〈クラウドの事例紹介〉

4 クラウド技術とオープンソース

藤田昭人　（株）IIJイノベーションインスティテュート

本稿では，クラウドコンピューティングに由来するオープンソース・クローンについて，その代表例であるHadoopを中心に筆者の経験も交えて紹介したい．クラウドコンピューティングの技術は本来大規模データセンタ向けに開発されたが，そのオープンソース・クローンを活用することにより比較的容易に構築可能な数台から数十台の小規模なクラスタ・システムにおいても，動画像などの大容量ファイルの格納やインターネットなどから取得し大量に蓄積されたデータの検索などのプライベートな用途にも有効に活用できるのではないかと筆者は考えている．

Hadoop MapReduce

今や"クラウドコンピューティング"は2009年のIT業界最大のムーブメントになりそうな勢いである．筆者はちょうど1年あまり前にこの分野に関する技術調査に着手したのだが，世間のあまりの変わりようにただ驚くばかりである．もっとも，当時は情報不足により，現在は情報過多により"クラウド"なるものの実相がサッパリ分からないことだけはあまり変わりがない．

最近では"クラウドコンピューティング"という用語はITビジネスでのバズ・ワードとして使われることが多いように感じられるのだが，これもユーティリティ・コンピューティングの概念と結びついた「使いたいときに使いたいだけ利用できるコンピューティング・リソース」のシナリオが影響しているのかもしれない．技術的には仮想化技術にばかりに注目が集まっている印象がある．Amazon EC2/S3などのクラウド・サービスを利用するという前提で着目すると，その基盤を支える大規模分散技術などは「事業者の問題」ということになるのだろう．事実クラウドの大規模分散技術は主にサービス事業者の手で開発されたプロプライエタリなソフトウェアによって実現されている．それゆえこれらの技術について詳細まで語られることは少なかったことも，研究者の注目を集めにくい1つの要因だったように思う．

しかしながら，近年これらのソフトウェアに関する限られた公開情報を元に再開発され等価な（ではないかもしれないが）機能を提供するオープンソース・クローンが登場するようになってきた．もちろん重要な情報が欠落した論文のみを頼りに開発されたシステムなので，この手のクローンにはおのずと限界があるのだが，現在のクラウドコンピューティングの領域においてそれ以外に手元で確認できる手段がないのも，これまた事実である．

クラウドコンピューティングを始めた筆者がが最初に手がけたのはHadoop[6),7)]であった．今やクラウド系オープンソース・クローンの代表格であるのでこのシステムの名前を目にした方々も多いのではないか．ちなみに，この風変わりな名前は開発者Doug Cuttingの娘さんの象のぬいぐるみの名前に由来しているそうだ．ロゴを見ればどんなぬいぐるみかは一目瞭然である．

■Hadoop と Google 開発技術の関係

MapReduce[1)]はGoogleが開発した分散並列処理のためのプログラミング・フレームワークであり，Map/Reduceと呼ばれる2つの関数を定義するだけで容易に分散並列処理アプリケーションが作成できる．ご存知のとおりGoogleが開発したシステムは原則的には非公開なのだが，その技術的な概要は論文やGoogleのサーチエンジンの基本的なアーキテクチュアに関して，彼らが起業する以前に書かれた論文が公表されている[2)〜5)]．

Hadoopに付属するMapReduceは文献1）の情報を参考に再開発されたオープンソース・クローンである．当初のHadoopはこのMapReduceとGoogle File System（GFS）[2)]を参考に開発されたHadoop DFS（HDFS）の2つのコンポーネントから構成されていた．その後もGoogleの分散並列処理基盤と分散ストレージの2階層から構成されるアーキテクチュアをなぞる形でコンポー

