、すでにカーネルの中にあるものでも十分なはずです。Synchronization frameworkは、いったいどのような特徴があるのでしょうか。

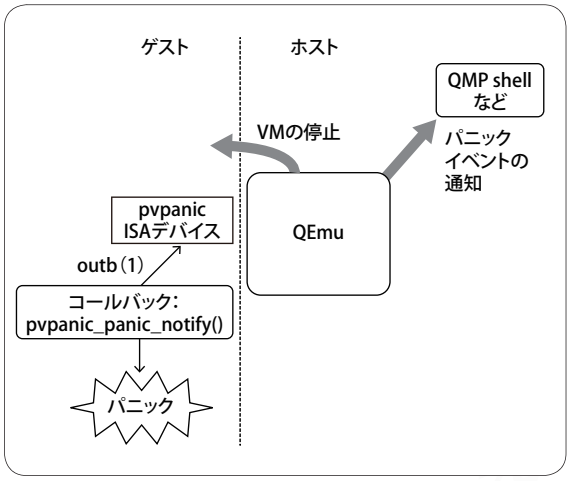
Synchronization frameworkには、

- timeline
- sync point
- fence

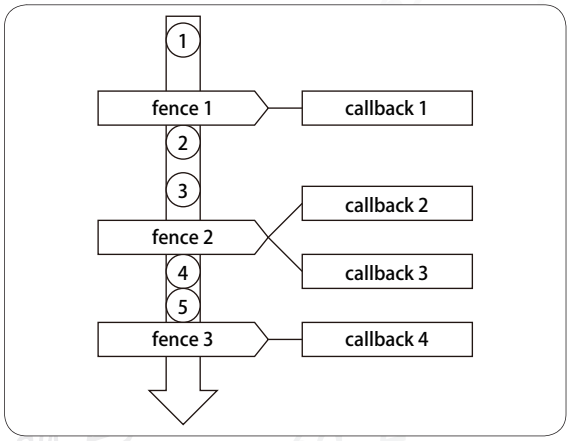
という3つの基本的な構造があります。timelineはその名のとおりのときの流れとなります。そのtimelineの中にいくつかsync pointが設置されています。各sync pointは対応するfenceを持っており、逆にあるfenceは1つ以上のsync pointに対応しています。シンプルな例であれば図6のようなイメージとなります。

sync pointには「シグナル」をすでに発行したか、まだ発行していないかの2つの状態があります。また、fenceにはファイルデスクリプタが

▼図5　pvpanicの動作



▼図6　timelineの流れ



▼リスト1　QEmuのhandle_event

```
static void handle_event(int event)
{
    static bool logged;

    if (event & ~PVPANIC_PANICKED && !logged) {
        qemu_log_mask(LOG_GUEST_ERROR, "pvpanic: unknown event %#x. n", event);
        logged = true;
    }

    if (event & PVPANIC_PANICKED) {
        panicked_mon_event("pause");
        vm_stop(RUN_STATE_GUEST_PANICKED);
        return;
    }
}
```

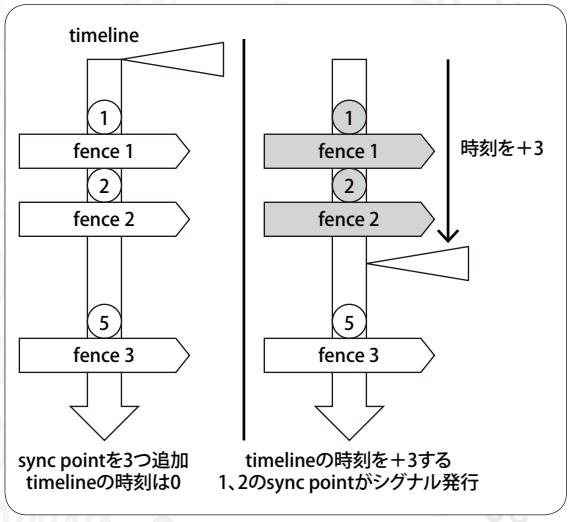
対応づけられており、このファイルに対してpollしたり、(タイムアウトを設定して)waitしたり、あるいはコールバック関数を設定したりできます。これらのpoll、wait、callbackは、そのfenceのすべてのsync pointがシグナルを発行したときにpoll、waitが終了し、callbackが実行されます。**図6**でいえば、sync point1から4がシグナルを発行すると、callback1から3が実行されますがcallback4は実行されません。

ここで問題になるのが「シグナルが発行される」とはなんなのか、というところです。実はこれはSynchronization frameworkを使う実装依存になっており、あるsync pointがシグナルを発行したかどうかを判定する関数をtimelineの作成時に指定できるようになっています。たとえば、Software synchronization objectsの場合は時刻のようなものでシグナルが発行されたかどうかをチェックします。

/dev/sw_syncというデバイスファイルをopen()する[注8]と、Software synchronizationのtimelineが作られ、その「時刻」が0に設定されます。このファイルには2種類のioctl()を行うことができます。1つはある「時刻」を指定してsync pointと、そのsync pointを持つfenceを作ること。もう1つは「時刻」を指定した数だけ進めることです。timelineの「時刻」が、sync pointの「時刻」以後になるとsync pointがシグナルを発行したとされます。つまり、まず「時刻」1、2、5にsync pointを作ると**図7の左**のようになります。次に、「時刻」を3進めてみると**図7の右**のようにtimelineの「時刻」が移動し、その結果「時刻」1、2のsync pointがシグナルを発行します。

注8) カーネル設定SW_SYNC_USERが必要です。

▼図7 /dev/sw_syncによる2つのioctl()



mempressureの導入

ここからは、3.10で入るちょっとした機能をいくつか紹介していきましょう。まずは以前の記事でも紹介したmempressure notificationです[注9]。もう一度簡単にこの機能を紹介しましょう。この機能が入るとmemory cgroupに新しく“memory.pressure_level”というファイルが追加されます。通知を受けるためのeventfdと、このmemory.pressure_levelとを開いて、それらのデスクリプタと受け取りたい「プレッシャーレベル」とをcgroup.event_controlに書くと、レベルに応じた通知をeventfdから受け取ることができます。

「レベル」には“low”、“medium”、“critical”の3種類があります。lowは新しい空きメモリを確保するべく使われていないメモリの回収を始めたときに通知されます。mediumはメモリを回収しようとしたものの、見にいったメモリの60%以上が回収できなかった(使用中であった)ときに通知されます。これはシステムがswapを開始したり、使用中のファイルのキャッシュを落とし始めているぐらいの時点です。まだ余裕はあるのでシステムの状態を調べて、アプリケーション側で使用中のキャッシュを解放するかどうか検討するだけの時間はあります。criticalは、見にいったメモリの95%以上が回収できなかったときに通知されます。ここまでくるとswapの

注9) http://git.kernel.org/cgit/linux/kernel/git/torvalds/linux.git/commit/?id=70ddf637eebe47e61fb2be08a59315581b6d2f38

I/Oでシステムがほぼ動作しない状態となっていて、OOM Killerでプロセスが強制終了されるのも間近です。ここまでくると今すぐに何らかの対策をしなければなりません。

スクリプト実行機能がオプショナルに

次に、UNIX系でスクリプトを書いているとお馴染みの、あのファイルの冒頭の「#!」からスクリプトインタプリタを指定する機能をカーネル組み込みではなくモジュールにしたり、あるいは完全に取り外したりできるようになりました[注10]。つまり、スクリプトをまったく使わない(使えない)システム(あるいはモジュールにしておけば、ルートファイルシステムをmountするまではスクリプトを使わないシステム)を作ることができるわけですが、ちょっと一般には使いどころに悩んでしまいますね。

randconfigの改善

ほかにもLinux kernelの設定項目を編集するプログラムのkconfigにもちょっと変わった変更が入っています。Linux kernelの設定パラメータは膨大な数になっています。もしかすると機能Aをオンにして機能Bをオフにしたときにだけビルドが失敗したり、起動に失敗することがあるかもしれません。しかし、その設定パラメータのすべての組み合わせをテストすることは現実的ではありません。その代わりにランダムに設定項目をオン／オフできれば、「人手ではなかなか起こり得ないけれど、バグが起きる組み合わせ」を見つけだす手出てにできます。すでに“make randconfig”によって「ランダムに設定項目をオン／オフする」ことができるようになっています。3.10では、このrandconfigの機能を2ヵ所改善しています。

1つめは“KCONFIG_PROBABILITY”という環境変数でランダムにオン／オフする確率を変更できるようになったことです[注11]。今までは単純なオン(y)／オフ(n)であれば50%／50%で、組み込み(y)／モジュール(m)／なし(n)のものでは33%／33%／34%というように、すべての選択子が均等に選ばれていました。これをたとえば“KCONFIG_PROBABILITY”に“N”と0～100の数値を設定すると、オンになるのがN%で、オフになるのが100－N%というように確率が変わります。ほかにも“N:M”／“N:M:L”といった設定が可能でそれぞれ**表1**のような確率になります(y:nは単純なオン／オフの設定項目で、y:m:nは組み込み／モジュール／なしの設定項目)。

2つめのrandconfigに関する変更は、この乱数のシード値を環境変数“KCONFIG_RANDSEED”で設定できるようになったことです[注12]。これによって同じシード値を与えれば常に同じ「ランダムな」設定を得ることができます。

まとめ

今回はLinux 3.10に入る機能であるpvpanicを中心に、mempressure、スクリプトの実行機能、randconfigの改善について簡単に解説しました。pvpanicがあれば、仮想マシンがパニックしていることを知ることができる(しかも、仮想マシンを停止してくれる)ので、git bisectによるカーネルのテストの自動化にも、仮想マシンで動いているサーバの状態監視にも使えるのではないでしょうか。SD

注10) http://git.kernel.org/cgit/linux/kernel/git/torvalds/linux.git/commit/?id=2535e0d723e4d7723b030f39fb350e436bdb983f

▼表1 “KCONFIG_PROBABILITY”による確率変更

	y:n	y:m:n
N	N:100-N	N/2:N/2:100-N
N:M	N+M:100-(N+M)	N:M:100-(N+M)
N:M:L	N:100-N	M:L:100-(M+L)

注11) http://git.kernel.org/cgit/linux/kernel/git/torvalds/linux.git/commit/?id=e43956e607692f9b1c710311e4a6591ffba1edf0

注12) http://git.kernel.org/cgit/linux/kernel/git/torvalds/linux.git/commit/?id=0d8024c6ebadb68f1154377c2e1996b4e649e4c8

IPv6化の道も一歩から

第7回 ネットワーク構築時の注意点と落とし穴(サーバ設定編)