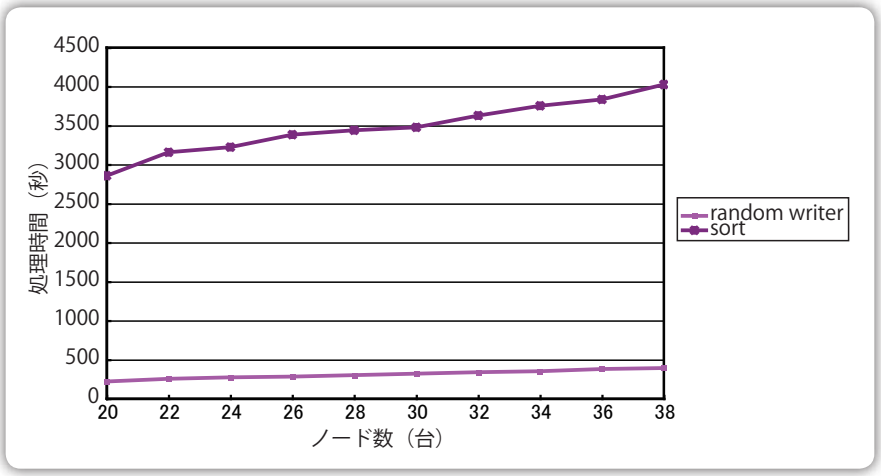

図 -1　Hardware Benchmark

ネントが追加されて現在に至っている．

■MapReduce の実行

私がまず手をつけたのは MapReduce であった．文献 2）で紹介されている MapReduce は「定型フォームの大量データを効率よく分散並列で実行してくれるプログラミング・フレームワーク」で，「map と reduce の 2 つの関数（Hadoop ではクラスだが）を書くだけで，任意のデータに対する分散並列処理を容易に実行できる」のだが，総データ量が数十ギガバイト以上にもなるテーブルやインデックスなどなかなか用意できるものではない．結局 Hadoop Wiki で紹介されていた Hardware Benchmark の方法で 40 台のクラスタの処理能力を計ってみることにした．今のところ事実上の Hadoop の標準的なベンチマークテストのようである．

Hadoop の Hardware Benchmark は，MapReduce のサンプル・アプリケーションである Sort をそのまま使って，MapReduce の処理能力を計測する非常に簡単なベンチマーク方法である．同じくサンプル・アプリケーションである RandomWriter を使って機械的に生成した入力ファイルを使用するので，Hadoop が動いている環境であれば直ちに性能を計測することが可能である．

RamdomWriter はデフォルトでクラスタを構成するノード 1 台あたり 1GB のランダム・データファイルを 10 本，すなわち合計 10GB のランダム・データを生成する．これを入力とした Sort の実行時間を計測して処理能力の指標とする．1 ノードあたり 10GB なので全体のデータ量はクラスタのノード総数に対しリニアに増大する．このベンチマークテストは，仮に同じスペックのコンピュータだけで構成されるホモジニアスなクラスタで，クラスタネットワークが理想的な状態（通信オーバヘッドがゼロ）である場合には Sort の処理時間はほぼ一定の値になることが期待される．が，実際にはスペックの異なるノードが混在するヘテロジニアスなクラスタであることが一般的であろうし，クラスタネットワークにはオーバヘッドが発生するので，なかなか目論みどおりには動かないようである．

図 -1 は筆者が使用する 40 台のクラスタでの Hardware Benchmark の実行結果をグラフにしたものである．2 種類のスペックの PC を各々 20 台ずつ接続したヘテロジニアスなクラスタで，Fast Ethernet のスイッチで相互接続をしている．また使用した Hadoop は 0.19.1 であり，NameNode と JobTracker は各々専用に 1 台ずつ割り当て，残る 38 台をスレーブとして使うコンフィグレーションを使用している．グラフはスレーブ PC の接続台数を 20 台から 38 台まで 2 台刻みで増加させたときの RamdomWriter および Sort の処理時間をプロットしたものである．もちろん Java なのでガベージ・コレクション等の影響が懸念される．この計測では都合 5 回計測して，その最大値と最小値を除外した 3 つの値の平均を処理時間とした．グラフから分かるように RamdomWriter/Sort ともノード数が増大するごとに処理時間もおおむねリニアに増大している．リニアに増加している原因は主にクラスタ・ネットワークのオーバヘッドに起因していると今のところ推測している．

■MapReduce アプリケーションの作成

添付されているベンチマーク・プログラムを実行するだけで分かることには限りがある．そこでテストを目的とした MapReduce アプリケーションを作成することにした．たまたま，Kikker と呼ばれるはてなブックマークの新着リストを利用したリコメンド・システム [14] が紹介されていたので，このソフトウェアをベースに MapReduce アプリケーションに作り直した．以降この作業での知見を述べる．