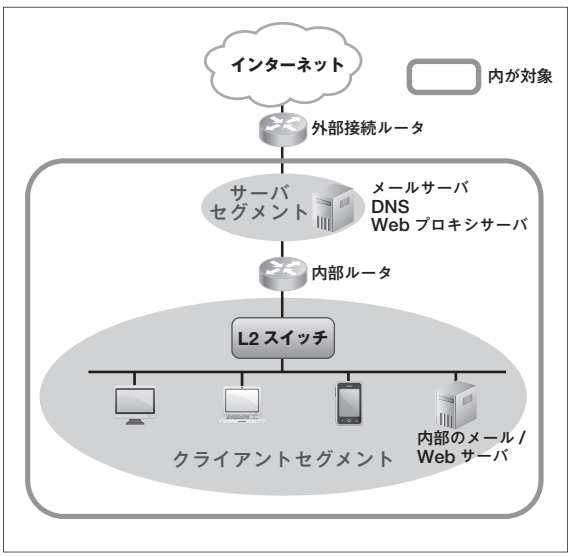
IPv6普及・高度化推進協議会　IPv4/IPv6共存WG　アプリケーションのIPv6対応検討SWG
廣海 緑里 HIROMI Ruri　渡辺 露文 WATANABE Tsuyufumi　新 善文 ATARASHI Yoshifumi　藤崎 智宏 FUJISAKI Tomohiro

前回のおさらい

前回(2013年5月号)は、試験ネットワークを構築し、端末を接続してIPv6通信ができることを確認するところまでを扱いました(**図1**、**表1**)。今回はデュアルスタックで動作するサーバを設定し、IPv6でのサーバへの接続を確認していきましょう。

その前に、前回のやり残し作業として、DHCPv6でDNS情報を端末に伝える部分について解説します。

▼図1　試験環境構成図



DNSサーバ情報をDHCPで取得させる

ルータのOフラグを有効にする

前回ルータの設定に関しては、インターフェースにIPv6アドレスを設定し、クライアントセグメント側にルータ広告(RA)でプレフィックスを流すというところまでを行いました。この状態ではクライアントにはDNSサーバ情報(リゾルバのアドレス)が設定されません。DHCPv6によりクライアントにDNSサーバ情報を取得させるには、ルータからOフラグの値を1にセットしたRAを流し、クライアントがほかの情報をDHCPv6で取得するように伝達します。Oフラグは、アドレス以外の情報を設定するフラグ(Other Configuration Flag)で、DNSサーバ情報以外にもNTPやSIPサーバなどの伝達に利

▼表1　試験環境のグローバルユニキャストアドレス(GUA)

	利用プレフィックス	設定方法
試験ネット	(サーバセグメント)2001:db8:0:1::/64 (クライアントセグメント)2001:db8:0:2::/64	―
ルータ	(サーバセグメント)2001:db8:0:1::1/64 (クライアントセグメント)2001:db8:0:2::1/64	手動設定
サーバ	2001:db8:0:1::10/64	手動設定
クライアント	2001:db8:0:2:fffe:[EUI64アドレス]/64	自動設定

用できます。O=1のRAを受け取ったクライアント端末ではDHCPv6クライアントを起動して、IPv6アドレス以外の情報をDHCPv6で取得するようになります。

ルータ設定内容

ルータによって細かいコマンドは違ってきますが、おおむね**図2**の設定を投入する形になります。実際に利用される場合には、コマンドリファレンスなどでご利用のルータOSで利用できることを確認のうえ、正確な文法で設定してください。

DHCPv6サーバの設定

各社のルータにはDHCPサーバ機能を実装しているものも多く、ルータでDHCPサーバを兼ねるようにもできます。しかし、ここではルータではなくLinuxマシンをDHCPサーバとして構築する場合を紹介します[注2]。

今回利用するDHCPv6プログラムは、CentOS 6.4に採用されているISCのDHCPサーバソフトウェアです。`yum install dhcp`を実行することでインストールできます。設定は/etc/dhcp/dhcpd6.confファイルに記述します。この試験環境では、クライアントセグメント（2001:db8:0:2::/64）を対象に、2001:db8:0:1::10をDNSサーバとして、example.jpを検索ドメインとして配布するように設定します（**リスト1**）。設定を記述したあと、`service dhcpd6 start`を実行してDHCPv6プログラムを起動します。

クライアント端末

クライアント側はネットワーク設定で「IPv6アドレスを自動設定する」を有効化しておくだけで、自動的に設定されます。万が一、クライアントにDNS情報が設定されていない場合には、アドレスの再取得や再起動、DHCPv6プログラムの設定の再確認を行ってください。Windows 7の場合、`ipconfig /all`と入力することで、インターフェースのアドレス設定情報とDNSサーバ情報を一度に表示させて確認できます。

注2） 話が前後してしまいますが、サーバへのIPv6アドレス設定については後半をご覧ください。

▼リスト1　/etc/dhcp/dhcpd6.confの設定例

```
option dhcp6.name-servers 2001:db8:0:1::10;
option dhcp6.domain-search "example.jp";
dhcpv6-lease-file-name "/var/lib/dhcpd/dhcpd6.leases";

subnet6 2001:db8:0:2::/64 {
}
```

▼図2　クライアントセグメントのインターフェース[注1]

```
①O=1にする
ipv6 nd other-config-flag（AlaxaIA、Cisco IOS）
rtadvd interface lan1 other-flag on（IIJ SEIL/X1）
protocols router-advertisement interface fe-0/0/2.0 other-stateful-configuration（Juniper SRXシリーズ）
ipv6 nd ra other-config-flag（NEC IXシリーズ）
ipv6 lan2 rtadv send 2 o_flag = on（YAMAHA RTXシリーズ、あらかじめipv6 prefix 2 2001:db8:0:2::/64を設定）
```

注1） 今回参考にしたルータの情報は次のとおり。**AlaxaIA AXシリーズ**（http://www.alaxala.com/jp/techinfo/archive/manual/AX3650S/HTML/11_10/CFGUIDE3/INDEX.HTM） **Cisco IOS**（http://www.cisco.com/cisco/web/portal/support/docs_listing.html?cid=282770988&locale=ja_JP&itag=prod_conf_guides_list） **IIJ SEIL**（http://www.seil.jp/support/tech/doc/command/） **Juniper Junos**（http://www.juniper.net/techpubs/en_US/junos12.3/information-products/pathway-pages/product/12.3/index.html） **YAMAHA RTXシリーズ**（http://jp.yamaha.com/products/network/routers/rtx1200/）

サーバのデュアルスタック化

DHCPv6でDNSサーバ情報を伝達するところまでできたので、次はサーバ設定に入っていきます。サーバに関する情報を**表2**にまとめました。サーバのOSはCentOS 6.4にしています。Linux系であれば大きな違いはないと思いますが、お使いのOSのコマンドと照らし合わせながら進めてください。

OSの設定

サーバのIPv6アドレスはRAを使わず、ifconfigコマンドやipコマンドを使ってEthernetのインターフェースに固定アドレスを手動設定します(今回の例ではipコマンドを使います)。IPv4は192.0.2.10/24、IPv6は2001:db8:0:1::10/64とします。

▼表2　サーバ情報

項目	値
ホスト名(FQDN)	v6test01.example.jp
ドメイン名	example.jp
IPv6アドレス	2001:db8:0:1::10/64
IPv4アドレス	192.0.2.10/24
OS	CentOS 6.4

```
# ip addr add 192.0.2.10/24 dev eth0
# ip addr add 2001:db8:0:1::10/64 dev eth0
```

設定を投入したら、同じくipコマンドで確認してみます(**図3**)。ARPキャッシュ、近隣キャッシュの情報もipコマンドで取得できます(**図4**)。

次に静的経路の設定をします。IPv4のdefaultは192.0.2.1。IPv6のdefaultは2001:db8:0:1::1/64です。

```
# ip route add default via 192.0.2.1
dev eth0
# ip route add default via 2001:db8:0:
1::1 dev eth0
```

経路の確認もしておきます。

```
$ ip -4 route show
$ ip -6 route show
```

ipコマンドによる設定は、OSを再起動すると消えてしまいます。再起動後も永続的に使用するためにはCentOSの作法に従い/etc/sysconfig/networkと/etc/sysconfig/network-scripts/ifcfg-<インターフェース名>ファイルに**リスト2**、**3**の設定を行います。これでOSの設定は完了です。

▼図3　ipコマンドでIPアドレスを確認

```
# ip addr show dev eth0
3: eth0: <BROADCAST,MULTICAST,UP> mtu 1500 qdisc pfifo_fast qlen 1000
    link/ether 00:xx:xx:xx:xx:xx brd ff:ff:ff:ff:ff:ff
    inet 192.0.2.10/24 brd 192.0.2.255 scope global eth0
    inet6 fe80::202:xxx:xxx:xxx:xxx/64 scope link
       valid_lft forever preferred_lft forever
    inet6 2001:db8:0:1::10/64 scope global
       valid_lft forever preferred_lft forever
```

▼図4　ipコマンドでL2の情報を確認

```
 IPv4
$ ip -4 neigh show

192.0.2.1 dev eth0 lladdr 00:00:XX:00:XX:XX REACHABLE

 IPv6
$ ip -6 neigh show

fe80::XXX:XXXX:XXXX:XXXX dev eth0 lladdr 00:00:XX:XX:XX:XX router REACHABLE
```

DNSサーバの設定

OSのIPアドレス設定が終わったら、次はDNS周りの設定です。まずは、この自ホストの名前解決の設定から始めましょう。

自ホストの名前解決のための設定

Linuxの場合、名前解決に利用する機構の順序を/etc/nsswitch.confで制御します。CentOS 6.4の初期設定では、hostsファイルとDNSの両方を参照するように設定されており、hostsファイルを優先するように設定されています（**リスト4**）。

続いて、名前解決に利用する/etc/hostsファイルの記述方法です。IPv4アドレスの名前解決と同様に「IPアドレス ホスト名」の順番でエントリを記載します（**リスト5**）。

DNSによる名前解決の設定は、/etc/resolv.confにて行います。IPv4でのリゾルバ指定と同様に、「nameserver IPアドレス」とエントリを記載します（**リスト6**）。nameserverレコードが複数存在する場合には、記載された順に従って問い合わせが行われます。そのため、IPv6/IPv4のデュアルスタック環境において、IPv6/IPv4それぞれのnameserverレコードを記載し、その順番を入れ替えることでDNS問い合わせクエリーに用いるプロトコルの優先順位を制御できます。

クライアントのための設定

自ホストの名前解決の設定が終わったら、次はDNSサーバとしての設定です。DNSは当初、UDPによる送信データの最大長が512オクテットと決められていました。その後、DNSが扱うデータの種類が増え、これに伴いデータ長が長くなってきたため、UDPによる送信データについてEDNS0という拡張を行いました（RFC2671）。IPv6の名前解決を行う場合やDNSSECを利用する場合には、送信データが512オクテットを超えることがあるため、EDNS0に対応したDNSサーバが必要となります。

CentOS 6.4に採用されているDNSサーバbind 9.8.2はEDNS0に対応しています。bindはnamed.confファイルでDNSサーバの設定を行い、別ファイルでリソースレコードを管理します。named.confファイルの設定では、まずIPv6

▼リスト2　設定を永続化する（/etc/sysconfig/network）

```
NETWORKING=yes
HOSTNAME=v6test01.example.jp
IPV6_AUTOCONF=no注3
```