まず MapReduce の並列実行の特性を活かせるよう

に map/reduce を定義するのは意外に難しいことである．MapReduce 版 Kikker では，はてなブックマークの新着リスト・ページから抽出した Web ページの URL リストを入力とし，クローリングと評価ベクトルの計算を行う MapReduce プログラムを書いたのだが，即時性が重要である Kikker では頻繁にクロールを行うため，1 回の処理において入力となる URL リストのエントリ数は比較的少数であった．MapReduce では，入力のエントリ数が少なければ並列度も小さくなってしまい，むしろ MapReduce を使うオーバヘッドの方が目立ってしまうケースがあった．やはりエントリ数が膨大なデータアーカイブをスキャンするようなアプリケーションでないと MapReduce の特性を活かせないような感想を持った．

また処理対象となるファイルのサイズが小さいとアクセス性能が低下してしまう可能性がある．これは MapReduce というよりはむしろ HDFS が原因なのだが，HDFS は GFS と同様の大容量ファイルを対象に高スループット・アクセスに最適化されたファイルシステムである．そのためサイズが小さなファイルでは通常のファイルシステムよりもアクセス性能が低いこともある．一般的な Web ページのファイル・サイズは数 KB ～数百 KB 程度なので，HDFS の上にクロールした Web ページを展開する設計はあまり得策ではない．

ちなみに Google ではこのような比較的データ容量の小さなファイルは BigTable [3] に格納する方法を取っているようである．BigTable は一般的なリレーショナル・データベースと同様にテーブル構造でデータ格納するデータベースである．個々のデータはメモリ上のテーブル構造の中に格納されるが，そのテーブル構造自体は複数に分割され，GFS にファイルとして格納される．したがって BigTable を活用すると，複数のデータ容量の小さなファイルをまとめたアーカイブ・ファイルな形で GFS には格納されるので，前述のような問題は回避できるようである．

MapReduce は確かに並列処理を比較的容易に実現できるプログラミング・フレームワークだが，効率の良い並列処理を実現するためには MapReduce 以外の要因も考慮してよく考えて設計する必要があり，仮にすでに実装された MapReduce プログラムの効率を改善するためには map/reduce のインタフェースを見直す必要があるのでプログラムを大きく修正する可能性もあるように思う．「とっつきやすいが使いこなすのは難しい」というのが私の個人的な感想である．

Hadoop DFS のスケーラビリティ

前述の MapReduce アプリケーションの作成の知見から，並列実行に特徴のあるクラウド向けアプリケーションを効率よく実行するためには，まず分散ストレージの最適な利用法を明らかにする必要があると考えた．たとえば Google File System のような大容量ファイルに最適化された分散ファイルシステムや BigTable のような並列性の高い分散データベースを最適に利用できるのはどのような条件の場合か？といった疑問である．

■HDFS のスケーラビリティの計測

そこでまず HDFS のスケーラビリティ特性に注目し，Hadoop に添付されている TestDFSIO を使って，対象を HDFS に絞ったベンチマークを試みた．

TestDFSIO はコマンド引数で指定した条件に基づいて任意のサイズのファイルを任意の数だけ作成・参照し，その実行時間，スループット，平均 IO レートと標準偏差を計測する MapReduce アプリケーションである．シーケンシャルにファイルを作成・参照できるオプションもあるのだが，デフォルトでは個々のファイルの作成・参照を map に割り当てることにより，並列的にファイル IO を行う．

筆者は HDFS の Replication Factor（RF）を 1 および 3 に設定して，ノード数を変更しながら順次 TestDFSIO を使って 1GB ファイル 100 本の作成と参照を行った．ベンチマーク環境は前述の Hardware Benchmark と同じである．処理時間やスループットはノード総数の増加に対しリニアに増加すると予測される．Replication Factor は HDFS 内部で保管するデータのコピー数を決めるパラメータで，3 を設定した場合には同じデータが異なるノードに 3 つ作成される．したがって RF を増やすと WRITE のデータ総量が増えるのでファイル作成の処理時間は増えるが，いずれかのコピーを READ すればよいのでファイル参照の処理時間は減ることが期待される．

得られた計測結果のうちスループットをグラフ化したものを図 -2 に，さらに値が突出している RF を 1 に設定した時のファイル作成を除外したグラフを図 -3 に示す．この計測でも都合 5 回計測して，その最大値と最小値を除外した 3 つの値の平均をスループットとした．スループットがノード総数の増加に対しリニアに増加する推測はある程度正しかったが，ノード数が 10 以下のファイル参照では異なる結果になった．可能性としてはノードの過負荷やネットワークの輻輳が原因と考えられるが，今のところ特定するに至っていない．また RF=1 のときに突出したスループット値が計測される理由も現時点では不明である．

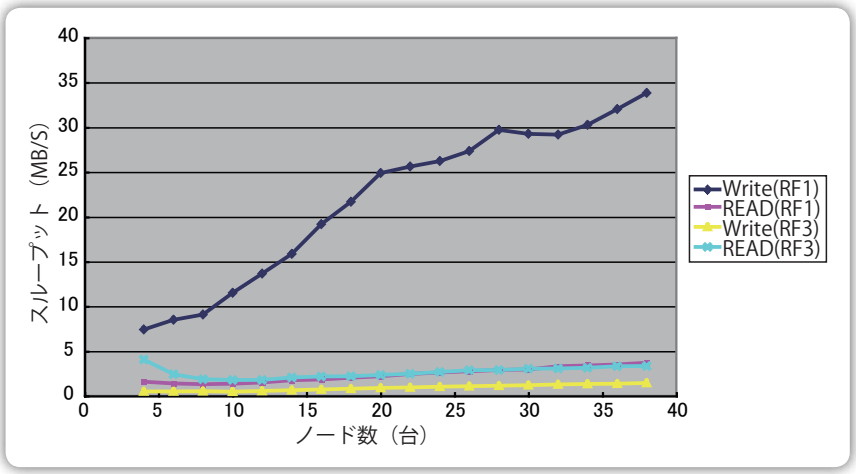

図 -2　TestDFSIO Throughput

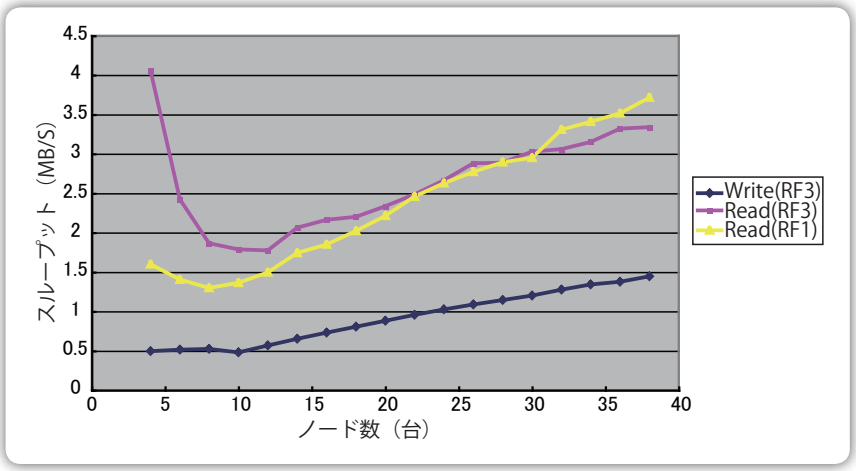

図 -3　TestDFSIO Throughput

その他のクラウド系オープンソース・クローン

以降，筆者が知る Hadoop 以外のクラウド系オープンソース・クローンについて紹介する．

■Kosmos File System（CLOUDSTORE）

Kosmos File System（KFS）[8] は HDFS 互換の大容量向け分散ファイルシステムである．HDFS と同様に GFS のアーキテクチャに基づいて設計されているが C++ による実装が試みられていることが最大の特徴である．実装言語としての Java と C/C++ の優位性の比較についてはさまざまな意見があるが，KFS に関してはファイルシステムへの多様なアクセスが可能なシステムコールを直接呼び出せるという点で C/C++ 実装の高速性が期待できるという指摘もある．

KFS は HDFS の外部インタフェースをサポートしており，Hadoop も 0.18 から KDF のサポートコードをバンドルするようになった．したがって KFS の上で Hadoop MapReduce アプリケーションを実行することが可能である．現在，筆者の環境では KFS を対象に前述の TestDFSIO の実行を試みている最中であるが，今のところ HDFS のパフォーマンスを凌駕するような結果は得られていない．特に TestDFSIO が生成する 1GB といった大容量ファイルを対象にしたアクセスについて HDFS のインタフェースを介した Java から C++ への呼び出しの振舞いは，パフォーマンスに大きな影響を与えると予測されるため細かく把握しておく必要があると考えている．

また筆者の周辺では C/C++ アプリケーションから GFS ライクな大容量ファイルシステムを利用する需要が存在する．Hadoop 自身は libhdfs と呼ばれる C アプリケーションから HDFS を呼び出すインタフェースが提供されているが，内部は JINI を使って Java で定義されたインタフェースを呼び出しているので大容量ファイルへのアクセス時のパフォーマンスに懸念がある．このような GFS ライクな大容量ファイルシステムに対する各種プログラミング言語からの効率の良い呼び出しインタフェースを考えた場合，多様なプログラミング言語からの