▼リスト3　設定を永続化する（/etc/sysconfig/network-scripts/ifcfg-eth0）

```
DEVICE=eth0
TYPE=Ethernet
ONBOOT=yes
BOOTPROTO=none

IPADDR=192.0.2.10
NETMASK=255.255.255.0

IPV6INIT=yes
IPV6ADDR=2001:db8:0:1::10/64
IPV6_DEFAULTGW=2001:db8:0:1::1

PEERDNS=no
```

▼リスト4　/etc/nsswitch.confのホスト名解決順序に関する初期設定

```
#hosts: db files nisplus nis dns
hosts: files dns
```

▼リスト6　/etc/resolv.confのnameserver設定例

```
nameserver ::1
```

▼リスト5　/etc/hosts設定例

```
127.0.0.1 localhost localhost.localdomain localhost4 localhost4.localdomain4
::1 localhost localhost.localdomain localhost6 localhost6.localdomain6
2001:db8:0:1::10 v6test01
```

注3）IPV6_AUTOCONF=noはアドレス自動設定を行わないための設定です。

でlistenする設定を記載します(**リスト7**)。

あとはforwarderやallow-*など各設定項目でIPv4アドレスの代わりにIPv6アドレスを記載するだけです(**リスト8**)。

なお、named-checkconfコマンドを用いると、簡単に設定の誤りを確認できます。正引き(ホスト名からIPアドレスを得る)名前解決のレコード登録をzoneファイルで行います。まずはnamed.confファイルで参照するzoneファイルを指定します(**リスト9**)。この設定はIPv4の場合と変わりありません。

続いてzoneファイルの設定です。IPv4アドレスの登録にはAレコードを用いますが、IPv6アドレスの登録にはAAAAレコードを用います(**リスト10**)。

▼リスト7　named.confのlisten-on-v6の設定例

```
listen-on-v6 port 53 { any; };
```

▼リスト8　named.confのallow-queryとallow-transferの設定例

```
allow-query { any; };
allow-transfer { 2001:db8:0:1::11; };
```

▼リスト9　named.confのzoneの設定例

```
zone "example.jp" {
  type master;
  file "example.jp.zone";
};
```

▼リスト10　zoneファイルのexample.jp.zoneレコード登録例

```
v6test01 IN A 192.0.2.10
v6test01 IN AAAA 2001:db8:0:1::10
```

▼リスト11　2001:db8:0:1::/64の逆引き設定例

named.confへの登録例

```
zone "1.0.0.0.0.0.0.0.8.b.d.0.1.0.0.2.ip6.arpa" {
  type master;
  file "test-env.rev";
};
```

test-env.revへの登録例

```
0.1.0.0.0.0.0.0.0.0.0.0.0.0.0.0 IN PTR v6test01.example.jp.
```

なお、named-checkzoneコマンドを用いると、簡単にエントリの不備をチェックできます。

逆引き(IPアドレスからホスト名を得る)名前解決のレコード登録を逆引きゾーンファイルで行います。まずnamed.confファイルで参照する逆引きゾーンファイルを指定します。続いて、指定した逆引きゾーンファイルにリソースレコードのエントリを記載します。

この逆引き登録の流れは複雑ですので2001:db8:0:1::/64を例に**図5**に図示します。まず、IPv6アドレスを省略表記を用いずに表します。続いて4文字ずつ**:**で区切っていたものを、1文字ずつ**.**で区切るように変更します。そして、文字の順序を反転し、末尾に**.ip6.arpa**を結合します。

2001:db8:0:1::/64の逆引き設定例は、**リスト11**となります。

このように逆引きの登録は手間がかかります。IPv6では、これまでより多くの端末がネットワークに接続されることが見込まれることもあり、IPv6の逆引き設定および運用を、IPv4で行っていた運用と同等に行うか、意見が分かれるところです。

Webサーバの設定

続いてWebサーバの設定です。WebサーバはCentOSに採用されているApache HTTP Server 2.2.15で説明を行います。Apache HTTP Serverの設定はおもにhttpd.confファイルにて行います(**リスト12**)。IPアドレスが影響する個所は、Listenするポート、バーチャルホスト、アクセス制御あたりかと思います。IPv4アドレスを記入していた個所にIPv6アドレスを記入することで設定できます。ただし、Listenポートおよびバーチャルホスト設定でIPv6アドレスを指定する場合、IPv6アドレスを角括弧**[ ]**でくくる必要があります。

SMTPサーバの設定

SMTPサーバはCentOSで採用されている

Postfixを用いて説明を行います。Postfixの設定は、おもに/etc/postfix/main.cfにて行います（**リスト13**）。まず、IPv6をサポートするかどうかの設定をinet_protocolsパラメータにて設定します。CentOS 6.4に同梱されているPostfix 2.6.6では、デフォルト値が"all"となっており、デフォルトでIPv6がサポートされます。

IPアドレスを指定することがあるパラメータがいくつかありますが、IPv6アドレスを指定する場合、IPv6アドレスを角括弧[ ]でくくる必要があります。ここでは、同一ネットワークとみなすアドレスを設定するmynetworksパラメータを例示します。

プロキシサーバの設定

プロキシサーバはSquidを用いて説明を行います。Squidの設定はおもに/etc/squid/squid.confにて行います。CentOS 6.4に同梱されているSquid 3.1.10ではIPv6をサポートしています。設定上でIPアドレスを扱う個所は、aclに関する設定です（**リスト14**）。

▼リスト12　Apache HTTP Serverの設定（httpd.confなど）

Listenポート設定例
```
Listen [2001:db8:0:1::10]:80
```
バーチャルホスト設定例
```
<VirtualHost [2001:db8:0:1::10]:80>
```
アクセス制御設定例
```
Order Deny, Allow
Deny from all
Allow from 2001:db8:0:1::/64
```

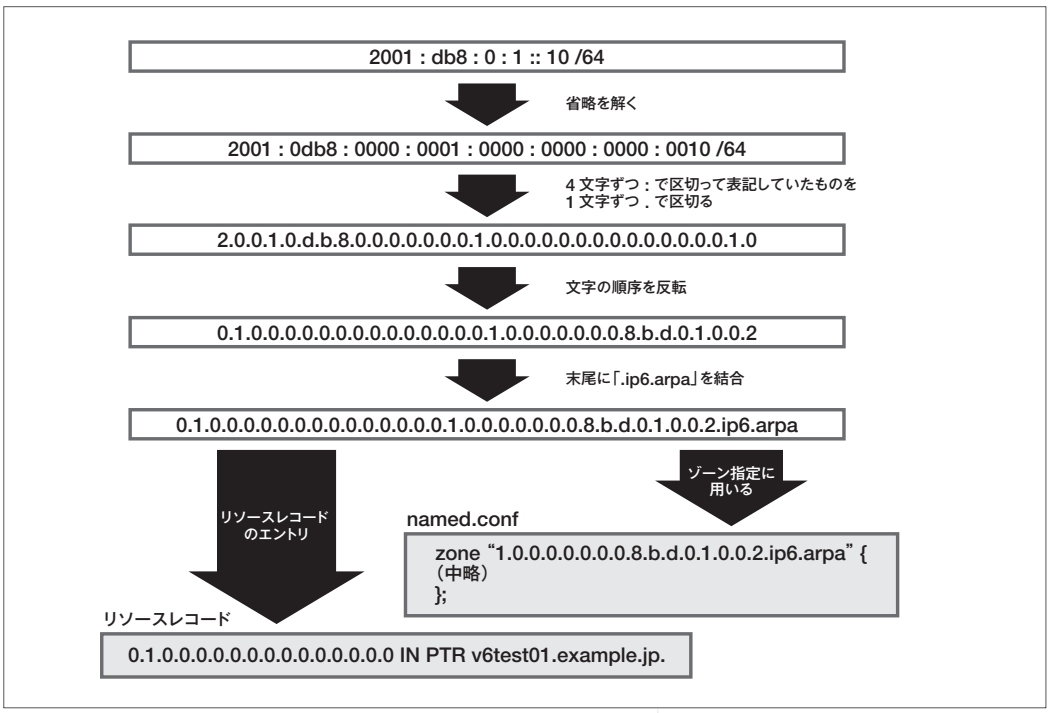
▼リスト13　Postfixの/etc/postfix/main.cfの設定

inet_protocols指定例
```
# Enable IPv4, and IPv6 if supported
inet_protocols = all
```
mynetworks指定例
```
mynetworks = 127.0.0.0/8, [::1]/128,
[2001:db8:0:1::]/64, [2001:db8:0:2::]/64
```

▼リスト14　squid.confのacl設定例

```
acl localnet src 2001:db8:0:2::/64
```

▼図5　bindのIPv6アドレス逆引き登録



動作確認

最後にサーバのIPv6での動作確認です。まずはDNSからです。DNSの動作確認では、IPv6で接続できること、正引きの問合せ結果としてAAAAレコードが返ってくること、IPv6アドレスの逆引き結果が返ってくることの3点を確認します。ここではdigコマンドを使ってIPv6で接続し、正引き(**図6**)と逆引き(**図7**)の問合せを行って結果を確認します。

続いてWebサーバの確認です。Webブラウザでhttp://[2001:db8:0:1::10]/を閲覧して確認してください。プロキシサーバの確認は、実際にプロキシとして2001:db8:0:1::10を指定して行います。上記Webサーバの確認で閲覧したページが正常に閲覧できることを確認してください。

最後にSMTPサーバの確認です。telnetで2001:db8:0:1::10の25番ポートに接続し、「220 ホスト名 ESTMP Postfix」の応答があれば、接続が確立しています(**図8**)。

終わりに

今回は、DHCPv6によるDNS情報配布とサーバ構築について解説しました。難しい設定はなく、簡単に構築できることがおわかりいただけたかと思います。次回はアプリケーション開発のIPv6対応について取り上げます。お楽しみに。SD

▼図6　digコマンドによるDNS正引き確認

```
$ dig @2001:db8:0:1::10 v6test01.example.jp -t aaaa
（中略）
;; QUESTION SECTION:

;v6test01.example.jp. IN AAAA

;; ANSWER SECTION:
v6test01.example.jp. 14400 IN AAAA 2001:db8:0:1::10
（中略）
;; Query time: 2 msec

;; SERVER: 2001:db8:0:1::10#53(2001:db8:0:1::10)
（以下省略）
```

▼図8　telnetによるSMTPサーバへの接続確認

```
$ telnet 2001:db8:0:1::10 25
Trying 2001:db8:0:1::10...
Connected to 2001:db8:0:1::10.
Escape character is '^]'.
220 v6test01.example.jp ESMTP Postfix
```

▼図7　digコマンドによるIPv6アドレスDNS逆引き確認

```
$ dig @2001:db8:0:1::10 -x 2001:db8:0:1::10 -t ptr
（中略）
;; QUESTION SECTION:

;0.1.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.8.b.d.0.1.0.0.2.ip6.arpa. IN PTR

;; ANSWER SECTION:
0.1.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.8.b.d.0.1.0.0.2.ip6.arpa. 14400IN PTR
v6test01.example.jp.
（中略）
;; Query time: 3 msec

;; SERVER: 2001:db8:0:1::10#53(2001:db8:0:1::10)
（以下省略）
```

バックナンバーのお知らせ　BACKNUMBER

バックナンバー好評発売中！常備取り扱い店もしくはWebより購入いただけます

本誌バックナンバー（紙版）はお近くの書店に注文されるか、下記のバックナンバー常備取り扱い店にてお求めいただけます。また、「Fujisan.co.jp」（http://www.fujisan.co.jp/sd）や、e-hon（http://www.e-hon.ne.jp）にて、Webから注文し、ご自宅やお近くの書店へお届けするサービスもございます。

2013年6月号

第1特集
わかった人だけメキメキ上達
ちゃんとオブジェクト指向できていますか?

第2特集
研修じゃ教えてもらえない!?
あなたの知らないUNIXコマンドの使い方

一般記事
・リアルタイム分散処理「Storm」ほか

1,280円

2013年5月号

第1特集
IT業界ビギナーのための お勧め書籍55+α
新しい季節に君へ

第2特集
覚えておきたい、ちゃんと使いたい!
正規表現をマスターしていますか?

一般記事
・バーチャルネットワークコントローラ2.0開発の実際

1,280円

2013年4月号

第1特集
僕（私）の言語の学び方
裏口からのプログラミング入門

第2特集
オブジェクト指向再入門
ソフトウェア開発に効くSmall Objectをご存じですか?

特別企画
・スクウェア・エニックス＋Skeed
「ゲーム開発の舞台裏」

1,280円

2013年3月号

第1特集
オープン環境でスキルアップ!
もっとクラウドを活用してみませんか?

第2特集
光、ギガビット、高速ネットワークを体験!
実践! ワイヤリングの教科書

一般記事
・「SSDストレージ」爆発的普及の理由

1,280円

2013年2月号

第1特集
UNIXコマンド、fork、pipeを復習し、高度なスクリプティングへ
シェルスクリプティング道場

第2特集
忙しいITエンジニアのための
超効率的勉強法

一般記事
・Samba 4.0.0ファーストインプレッション

1,280円

2013年1月号

第1特集
いざというときに備える
システムバックアップ

第2特集
IT業界のキーパーソンに聞く
2013年に来そうな「技術」・「ビジョン」はこれだ!

特別付録
法輪寺電電宮情報安全護符シール

1,380円

Software Design　バックナンバー常備取り扱い店							
北海道	札幌市中央区	MARUZEN&ジュンク堂書店　札幌店	011-223-1911	神奈川県	川崎市高津区	文教堂書店　溝の口本店	044-812-0063
	札幌市中央区	紀伊國屋書店　札幌本店	011-231-2131	静岡県	静岡市葵区	戸田書店　静岡本店	054-205-6111
東京都	豊島区	ジュンク堂書店　池袋本店	03-5956-6111	愛知県	名古屋市中区	三洋堂書店　上前津店	052-251-8334
	新宿区	紀伊國屋書店　新宿本店	03-3354-0131	大阪府	大阪市北区	ジュンク堂書店　大阪本店	06-4799-1090
	渋谷区	紀伊國屋書店　新宿南店	03-5361-3315	兵庫県	神戸市中央区	ジュンク堂書店　三宮店	078-392-1001
	千代田区	書泉ブックタワー	03-5296-0051	広島県	広島市南区	ジュンク堂書店　広島駅前店	082-568-3000
	千代田区	丸善　丸の内本店	03-5288-8881		広島市中区	丸善　広島店	082-504-6210
	中央区	八重洲ブックセンター本店	03-3281-1811	福岡県	福岡市中央区	ジュンク堂書店　福岡店	092-738-3322
	渋谷区	MARUZEN &ジュンク堂書店　渋谷店	03-5456-2111				

※店舗によってバックナンバー取り扱い期間などが異なります。在庫などは各書店にご確認ください。

デジタル版のお知らせ　DIGITAL

デジタル版Software Design 好評発売中！紙版で在庫切れのバックナンバーも購入できます

本誌デジタル版は「Fujisan.co.jp」（http://www.fujisan.co.jp/sd）と、「雑誌オンライン.com」（http://www.zasshi-online.com/）で購入できます。最新号、バックナンバーとも1冊のみの購入はもちろん、定期購読も対応。1年間定期購読なら5%割引になります。デジタル版はPCのほかにiPad／iPhoneにも対応し、購入するとどちらでも追加料金を払うことなく、読むことができます。

July 2013

NO.21

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

jusが歩んできた30年

jusは1983年6月に設立され、今年の5月で30周年を迎えました。そこで今回は普段お届けしているイベントレポートではなく、jusが歩んできた30年の道程を紹介します。

設立の経緯と初期の活動

UNIXが日本に伝来したのは1970年代の後半です。当時はまだインターネットもなく、UNIX関連の情報交換をするにはユーザ同士で集まって話をするのがもっとも効果的な手段でした。そこで、1980年代に入って各社からUNIXマシンが登場したのを契機に、ベンダに依存しないUNIXのユーザグループとしてjusが設立されました。初期の顕著な活動としては、UNIX関連の研究成果を発表するUNIXシンポジウムや、UNIXマシンおよびソフトウェアの展示会であるUNIX Fairが挙げられます。

UNIX／インターネット発展への貢献

1980年代後半から1990年代に入ると、UNIXはビジネスにも積極的に利用されるようになり、情報交換の場であるjusも重要な存在となりました。とくにUNIX Fairは、メーカー各社がUNIXマシンを製造しさらにOSも開発していた当時において、それらの製品が一堂に会する貴重なイベントでした。最盛期には会員数約2500人、UNIX Fairの来場者数は約4万人に達しました。

また、この時代は日本のインターネットの黎明期でもあります。当初はUNIXマシン同士をUUCPで接続したJUNETが形成され、それが後にインターネットへと発展していったのですが、当時のUNIX Fairは各社UNIXマシンの相互接続性検証の場としても利用されました。jusは日本のインターネットの発展にも大きく貢献したのです。

jusとインターネット技術との関わりはその後も続いており、1996年に始まったインターネットコンファレンス（IC）、その翌年スタートのInternet Week（IW）、2008年からはIPv4アドレス枯渇対応タスクフォースに参加しています。

■Windowsとの共存

1990年代後半に入ると、UNIXマシンよりも廉価なWindows系OSを搭載したマシンが普及し始めます。UNIXとは異なるアーキテクチャを持つWindowsとの共存は、技術的にもビジネス的にも大きな課題となりました。この時期のjusはWindows系ユーザグループとの共催によるワークショップや、UNIX Fairの後継イベントとなるNetwork Users'などを開催しました。

■個人がUNIXを使い学ぶ時代へ

Windowsの普及と前後して、UNIXの世界にもPCで動作するLinuxやFreeBSDなどが登場し、個人でもUNIXマシンが手に入る時代になりました。このような流れに対応してjusは1993年に個人環境研究会を設立し、ワークショップなどを通じてPC-UNIX関連の情報交換の場を提供しました。

また、1994年には月例の勉強会を始めました。セミナー参加費1万円以上が当たり前だった時代に、個人が仕事帰りに自費で勉強できるように参加費を1,000円に設定しました。20年経った今ではコミュニ

ティ主催の勉強会が毎日のように行われていますが、jusの勉強会はその先駆けになったものと自負しています。

活動はUNIX以外の分野にも

■オープンソースへの貢献

1990年代末にはオープンソースの概念が登場します。これまでの活動でBSDやLinuxなどのグループとつながりを持っていたjusは、オープンソース関連の企業、コミュニティ、ユーザが集う場としてオープンソースまつりを1999年と2001年に開催しました。2002年以降は関西オープンフォーラム（KOF）を関西地区のコミュニティとともに開催しています。

■軽量言語コミュニティへの貢献

2000年代に入ると、プログラミング言語コミュニティとの交流が活発化します。以前からYARPCなどのイベントを通してPerlやRubyのコミュニティとは付き合いがあったのですが、ここにPHPとPythonが加わる形で2003年からLightweight Languageイベントが始まりました。これはほかの言語コミュニティも巻き込んだ大きな流れとなり、真夏の一大イベントとして定着しています。

■地域コミュニティへの貢献

jusでは設立当初からjus関西UNIX研究会を定期開催し、これは関西地区におけるUNIXコミュニティの形成に寄与しました。1990年代に入ると東海地区でも研究会が行われるようになり、どちらも2000年代途中まで続きましたが、2007年からはこれらを引き継ぐ形で全国で研究会を開催し、各地のコミュニティとの交流を深めています。

◆ ◆ ◆

ここ数年の活動はおおむね一定しており、年中行事としてLLイベント、KOF、IC、IW、定例会として勉強会と研究会を行っています。このほかにガジェット系コミュニティとの合同イベントとしてGadget1を開催しています。これらの活動からわかるように、近年はjus単独で運営する行事は勉強会ぐらいで、ほかのコミュニティとのコラボレーションが不可欠になっています。

まとめ

こうして振り返ってみると、当初はUNIXを対象とするグループとしてスタートしたのが、時代のニーズに応じて新しい概念に柔軟に対応してきたことがわかります。これはそれぞれの技術領域に対応できるグループがほかになかったという当時の状況もありますが、jusが団体名や技術領域に固執せず、必要とされることに自ら進んで取り組んできた結果であり、これが30年もの長きに渡って団体を継続することができた秘訣ではないかと考えています。

最近はコミュニティが自主的に行事を開催できるようになったので、コミュニティそのものを支援する役割は減りましたが、これからも多くの人がUNIX的なものにお世話になるはずで、そこにjusが貢献できる場面はきっとあるでしょう。その想いを胸に活動していきます。これからもjusをよろしくお願いします。

■30周年記念行事のご案内

jusは毎年7月に定期総会と併設行事を行っていますが、今年の併設行事は30周年記念イベントとして開催する予定です。

- **タイトル：日本UNIXユーザ会のこれまでとこれから ～jus設立30周年記念～**
- **日時：2013年7月20日（土）14：30～17：30（終了後に懇親会を予定）**
- **場所：ハロー貸会議室 四谷**
- **URL：http://japanunixsociety.doorkeeper.jp/events/4111**

詳しくはjusのWebサイト[注1]やイベント情報ブログ[注2]、および公式Twitterアカウント（@jus_pr）で発表します。ぜひご参加ください。SD

注1）URL http://www.jus.or.jp//
注2）URL http://jus-event.blogspot.jp/

Hack For Japan
エンジニアだからこそできる復興への一歩

Hack For Japan

第19回 Hack For Japanスタッフ座談会［後編］

IT で復興支援を考えるHack For Japanとして活動を続けていくにあたり、「これまでの振り返りと今後に向けた決意」をテーマにスタッフで座談会を開きました。今回はその後編です。

福島の問題

佐々木：福島の問題は、あらためて一度集まって話しをして、いま誰が困っているかを見極める必要があります。自立して普通に生活している人は増えていて、とくに若い人はもうアパートを借りて住んでいたりします。残っているのはお年寄りの方ばかりです。会津にある私の会社の裏が避難住宅なので見るんですけれども、明かりがポツンとしかなくてほとんど人が来なくなってきています。