インタフェースを整えやすい C/C++ による実装の KFS は一定の役割を担うのではないかと想像している．

■2 つの BigTable クローン

Hadoop のサブプロジェクトである HBase[9] および Hypertable[10] は共に BigTable のアーキテクチャを踏襲するクラウド系分散データベースである．両者は元々 1 つの開発チームであったが，実装方針の違い，特にプログラミング言語として C++ と Java のいずれを採用するか？という問題で議論があり，2 つのグループに分かれて開発を進めることになったそうである．

オリジナルの BigTable が GFS/Chubby といった Google の他のシステムを必要とするのと同様に両者ともシステムを稼働させるためには外部システムが必要となる．GFS の代替システムとしては両者とも HDFS を使用しているが，分散ロックマネージャである Chubby に関しては，HBase が同じ Hadoop のサブプロジェクトである ZooKeeper のコードを利用するのに対し，Hypertable は独自に開発した Hyperspace（Hypertable にバンドルされている）を利用する．

一般にクラウド系分散データベースだと目されている BigTable だが，実際には（大量のデータを格納できる）並列検索をサポートするデータ・ストレージ，すなわち GFS ライクな分散ファイルシステムが不得手な，小容量のファイルを大量に格納できるストレージ・システムだと筆者は理解している．クラウドでの主要なアプリケーションと目されている SaaS などの Web アプリケーションではこのようなストレージは必須なので，前述の両者は既存の SQL データベースとの対比で優劣を評価されることになるのであろう．特にプライベート・クラウドでは両者が特徴とするスケーラビリティはそれほど重視されない．数十から 100 ぐらいまでの小規模〜中規模のクラスタでの使用において既存の SQL データベースに対し何らかの優位性を発揮できるのか？が課題である．

同時に優れたプログラミング・インタフェースのサポートも重要である．この分野では既存の SQL データベースは大きな優位性を持っている．Web アプリケーション向けの各種スクリプト言語に対応したクラウド系分散データベースの優位性を発揮できるプログラミング・インタフェースが重要であろう．BigTable クローンはこの分野の開発ベースになり得るのだろうが，Thrift による各種スクリプト言語への対応済みの Hyperspace がこの面では若干優位にあると思う．

■Amazon のシステムのオープンソース・クローン

本稿ではここまで取り扱ってこなかったが，Amazon のクラウド系技術に対するオープンソース・クローンも存在する．Eucalyptus[12] は Amazon Web Service（AWS），すなわち Amazon EC2/S3 のクローンである．また KAI[13] は Amazon が開発したクラウド系分散データベースである Dynamo のクローンである．残念ながら筆者は今のところいずれのシステムも手を付けていないので，ここでは紹介するにとどめたい．

おわりに

筆者は過去 1 年間，クラウド技術の 1 つの事例として Hadoop の振舞いを理解することに多くの時間を費やしてきた．着手した段階で明確な目標が存在したわけではないので，どちらかといえば具体的な研究目標を見出すための試みということになる．

この試みを通して筆者自身が自覚しつつあるのは「クラウド技術はその前提となるクラウド・サービスから切り離しても有効性の高い技術なのではないか？」という仮説である．この技術は本来，大規模なデータセンタでの運用を前提にコンピューティング・リソースをサービスとして提供する目的で開発されてきた．数千台規模のクラスタでなければ有効に活用できないのだとしたら既存のクラウド・サービス以外の活用は見込めないのだが，数台から数十台程度の規模の一般的なクラスタでも有効な活用法が見出せるのだとしたら，昨今の PC ハードウェアのさらなる低価格化とあいまって，より一般的なコンピューティング技術として定着する可能性がある．

仮説の検証の過程において何らかの結論を述べるのは非常に難しいが，現時点で筆者が認識する課題は次のとおりである．

- 現在のクラウド系オープンソース・クローンを安定的に稼働させるのには非常に骨が折れる．その最大の原因はこれらのソフトウェアが汎用的な基本機能しか提供しておらず，多数のチューニング・パラメータを有しており，その適値を見出すのが難しいことにある．対象とするソリューションを絞り込むことにより，この問題を単純化できるのではないかと考えている．
- クラスタ全体のパフォーマンスは当初想像していた以上にクラスタ・ネットワークのパフォーマンスに依存しているように見える．クラウド系オープンソース・クローンの通信方式を見直す作業は重要だと考えているが，やはりクラスタ・ネットワークの帯域幅の拡大が直接的な効果があるように思う．高速な LAN スイッチの低価格化を期待している．
- 並列実行によるパフォーマンス改善がクラウド技術の特徴であるが，それを効果的に活用するためにはアプリケーションを開発する上での工夫が必要なのではな