IT で何か、という観点では、福島で活動をされている方のサイトをいくつか見てみたのですが、リアルタイムの情報が載っていないのです。「こういう支援をしています」という情報はあっても、いつどこで誰がというものはなく、情報の更新が止まっている感じがします。

問題が複雑だということもあるのかもしれませんが、いま問題意識を持って動いている人がいるのかどうかよくわからない状況です。ですので、そういう人を福島で探してみるのもありかもしれませんね。エフスタさん[注1]とかはどうなんでしょう。

鎌田：エフスタさんはITのコミュニティを育てて、人が離れていっている現状を何とかしようという活動をされていますね。

佐々木：そろそろ何かやらないとマズいという雰囲気にはなっています。福島は広すぎるのでもともと分かれていたのですが、エリアとして分断され始めているので、IT を使ってできることは多いのかなと思います。ただ繰り返しになりますが、一度いろいろな人を集めて議論しないと、誰が何で困っているかわからないという状況ですね。

高橋：震災直後の4月末に会津でみんなで話し合うイベントを行いましたよね。ああいった場をもう一度持つのが良いでしょうか。

佐々木：そう思います。会津は風評被害はあっても直接の被害はないエリアですが、中通りと沿岸部にはここら辺でテコ入れするのが良いかなと思います。福島は最近イベントが少ないので、何か目的意識を持ってみんなで集まってやると良いのではないかと思います。

沿岸部の人以外は、確かにみんな元どおりの生活にはなってきているんです。震災前と後で生活が変わったかというと、やはり沿岸部の人以外はあまり変わっていないんです。沿岸部の人は生活が極端に変わって全然違う人生になっているのですが、本来サポートすべき人たちがあまりアクティブに動いていないという傾向があると思っています。そこを今年は1つのテーマにして、できることを積極的にやっていきたい、という感じがしています。いろいろな人がちょくちょく来てくれるのですが、事例としては南相馬が一番目的意識が高いかなと思うので、そこが参考になるかなと思います。

スキルマッチングとOpenData

高橋：メディアへのアピールなどは広くドライブかけてやっていくべきでしょうか。マッチングのしくみを知らせるという観点では、広く伝えるべきだとは思うのですが。

及川：やはり「何でもやれますよ」という感じで言うのは少し無理があると思っていて、どこまでできる

注1）福島で活動しているITエンジニアのためのスキルアップ応援コミュニティ。

参加者紹介（Hack For Japanスタッフ）

及川 卓也
Hack For Japanの立ち上げメンバーの1人。普段はGoogleでChromeを担当しているほか、知り合いのスタートアップやNPOに助言を与えたりしている。今回はGoogle Hangoutで参加。

小泉 勝志郎
サンキュロットインフォ代表。スマートフォンアプリ開発関連の事業を行う。震災で失業者が多い南相馬でアプリ開発者育成をした。震災復興では「うらと海の子再生プロジェクト」にてIT関連を担当。

岩切 晃子
（株）翔泳社勤務。毎年開催されているデブサミを運営。岩手県釜石市出身でHack For Japanの活動で岩手とのパイプ役として奮闘している。

鎌田 篤慎
普段はヤフー（株）が公開するAPIなどの利用促進、デベロッパリレーションなどを業務としている。Hack For Japanでは復旧復興支援データベースAPIへの改善要望をまとめ、国に提言した。

佐々木 陽
会津若松の（株）GClueの代表取締役。Android、iOSアプリケーション開発が主な事業。未来の主戦力となるエンジニアを育てるため、大学生などに教える活動を10年間行っている。

佐伯 幸治
Hack For Japanではコピーライティングをおもに担当。普段はフリーランスとしてWebや紙媒体の編集制作・コピーライティングに携わっている。

関 治之
Georepublic Japan社CEO。Geo Developerとして位置情報系のサービスを数多く立ち上げてきた。Hack For Japanでは、復興マッピングやオープンデータハッカソンなどを実施している。

石野 正剛
震災直後に福島第一原発から放出される放射性物質と風向きを地図上に可視化するスマホアプリ「風@福島原発」を開発した。富士通（株）でソフトウェアのUXデザインを担当している。

高橋 憲一
普段は（株）スマートエデュケーションのエンジニアとしてiOSやAndroidの子供向け知育アプリ開発を行っている。最近は東北TECH道場の講師として宮城県の石巻を頻繁に訪れている。

かわからない状況で大風呂敷を広げるのは怖いというのもあります。一方、状況が変わっている中で、いろいろな地域でちょっと街に出れば日常があるところなのに、いつまでもボランティアに頼んでいいのかと被災者側が感じてしまう部分もあるのではないかと思います。

1つは“何かそのしくみを使ってできるもの”を地道に作っていくというのが良いのではないかと思います。たとえば、石野さんが進めている風@福島原発[注2]のサーバ側の開発の部分を手伝ってもらうとか、そういうのがいいんじゃないかと思います。

スタッフ側がやろうと思ってもいまはちょっとできなかったり、もしくは、この辺りを変えたいと思っていままで少数でやっていたものに対して、ほかの人に加わってもらうとか。たとえば風@福島原発の花粉版、PM2.5版を作るようなことはどうでしょう。

石野：実はすでに3部作ということでアプリは出してあって、花粉とPM2.5についても風@福島原発と同じやり方でできています。その話に関連して（ここに集まった皆さんと）共有したいことがあります。今年は花粉がもの凄かったですよね。そしてPM2.5も注目されてましたよね。花粉のほうは去年作ったのですがトラブルはありませんでした。ところが今年はサーバがダウンしました。原因はデータソースである環境省のサイトにあって、ブラウザで見るとタイムアウトしてしまうという状況だったのです。2日後にはサーバが増強されて改善はしたのですが、そのときに某ISPの方と「官公庁のデータの出し方やサーバのトラブル対応の仕方というのは何も変わっていない。こういうのは担当者を集めて話し合わないと駄目ですね」というような話をしました。

関：OpenDataの観点でのいい事例ですね。

アイデアはまだまだあるはず

及川：テーマを決めて意見を募ってみたほうが良いのではないかなと思います。先日、Hack For Japanにかかわっている経緯で東日本大震災アーカイブ[注3]で、関さんが技術面、私が利活用ワーキンググループのメンバーになっており、どう利用するか、どんなことができたらいいかというような議論をしました。

東日本大震災データアーカイブのようなものは、自治体の人たちが震災の記録を取っていったりアーカイブを用意するにはそれなりのコストがかかって大変なことだというのがあります。ただ、OpenDataというイニシアチブもあって進めたいというときに、一生懸命アーカイブとして作ったものがどう使われて、それが地元の人にとってどう役立つのかを考えてみる。

たとえば地元の経済や観光にどう利用できるのか。ざっと考えただけでも、以前セカイカメラで

注2）石野正剛氏が作成した、福島原発を中心とした風向きを調べられるスマートフォンアプリ。

注3）http://shinsai.mapping.jp/

やっていたような、ARで震災前の写真を背景に重ねられるような例も含めて、ジオタグだとかビジュアルなイメージだとか、いくらでも考えつくアイデアがあります。ちょっと提案するだけで、中越の地震や阪神大震災を体験をされた方々が、そのときにこういうものがあったら良かったし、いまはないので紙芝居的にやっているけれども、実はあれはITでできるはずですよね、というようなアイデアがまたどんどん出てくるんですよね。

いまの話は1つの例だけれども、そんなふうに地元の方に一緒に入ってもらって、テーマを持ってやっていくとプロジェクトのアイデアは出てくるのではないかなと思います。

マッチングについては、1つはスタッフ自身がやろうと思っていること、やりかけていることで、誰か一緒にやらないかと呼びかけをしていくということ。それと、テーマを絞った形でプロジェクトの募集をしてみる。これらを通じて実際のマッチングにつなげられるような、我々の考えているメンター制度も含め、いったん上手くいくようなプロジェクト例を用意していってみると良いのではないかなと思います。

関：そういう意味ではspending.jp[注4]のクローンを作りませんか？というのはやってみたいですね。あれはマニュアルもできているので。

岩切：いっぱい作りましょうみたいな。

佐伯：量産！

関：いま8都市あるんです。

及川：あれいいですよね。GitHubにあるやつ[注5]を見てたんですけども、簡単にできそうで。あれを一気に作りましょうと呼びかけるのもいいですよね。

関：あれはGitHubページを使っているのでサーバもいらないんです。

石野：日本全国版は作れないのでしょうか。

関：国政版も作りたいのですが、データがかなり複雑になるので行政側の詳しい人を巻き込む必要があります。それはそれでやりたいと思っています。

データ公開をうながすしくみ作り

及川：先日、岩手の大学の先生方と話したときに、ないものはないで、わからないときはわからないで仕方ないとして、でもそれを変えるために、1つは前にやったような提言書を作るのもありますが、その場でデモ用のデータを作って、もしデータがあればこういうことができるというのをどんどん見せていく必要があるというような話をしました。

岩切：何せ見たことがないですからね。

関：結局、鶏と卵なんですよね。どちらか無理矢理にでも作る必要がある。

高橋：石巻でやっている東北TECH道場[注6]で、自分がいる場所の津波の高さをAR的に見せてくれるアプリを作ってくれた参加者がいました。その津波の高さのデータは市役所から紙のデータとしては手に入れられたのですが、デジタルデータにするためにそれを手で入力する必要がありました。これはデジタルデータがもしあれば、という良い事例になると思います。

岩切：自治体の現場からは、「データを公開してほしいという要望はたくさんあるのだが、なかなか手も回らないし、どう公開すれば良いかわからない」という事情が聞こえてきます。だから「公開はこうやるんですよ」というのが提案できると良いのかなと思います。

鎌田：このあいだのHackathonの東京会場でも、千代田区の方が、「できてもExcelで公開するくらいです」と言っていました。

関：そういうガイドラインづくりは進みつつあるので、徐々に整備されていくと思います。ちょっと時間はかかると思いますが。

岩切：大学の先生が来てくれて公開用のデータも作ってくれていると、研究用のテーマにしてくれるような状況だとデータの作り方も学べるからありがたいというような話もありました。

関：OpenData周りではそういう話は結構あって、

注4）自分の収入を入力すると、納めた税金がどこで使われているかを見ることができるしくみ。
注5）**https://github.com/orezeni/orezeni.github.com**
注6）**https://sites.google.com/site/tohokudojo/** Hack For Japanからも講師を派遣しています。

Code for America[注7]の日本版をやりたいと考えています。どういうしくみかというと、エンジニアが1年間かけるのですが、まずは3ヵ月自治体に張り付いて働いてもらい問題を把握します。その後本部に戻って、フェローと一緒にアプリケーションをどう作るか、ワークフローをどう変えるかということを残りの9ヵ月かけて作るというものになっています。無償ではなく、それなりの対価をもらってやるしくみで、いくつか受け入れ側の話をしているところで、被災地のどこかは入れたいと考えています。1年間、凄く優秀なエンジニアが中に入って動いたら、相当良いものができるはずだと思います。

まとめ

高橋：今後、Hack For Japanをどうしていきたいかという話をすると……

関：Hack For Japanのwikiを作って、みんなにもっとHackathonをやってもらうようにしようというのはやりたいです。もっとコミュニティの力を生かせる組織にするべきだと思ってるんですよ。

スタッフが企画してというのではなく、やりたいことをHack For Japanの名前を使ってできる、ブランドやみんなのリソース、知恵を使ってやりたいことができるというようなものに、被災地のコミュニティが生まれるような手助けをしたいと思っています。あとは先ほど言ったCode for Japan。

高橋：去年のIT Bootcampでやったような教育活動も進めていきたいですね。

岩切：Hack For Japanとしての教育プログラムを作って、宮城、岩手、福島で、年1回とかでやるのは良いのかも。

高橋：何ヵ所かでやるとなると講師の数も必要になるので、スキルマッチングでアプリ開発を教えられる人を募集するのも良いかもしれませんね。

小泉：ほかにはStartup Weekend[注8]と組むとおもしろいと思います。Hackathonの場合プロジェクトがその場限りになりがちですが、Startup Weekendのプロジェクトは続いているものが多いです。

及川：大賛成です。Startup Weekendとジョイントしてやるのも良いし、そこのエッセンスを持って来てやるのも良いと思います。

小泉：通常のHackathonだとエンジニア以外の人は途中から暇になりがちなのですが、Startup Weekendの場合はそういう人たちも暇にならずに頑張れる印象があるんです。

及川：そのとおりで、いくつかやることがあって、たとえばカスタマバリデーションという街に出てアンケートをとって使えるかどうか考えるという作業があります。プログラム以外の仕事が山ほどあるので、エンジニア以外の人もやることが十分あるんです。

関：ぜひやりましょう。ほかの団体との連携は今年はもっと進めたいと思っています。

及川：Hack For JapanがITでの復興支援のすべてを担う必要はなくて、ほかの人たちがどこをどう埋めようとしているか、どういう協力体制を取っていけるかを話すのは良いと思います。

関：ITで支援活動をしてた人たちを集めて、一度話をする場を持つのは凄くいいなと思いますね。

及川：何も生まれなくても、知り合っておくだけでも良いと思っています。「この人に連絡できればこの辺は何とかなる」と顔が浮かぶような状況になっていて、何かあったときに早い段階でコンタクトが取れれば良いのかなと。

岩切：Hack For Iwateに県庁の人が入ってくれたのは、また何かあったときに助けてほしいと言うことができるからとのことなので、最低限の目標としてはそういうことで良いのかなと思います。

及川：防災、減災、教育といくつか柱があって、これらでちゃんと成果を出していきたいですね。そして「Hack For Japanというのは志を一にする緩やかな集まり」ということで、いままではスタッフ側から提案をしたりイベントを開催するという感じでしたが、それをやりつつ、賛同してくださる方が自発的にHack For Japanの名前の下にいろんな活動を開始してもらえるようにするのが良いと思います。今年の前半のうちには形にしていきたいですね。SD

注7）http://codeforamerica.org/
注8）http://startupweekend.jp/

温故知新
IT むかしばなし

第24回

6809/OS-9 /6829 MMU

たけおかしょうぞう　TAKEOKA Shouzou　take@takeoka.net

はじめに

6809（MC6809）は、モトローラ（現フリースケール）で6800（MC6800）の後継として1979年に発売された、豊富なアドレッシングモードを備えた8bit CPU[注1]です。C言語の「*p++」や「*--p」にあたる命令を持っていました。

6809は、アドレッシングモードが強力であったため、アセンブラで位置独立（PIC：Position Independent Code）なコードを書くことは非常に容易でした。プログラムコード付近に置いた定数は、PC（プログラムカウンタ）相対アドレッシングでアクセスし、関数のローカル変数は、スタック上に取り、スタックポインタ相対でアクセスします。OSが用意してくれた（グローバル変数用の）データ領域の先頭は、プロセス生成時にUレジスタ（ユーザスタックポインタ）に入ってきますから、Uレジスタ相対でアクセスします（スタック領域も、プロセス生成時にOSが用意してくれます）。このようなコーディングにすると、自動的に、独立なプロセス同士は再入可能（リエントラント）になり、複数のプロセスで、メモリ上の1つのバイナリコードを共有できるようになります。

6809は8bit機として珍しく、上記のようなレジスタ相対のアドレッシングが強力で、なんの苦もなくアセンブラで記述できました。

6809とOS-9

6809には、1980年ごろから「OS-9」（後にOS-9/6809）というマルチユーザ、マルチタスクのOSがありました。OS-9はUNIXを参考に、6809のアセンブラで記述されたOSです。OS-9には、Level1とLevel2があり、Level1は特殊なハードウェアのいらない64KBまでのメモリ空間で動きました。Level2は、6809に外付けでMMU（後述）を加えたハードウェアで動作するものです。OS-9 Level2はマッパを前提としていましたが、OS-9 Level1はそういうアドレス変換機構をもたないハードウェアでマルチタスクを実現していました。そのため、やはり実行バイナリコードに工夫がありました。

OS-9 Level1

OS-9 Level1は、8bit CPUである6809の64KB空間（I/Oを含む）で動作したので、モジュール化が徹底しており、デバイスドライバや、重要なシステム機能もモジュールになっていました。システムのモジュールの多くは「マネージャ」という名前で呼ばれ、「xxxman」という名称になっていました。小さなメモリ上には、不要なモジュールは読み込まず小さな構成でも動作するようになっていました。

UNIXをお手本にしたOS-9ですので、アクセス制御は、すべてファイルのアクセス権限になっていました。タスクは、UNIXと同様「プロセス」と呼ばれ、プロセス間通信は、UNIXと似たPIPEで行いました。PIPEはpipemanにより実現され、上位のIOManによって、ファイルと一緒に一元的に管理されていました。UNIXのshellと同様の文法が、OS-9のコマ

注1）当時は「究極の8bit CPU」とも呼ばれていました。

ンドラインにも実現され、「>」、「<」や「¦」を使用したI/Oリダイレクトや、プロセスの入出力の接続が可能でした。

OS-9は、その実行バイナリに、PICなプログラミングを要求しました。PICなコードは、メモリ中のどこに置いて実行を行っても、正しく動作するプログラムです。また、OS-9のプロセスのコードは、同時に再入可能性(リエントランシィ)も要求されました。

OS-9用のBASICとCコンパイラは早期に供給されたので、とくに問題はありませんでしたが、OS-9に対応していないコンパイラや、アセンブリソースコードでは、Uレジスタはスクラッチパッド(一時的ワーキング)として、非常に頻繁に使われましたし、グローバル変数のアクセスは、直に変数のアドレスが埋め込まれるものだったので、事実上、使い物になりませんでした。

メモリ上バイナリの再利用?

OS-9のケチケチ精神が発揮されているのは、プロセス生成時に実行バイナリを探すところでしょう。OS-9/6809では、実行の終了したプログラムコードは管理されておらず、メモリのヒープ領域の候補として放置されています。OS-9は、あるプロセスを開始するとき、まず、全メモリ空間をバイト単位でスキャンし、バイナリコードのヘッダを見つけます。そして、そのヘッダに基づき、バイナリの全バイトのCRC(巡回冗長検査)コードを計算します。そのCRC値が正しい、つまり、そのバイナリがメモリ中で壊れておらず、それが所望のバイナリコードであれば、そのバイナリを再利用します。メモリ上に所望のバイナリコードが見つからない場合、ファイルシステムからメモリ上にバイナリをロードします。

前回紹介したMP/Mは、ローダがメモリ展開時にPRL(Page Relocatable)フォーマットのアドレス解決を行うのに対し、OS-9ではPICを採用し、実行時にまでアドレス計算をさせていたのが対象的です。

6829 MMU

6809にはモトローラの標準品として6829というMMU(Memory Management Unit)のLSIが供給されていました。6829は、メモリマッピング(ページ単位のアドレス変換)だけを行うもので、不在ページの検出などはできません。6829は、複数のタスクのページ変換を行うことが前提となっており、タスクごとに異なるメモリマッピングを使用することで、タスク間のメモリ干渉が発生しないようになってました。

それを正しく行うためには、6829の制御レジスタをユーザプログラムから操作されてはいけません。また、OSからユーザタスクへ制御が移るときに、同時に、OSのメモリマッピングから、ユーザタスクのメモリマップへの切り替えも必要です。それを実現するために、6829は「ヒューズレジスタ」というものを備え、OSから、ユーザタスクに制御が渡る直前に、OSがヒューズレジスタにクロック数を設定し、そのクロック数が経過したときに、6829のメモリマッピングが切り替わり、制御レジスタへのアクセスは不能になる、という仕様でした。

次にOSのメモリマップになり、制御レジスタが操作可能になるのは、割り込み(ソフトウェア割り込み含む)が入ったときです。という純正の6829 MMUがあった6809システムですが、OS-9 Level2は、6829を使用せずに独自のMMU(マッパ)を使用したシステムで多く動作していました。

◆

OS-9/6809は、円熟期には、富士通FM-11、FM-7/8、日立MB-S1という日本でもポピュラーな機械や、アメリカではTRS80 ColorComputer(CoCo)という安価な機械でも動作したので、知っている方も多いかもしれません。

OS-9/68000は、MMUのない68000用も開発され、シャープのX68000で動作し、また、CD-iなどのシステムで長く使用されました。SD

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。

Report

KDDIと技術評論社、「第2回　察知人間コンテスト」のグランプリを発表

KDDIが提供するオープンな「SATCH SDK」を使用したAR（仮想現実感）アプリ開発コンテスト、「第2回察知人間コンテスト」の決勝戦および授賞式が2013年5月29日に開催された。今回で2回目となる本コンテストは、2012年11月1日～2013年1月3日の期間で作品（アイデア）の募集が行われ、1次、2次審査を経たのち最終審査（決勝戦）が行われる。グランプリ賞金は100万円で、さらに副賞としてKDDIより「SATCH VIEWERサイト」および開発者向けサイト「SATCH Developers」を中心に支援が受けられる。

会場は東京・代官山の「シアターサイバード」。50点の応募の中から決勝戦には6作品がノミネートされ、審査員・観客が見守る中、プレゼンテーションとデモンストレーションが行われ会場を沸かせた。受賞結果は次のとおり。

- グランプリ……他人のデバイスのカメラ映像を自分のデバイス画面に表示できる「乗っ取りカメラ」（秋山裕志氏）
- 準グランプリ……ボードを手作りして遊べるすごろく「てづくりARすごろく♪」（チームM.Lab）
- 特別賞……折り紙の折り方をARで教えてくれる「折り紙AR」（弘田月彦氏）、色を認識して音を再生させる「音色カメラ」（田中雅也氏）、落ちてくる立体物の輪郭を描いて得点を競うゲーム「Trap Hole Drawing Game」（H.P.s.b.l）

▲表彰者と審査員による記念撮影

CONTACT　SATCH Developers site
URL http://satch.jp

Hardware

NECアクセステクニカ、「Draft IEEE802.11ac」対応ホームルータのエントリーモデルを発売

NECアクセステクニカ㈱は、無線LAN規格IEEE 802.11の新規格「Draft 11ac」に対応したWi-Fi（無線LAN）ホームルータ「AtermWF800HP」を発売する。