いか？と筆者は考えている．クラウド・アプリケーション向けのフレームワークやデバッグ環境について議論されることは今のところ少ないように思うが，今後は重要な課題の 1 つになっていくように思う．

・クラウドコンピューティングは特にサーチ・エンジンを始めとする大量のデータを解析し有益な 2 次的情報を抽出する用途で効果を発揮することが知られているが，利用者が所有する小規模クラウドをこの用途に活用するためにはデータ収集が大きな課題となる．既存のクラウド・サービスや他の小規模クラウドとのデータの交換や共有を図るための手段も今後の重要な課題となると思う．

最近ではクラウドコンピューティングをコンピューティング・リソースを従量課金で提供するサービス，すなわちユーティリティ・コンピューティングと捉える考え方が提唱されている．その説明においてクラウド・サービスを提供するデータセンタは発電所に例えられることが多い．今日このシナリオの自家発電から発電所へと移行していった過去の経緯を根拠に，小規模クラウドの将来性に疑問を呈する意見も多いのだが，それならば工場や研究施設に設置される自家発電設備，近年一般家庭に普及し始めている家庭用発電設備，さらに今後は増加してくるであろう電気自動車向けのスタンドなどはどうなのか？というのが筆者の反論である．

筆者が関心を持つ小規模クラウドとはこういった小規模発電設備と同じ位置づけの技術なのではないかと考える．エコロジー的側面を考慮すれば，大規模データセンタによるスケール・メリットの追求は最善の解決方法になるとは限らない．無数の小規模クラウドを集約する形態のクラウド・サービスもまたクラウドコンピューティングの 1 形態として成立し得るのではないかと筆者は考えている．

参考文献

1）Dean, J. and Ghemawat, S.：MapReduce, Simplified Data Processing on Large Clusters, Communications of the ACM, Vol.51, No.1, pp.107-113（2008）.
http://labs.google.com/papers/mapreduce.html
2）Ghemawat, S., Gobioff, H. and Leung, S.-T.：The Google File System, Proceedings of the 19th ACM Symposium on Operating Systems Principles, pp.20-43 (2003).
http://research.google.com/archive/gfs-sosp2003.pdf
3）Chang, F., Dean, J., Ghemawat, S., Hsieh, W. C., Wallach, D. A., Burrows, M., Chandra, T., Fikes, A. and Gruber, R. E.：Bigtable：A Distributed Storage System for Structured Data, 7th USENIX Symposium on Operating Systems Design and Implementation (OSDI), pp.205-218 (2006).
http://research.google.com/archive/bigtable-osdi06.pdf
4）Burrows, M.：The Chubby Lock Service for Loosely-coupled Distributed Systems, 7th USENIX Symposium on Operating Systems Design and Implementation (OSDI)(2006).
http://research.google.com/archive/chubby-osdi06.pdf
5）Brin, S. and Page, L.：The Anatomy of a Large-Scale Hypertextual Web Search Engine, Computer Networks, Vol.30, pp.107-117 (1998).
http://infolab.stanford.edu/~backrub/google.html
6）http://hadoop.apache.org/common/docs/r0.19.1/quickstart.html
7）http://wiki.apache.org/hadoop/
8）http://kosmosfs.sourceforge.net/
9）http://hadoop.apache.org/hbase/
10）http://www.hypertable.org/
11）http://hadoop.apache.org/zookeeper/
12）http://open.eucalyptus.com/
13）http://sourceforge.net/apps/mediawiki/kai/index.php?title=Main_Page
14）神林　亮：私がチャレンジした SBM データマイニング：小特集 ソーシャルブックマークは進化し続ける！〜ソーシャルブックマーク研究会の議論から，情報処理，Vol.49 No.12, pp.1421-1423 (Dec. 2008).

（平成 21 年 9 月 15 日受付）

藤田昭人（**正会員**）
akito_fujita@mvg.biglobe.ne.jp
（株）IIJイノベーションインスティテュート研究員．構造化オーバーレイ技術の研究に従事．現在，クラウドコンピューティング関連の事業化を目指している．