同社はすでに4月に「AtermWG1800HP」などDraft 11acに対応したホームルータを発売しているが、今回の新製品は、スマートフォンユーザ向けに手軽にDraft 11acを利用できることを目的に開発されたエントリーモデル。アンテナは本体内蔵、33×97×146mmの小サイズを実現したコンパクトでスタイリッシュなデザインとなっている。Draft 11ac対応製品のなかでは、国内最小クラスのサイズをほこる。

また、スマートフォンでQRコードを読み取りすることで、Wi-Fiから回線までワンストップで設定できる「らくらくQRスタート2」機能を搭載。スマートフォンだけで簡単に無線LANを利用開始できる。想定価格は9,500円前後。発売は6月20日から。

また、同社は7月に、IEEE802.11n対応の従来機種にらくらくQRスタート2の機能を搭載したリニューアルモデル「AtermWG600HP」「AtermWF300HP」「AtermWG300HP」の発売を予定している。

■AtermWG1800HPの特徴

- Draft 11ac/11n/11a（5GHz）と11n/11b/11g（2.4GHz）を同時利用可能
- NEC先端技術（μEBG構造＋μSRアンテナ）の採用による高速化と小型化を実現
- インターフェースは、WANは100BASE-TXを1つ、LANは100BASE-TXを3つ搭載

▲AtermWF800HP

CONTACT　NECアクセステクニカ㈱
URL http://www.necat.co.jp

Topic

UEIとD2C、スマートフォン向けオリジナルゲーム開発コンテスト「第3回　9leap」開催

㈱ユビキタスエンターテインメント（以下、UEI）と㈱D2Cは、プログラマを目指す青少年の育成を目的としたスマートフォン向けオリジナルゲーム開発コンテスト「9leap（ナインリープ）」を5月1日より開催。前期は8月31日まで作品の募集を行っている（後期は9月1日より開始し、12月31日まで受付）。

前期と後期､それぞれの期間で優秀作品の発表を行い、受賞作品の開発者には最新モデルのMacBook ProまたはMacBook Airが進呈される。さらに、前期後期を通じてもっとも高い評価を得た3作品には「最優秀賞」として、2014年3月17日から21日にかけてサンフランシスコで開催予定のゲーム開発者会議「Game Developers Conference」への視察旅行に、無料で参加できる権利が与えられる。コンテストの概要は次のとおり。

＜募集内容＞
JavaScriptで書かれたスマートフォンで動作するオリジナルゲーム作品
＜応募資格＞
小学生以上25歳以下の学生

CONTACT　9leap公式サイト
URL http://9leap.net/

Hardware

ぷらっとホーム、Webフィルタリングアプライアンスの「小規模版」を発売

ぷらっとホーム㈱は5月13日、Webフィルタリングを簡単に導入できるアプライアンス製品「EasyBlocks Webフィルタリング向けProxyモデル powered by i-FILTER 小規模版」を発売した。

同製品は国内トップシェアのWebフィルタリングソフト「i-FILTER」のエンジンを採用し、書き込みやアップロードなど情報漏洩の原因となるような行為をブロックしたり、標的型攻撃などによる悪性Webサイトへの通信を検知／遮断したり、といった対策が手軽に行える。

ソフトウェアインストールなどは不要で、Webインターフェースも利用可能であるため、自分でサーバ構築を行う場合と比べて簡単に導入／運用できる。価格はオープン価格。1年間は専用サポートデスクの利用が可能（2年目以降は、年間140,000円（税込））。

▲EasyBlocks Webフィルタリング向けProxyモデル powered by i-FILTER 小規模版

CONTACT　ぷらっとホーム㈱
URL http://www.plathome.co.jp

Hardware

Zabbix社とぷらっとホーム、Zabbix 2.0搭載アプライアンス製品を提供開始

5月29日、オープンソース統合監視ソフトウェア「Zabbix」を提供するZabbix社は、ぷらっとホーム製のハードウェアを採用し、最新版Zabbix 2.0を搭載したアプライアンス「Zabbix Enterprise Appliance」の製品提供を開始した。

本製品の導入により、サーバ購入、検証、セットアップ、Zabbixのインストールと初期設定、データベースのチューニング、監視処理を最適に行うための周辺ソフトウェアの設定といった煩雑な作業を行うことなく、システム監視、障害通知、グラフ表示などのシステム監視運用業務を容易に開始できる。

本体価格は298,000円（税抜）。H/W先出しセンドバック保守やZabbixアップデートなどの保守費用は、100,000円（税抜）。また、本製品を購入したユーザは2013年7月より開催するZabbix入門トレーニング（通常18,900円（税込））に無料で参加できる。

▲Zabbix Enterprise Appliance

CONTACT　Zabbix社
URL http://www.zabbix.com/jp

Service

at+link、ioDrive2搭載のサーバパッケージ／プライベートクラウドサービスのハイスペックプランの提供を開始

ioDrive2搭載サーバを提供開始

㈱リンクと㈱エーティーワークスが共同で展開しているホスティングサービス「at+link」は、ソーシャルゲームやアプリ事業者向けサーバ環境「at+linkアプリプラットフォーム」において、Fusion-io社の超高速NAND型フラッシュメモリ「ioDrive2」を搭載したサーバの提供を5月8日より開始した。

at+linkアプリプラットフォームは、2010年11月から提供を開始しているソーシャルゲームアプリ提供用の専用サーバパッケージ。複数のWebサーバとioDriveを搭載したDBサーバ、大容量回線やネットワーク機器のほか、アプリの運用に必要な機能をまるごとパッケージにし、初期費用無償で提供する点を特徴としている。

このたび、at+linkはioDriveの次バージョンであるioDrive2を搭載したサーバの提供を開始した。同社の調べによると、ioDrive2は、ioDriveの3倍以上の速度、HDD（SATA）の約80倍もの速度が出たという。

アプリプラットフォームの価格は月額330,000円（税抜き）ioDrive搭載サーバなしプランの場合は180,000円～。WebサーバとDBサーバの台数はコンテンツの内容や規模により自由に構成できる。

月額25,000円のハイスペックプランを提供開始

at+linkは、プライベートクラウドサービス「at+linkプライベートクラウド・ライト」において、上位スペックプランの提供を5月15日より開始した。

at+linkプライベートクラウド・ライトは、2013年2月から提供を開始しているプライベートクラウドのホスティングサービス。サーバ1台をまるまる自社専用のクラウド環境として利用でき、初期費用0円、月額15,000円（税抜き）の定額で、仮想サーバを必要なタイミングで専用の管理パネルから手軽に作成できる。

このたび、従来より提供していたプランAよりもCPUスペックが高く、メモリやディスク容量が増え、かつディスクもRAID構成で、バックアップ用ディスクも搭載された、よりデータ保全性が高いプランBを追加した。価格は初期費用150,000円、月額25,000円（税抜き）。

また、プランBの提供開始と同時に、プランA/Bのいずれにおいてもカスタムの使用が可能になった。WindowsやUbuntu、DebianなどのOSのISOイメージをVPSにアップロードして自由に使える。

▼at+linkプライベートクラウド・ライト仕様と料金

	プランA	プランB
CPU	Xeon E3-1265Lv2	Xeon E5-2603
メモリ	8GB	24GB（最大 48GB）
HDD	500GB＋500GB（バックアップ）	1TB×3［RAID 1＋バックアップ］
グローバル IP アドレス	/28（16）	/27（32）
初期費用	0円	150,000円
月間利用料	15,000円	25,000円

CONTACT at+link
URL http://www.at-link.ad.jp

Software

シマンテック、クラウドサービス利用のセキュリティを向上する「Symantec O_3」を発表

㈱シマンテックは5月20日、統合セキュリティサービス「Symantec O_3（シマンテック オースリー）」を発表。同日より提供を開始した。このサービスは同社が完全子会社化したVeriSignの認証技術を生かした最初の製品となる。

いまや企業内でも多種多様なクラウドサービス――Google Apps、Office 365、Salesforce、AWSなど――を複数利用するようになっているが、一方で煩雑化するパスワードや複数のデバイスでの利用、コンプライアンスの観点などから企業での管理が問題にもなっている。

この問題を解決するために、Symantec O_3では、二要素認証も可能なシングルサインオン、クラウドサービスによって異なるログデータの一元管理・可視化、状況に応じたセキュリティポリシーの設定といった機能が提供される。さらにモバイルデバイスのセキュリティ対応には、「Symantec App Center」との連係も図れ、より安全な運用が可能になる。参考価格は年間1ユーザあたり3,000円から（1万人規模での導入の場合）。

CONTACT ㈱シマンテック
URL http://www.symantec.com/ja/jp/index.jsp

Report

デル・ソフトウェア、「Dell Softwareパートナー・イベント」にて製品のポートフォリオを拡充を発表

5月16日、ホテル日航東京（東京都港区）において、Dell Softwareパートナー・イベントが開催された。

同イベントを主催するデル・ソフトウェア㈱は、2012年までは日本クエスト・ソフトウェア㈱（以下、クエスト）として、「NetVault Backup」「vRanger」などのデータ保護をはじめとする数々のソフトウェア製品を販売してきた。2012年9月に米国Dell社が米国Quest Software社を買収したため、同社も2013年2月1日に、社名をデル・ソフトウェアに変更している。

こうした背景のなか、イベントが行われた5月16日、同社はソフトウェア事業のさらなる拡大のため、ソフトウェア製品のポートフォリオを拡充することを発表。今後はクエストの製品群に加えて、Dellが買収したKACE、SonicWALLなどの製品群が加わる。

イベントの第1部では、デル㈱の代表取締役社長の郡信一郎氏、デル・ソフトウェアの代表取締役社長バスター・ブラウン氏が登壇し、日本における事業戦略について発表した。

Dellというと、直販でのビジネスモデルのイメージが強いが、デル・ソフトウェアになってもパートナー企業との協業によるチャネル販売を継続していくことが明らかにされた。これまでクエストと協業し、同社の製品を販売してきたパートナー企業は、今後は、Dellが用意するパートナープログラム「PartnerDirect」に移行して、同社の製品を販売していく。

第2部では、同社が扱う各種製品の最新動向について発表が行われた。データ保護については、テクニカルサービスマネージャの下館英之氏より「NetVault Backup」（以下、NVBU）などの発表が行われた。

NVBUは、Linux市場では10年連続で50%以上のシェアを持つバックアップソフトウェア。

最新のNVBU 9.0.1ではWindows Server 2012とNovell OESを新たにサポートした（Windows Server 2012は一部制限事項あり）。また、EMC Data Domain Boostのサポート、NetVault SmartDiskのレプリケーション機能拡張、Dell DR4000（Dell重複排除アプライアンス）のサポートなども行われているという。


▲デル・ソフトウェアの代表取締役社長バスター・ブラウン氏

現在、NVBUのWindows Server 2012のサポートは、「Windows Server 2012をNVBUサーバとして利用できない」「ReFSファイルシステムには対応していない」といった制限つきのサポートだが、2013年の第2四半期中にはフルサポートされる予定とのこと。

第2部の後半では、Dell傘下になったことで新たに加わったファイアウォール「SonicWALL」、システム管理「KACE」といったアプライアンス製品についても紹介が行われた。

CONTACT デル・ソフトウェア㈱
URL http://www.questsoftware.jp

Software

デル・ソフトウェア、レプリケーションソフトウェア「SharePlex 8.0」をリリース

デル･ソフトウェア㈱は5月23日、データレプリケーションソリューションの最新バージョン「SharePlex 8.0」をリリースした。

これまでOracleデータベース間でデータレプリケーションを実現してきた経験を活かし、最新版のSharePlex 8.0では、Oracle変更データを多様な構造化／非構造化データベースに転送できるようになった。具体的には、チェンジ・データ・キャプチャ（CDC）を使用して、SQL Server、DB2、Sybase、Netezza、Teradataなどの主要な構造化データベースのほか、HadoopやGreenplumなどの非構造化データベースを含むさまざまなデータベースに転送できる。

CDCは、データソース内で変更のあったデータのみが転送されるため、従来のデータ転送に比べ、より高速に転送が行える。

標準価格は、Oracle Standard Edition用ライセンスの最小構成で135万円～（初年度保守費用込み、税抜き）。移行専用の期間ライセンスも提供している。出荷開始は、2013年7月の予定。

CONTACT デル・ソフトウェア㈱
URL http://www.questsoftware.jp

Letters from Readers

Google Glass、使ってみたいけれど

Google Glassがいろいろと物議を醸していますね。「顔認識技術、使えると便利だけれどプライバシー的に問題アリ」「ナビゲーション機能、歩きながら使っていて危なくないのか」などなど。しかし、一番気になるのは、ぶつぶつしゃべりながら操作しているところを人に見られるのはなんだか恥ずかしそう……。そんなことを気にしているのは、担当だけでしょうか?

2013年5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　IT業界ビギナーのための お勧め書籍55冊+α

今年、IT業界に入ったばかりの新人さん向けに、それぞれの分野で実績をあげている方々に、技術の源となっている本や、ビジネスや自分のこころのあり方の柱となっている本を紹介してもらいました。

このために購入した感じ。専門書は1冊の価格が高いので、できるだけ失敗を避けたい。推薦情報は参考になる。

神奈川県／masakiasaさん

数冊購入したくなりました。また既読の本もあり感慨深い特集でした。

東京都／IYOSHIFUさん

私自身が新卒(IT業界ビギナー)なので、今回の特集は非常に参考になりました。いくつか気になる書籍もありましたので、購入を検討しています。

東京都／ひよこ大佐さん

55冊すべては読めませんが、1冊でも2冊でも読破しようと思っています。

大阪府／澤下さん

今回挙げられた推薦書を読んでみて、どの程度理解できるか。そんな見方をしてみれば、自分がその分野にどの程度精通しているかの目安にもなりそうですね。

第2特集 正規表現をマスターしていますか?

正規表現はいろいろな場面で使用できるため、マスターすれば非常に強力な武器になります。基本的な使い方から、正規表現エンジンのしくみまで幅広く解説しました。

よく使う割に独立して勉強することが少なかったので、実用的な内容がまとまっているのはありがたい。

埼玉県／山本さん

シェルスクリプト、awk、Perlが使えれば一通りのサーバ運用は可能になると考えています(Pythonもできればなお良い)。サーバ運用をしていくうえで、ログ解析はやっぱりPerlの正規表現。今回は正規表現についてあいまいに覚えていた部分を明確にし、こんな表現もあったんだと気づかされることがあり、たいへんためになりました。個人的には定期的にこのような気づきが必要だと思っています。

大阪府／しろたんさん

第2特集第4章「正規表現エンジンの種類としくみ」の内容は凄かったです。著者の素養とユーモアを感じさせるとともに、GNU信者であることを匂わせる書き方にとても惹きつけられました。

東京都／ひろやすさん

正規表現を日常業務で使っているという読者は多いようです。ある程度使いこなしている方でも、正規表現エンジンのしくみは、目新しい情報だったのではないでしょうか?

一般記事　バーチャルネットワーク コントローラ2.0開発の実際

OpenFlowを簡単に導入できるフレームワーク「バーチャルネットワークコントローラ2.0」の開発秘話を紹介しました。

最近は本当にソフトウェアが担う領域が拡大してきたなあ、と実感します。

東京都／k4aさん

ネットワークの仮想化は現在話題になっており、ネットワークの構築および管理する立場としてSDN(Software-Defined Networking)とOpenFlowの知識を習得することができて良かったです。

茨城県／Rubyさん

SDNは、エンドユーザには身近な技術ではないかもしれません。しかし、先日、The Linux Foundationが

SDNのオープンソースフレームワーク「OpenDaylight」プロジェクトを立ち上げましたし、ネットワーク業界全体としてみると、SDNの取り組みは着々と進んでいるようです。

一般記事　Hadoopは基幹業務をどう変えるのか

ソフトバンクモバイルのバッチ処理におけるオープンソース活用、とくにHadoop導入の過程を紹介しました。

Hadoopは、今後、扱うデータが増加してくるであろう会社には必要になってくると思う。

大阪府／Ikepさん

IT関係の会社の業務を知ることができた。完成された商品が出回る中、作業の流れや工夫が述べられている記事はたいへん参考になる。

高知県／音田さん

業務処理にもどんどんOSSが使われるようになってきました。とくにHadoopは利用するのが難しいと言われているだけに、実際に業務処理への導入事例は参考にすべきことが多かったのではないでしょうか。

連載

腱鞘炎になりかけてから自宅でも職場でもエルゴノミクスキーボードを使っています。好奇の目もなんのその。マニアックな「偏愛キーボード図鑑」を楽しみにしています!

神奈川県／kesuidaさん

「iPhoneブックアプリ開発」というiOSの連載が新たにスタート。たいへんうれしいです。残念なのは、私がいまだにSnow Leopardユーザであること。OSをアップデートしなさい、というお告げなのでしょうか。

福岡県／池内さん

自宅ラックのススメを見て「自分も欲しい」といけないフラグが立ってしまいました。以降も楽しみにしています。

石川県／Keiさん

5月号より、4つの新連載が始まりました。読者のみなさんからの感想もたくさんいただきました。さらに、6月号、7月号でも新連載が目白押しです。また、感想をどしどしお送りください。

エンジニアの能率を高める一品

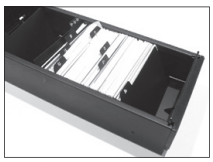
仕事の能率を高める道具は、ソフトウェアやスマートフォンのようなデジタルなものだけではありません。
このコーナーではエンジニアの能率アップ心をくすぐるアナログな文具、雑貨、小物を紹介していきます。

スマデポ（名刺整理器）

2,100円（税込）／カール事務器　http://www.carl.co.jp/


▲写真3

今回は名刺の整理グッズを紹介します。「スマデポ」は名刺1000枚を収納できる箱です。付属の見出しカードで分類しながら収納できます（写真1）。本製品の独特なところは、箱のふたに名刺を挿し込むスリットとiPhoneを立てかける台があること。そこに名刺とiPhoneをセットして撮影すれば、最適なサイズと角度で名刺を撮影しデジタル化できます（写真2）。iPhone用アプリで撮影とデータ管理を行い、名刺は箱に入れて保管しようというわけです。ただ、担当は名刺（紙）を箱だけで管理するのも悪くないと思います。ファイル型の名刺整理グッズと違い、ポイポイ収納できますし、並び替えや分類の変更もすぐにできます。箱の側面（前後）が前に開く（写真3）ので、箱に入れたままで名刺をパラパラとめくりやすいのも便利。撮影すら億劫という人は箱に分類して整理するだけでも、いざというとき役立ちます。


▲写真1


▲写真2

5月号のプレゼント当選者は、次の皆さまです

①スマートフォン用車載ホルダー形FMトランスミッター ..... 埼玉県　南雲浩二様
②ツイスバソーダスターターキット ..... 埼玉県　新谷明彦様

★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
August 2013

2013年8月号
定価1,280円　176ページ

7月18日発売

［第1特集］理論から実践まで

3Dプリンタが未来を拓く理由

簡単にモックアップが作れる！　カスタム製品ができると話題の3Dプリンタ。その実力はいかに？　さまざまな疑問を解消しつつ、理論を押さえ、実際にモノを作るときに何が必要なのか、現場のプロがこれまでの実績を公開します。3Dデータの作成から、3D出力までいっきに紹介します。

［第2特集］ソフトウェアのバグを見抜く

エンジニアの「慧眼＝見通す力」の身につけ方

ソフトウェアのレビュー方法から見えてくるバグのありか。ソースコードの姿から見えてくる、さまざまなバグの形。銀の弾丸はあるのか？　バグを狙い撃つ方法論を紹介します。

［一般記事］

エンジニアとして楽ちんに生きるためには

※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。

休載のお知らせ

「システムで必要なことはすべてUNIXから学んだ」（第10回）は都合によりお休みいたします。

SD Staff Room

●雑誌を作りながら書籍を作らねばならず、久々に自分に喝をいれつつ進行。この号が出るときちょうど見本が出ます。乙武さんのイタリアンレストラン入店拒否事件でネットが盛り上がっていることを、備忘として書いておこう。1月後にはどうなっているのかな。人の噂も七十五日。ネットの噂はもっと短い？（本）

●この号が出る頃には、Haswellの新しい名前が決まって、Appleの新製品やらNexusの新型やらが出てきているのであろう。新製品の購買意欲があるということは元気な証拠だ。モバイル系買うと持ち歩きたくなるが、どんどん増えて鞄が重くなって困る。でも一番重いのはMacBook Proなんだよなぁ。（幕）

●秩父方面に遊びに行きました。山の斜面に作られた東屋にはハンモックがあって、『よつばと』愛読者の子どもたちは興味津々。ところが娘はほんとうにマンガのようにクルリと一回転して地べたに落下し、痛さと恐怖でもはや乗るのを拒絶。でも帰り際にはリベンジを誓ってたんで、いい経験になったかな？（キ）

●たまに外食するときは何を食べようか迷います。無性にカレーライスが食べたくなるときがあるのですが、必ず「そんなの家でも食べられるだろ」という葛藤が起こります。しかし、外食でしか食べられないものとなると値段が高い。いろいろ考えたあげく結局、うどんになります（家でも食べられるだろ!）。（よし）

●念願のフードプロセッサーを購入。最近よく作るのがなすディップ。焼きなす3本の皮を剥いて、練りゴマ、オリーブオイル、レモン汁大さじ1、塩小さじ1、にんにく1かけ（すりおろし）、クミン少々でまわすだけ。パンにのせて、パセリか刻んだトマトをのせると、ちょっと小洒落た前菜です。（北）

●父親が無事退院しました。といってもしばらくは自宅療養を言い渡され、遠出することはできないようです。再発リスクもあるのですが、発生原因が解明されていないそうなので、とりあえず食生活を改善中。私は父親の体質を受け継いでいるらしいので、一緒に食生活を改善したほうがいいんだろうな。（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
2013年7月号

発行日
2013年7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。