大規模分散処理　ビッグデータ　サービスプラットフォーム

Technology Reports

社会・産業の発展を支える「モバイル空間統計」
―モバイルネットワークの統計情報に基づく人口推計技術とその活用―

社会の頭脳システム ―モバイル空間統計を支える大規模データ処理基盤―

ドコモは大規模データの解析により，「モバイル空間統計」に代表される新たなサービスを開発，実行する「社会の頭脳システム」の研究に取り組んでいる．本稿では，大規模データ処理基盤としてのHadoop™ *1 システムの設計，構成，運用，プログラム開発について解説する．本システムは，市販のコモディティIAサーバ1,000台超から構成され，監視，運用にはオープンソースソフトウェアを活用した．さらに，MapReduceによる大規模プログラムを開発し，その実行最適化を行った．

先進技術研究所
石田 創　趙 晩熙
石井 健司　國頭 吾郎
中山 誠　鈴木 亮平
越智 大介　川崎 紀宏
大町 正史
ドコモ・テクノロジ株式会社 マルチメディア事業部
甲本 健

1. まえがき

大規模なデータを解析し，その結果を基に新たなサービスを開発，実行するシステムの研究を行っている．ドコモはこれを「社会の頭脳システム」[1]と呼ぶ．社会の頭脳システムは，携帯電話ネットワークの大規模な運用データを処理することで社会の動きを捉え，その結果を基に新たな領域のサービスを開発，実行することをねらいとしている．社会の頭脳システムで実現した新サービスの好例が「モバイル空間統計」である．

大規模データ処理基盤として，Hadoop[2]が注目されている．Hadoopは，市販のコモディティIAサーバ*2を活用し，大規模データの蓄積機能であるHDFS（Hadoop Distributed File System）™ *3と，大規模データの処理機能であるMapReduceからなる．Hadoopは，サーバのハードウェアの故障を別のサーバで代替し，システム全体でサーバ故障の影響を吸収するミドルウェア*4である．したがって，サーバ故障により利用可能なサーバ台数が減少しても，データの消失やシステム停止などの問題発生を抑止できる．また，MapReduceは，大規模なデータを複数のサーバで並列分散処理することにより処理を高速化するフレームワークであり，利用可能なサーバの台数に応じて柔軟に処理性能を向上できる．これにより，大規模なサーバ群から構成されるサーバクラスタシステムにおいて，多少のサーバ故障を許容し，システムの保守運用の稼動を低減することが可能である．

ドコモは，社会の頭脳システムの大規模データ処理基盤として，1,000台超のサーバで構成された大規模サーバクラスタ上で動作するHadoopシステム（以下，Stevia）を構築し

＊1 **Hadoop™**：Apache Software Foundationの登録商標．
＊2 **IAサーバ**：Intel社製のマイクロプロセッサ，およびIntel互換プロセッサを搭載したサーバ．内部構造はパソコンとほぼ同様であり，他社製プロセッサベースのサーバと比べて安価であることが特徴．
＊3 **HDFS™**：Hadoopで使用する分散ファイルシステム．HDFSはApache Software Foundationの登録商標．
＊4 **ミドルウェア**：OSと実際のアプリケーションの間に位置し，さまざまなアプリケーションに対して共通の機能を提供するソフトウェアのことで，アプリケーション開発の効率化が可能となる．

た．Steviaは，すべて市販のコモディティIAサーバとネットワーク装置から構成され，オープンソースソフトウェアを活用して監視，運用システムを設計することにより，安価かつ短時間での構築を実現した．さらに，Hadoopの利点を活かし，少人数での保守運用を可能にした．また，MapReduceによる大規模データ処理プログラムを開発し，その実行最適化により，大規模サーバクラスタの利用効率向上を実施した．

本稿では，Steviaに関するコモディティIAサーバによるシステムの設計と構築，オープンソースソフトウェアを活用した監視・運用システムと保守運用，および大規模データの分散処理プログラムの実行最適化について解説する．

2. 大規模データ処理基盤の設計と構築

2.1 Hadoop

Hadoopは，オープンソースの大規模データ処理基盤であり，HDFSの分散ファイルシステムとMapReduceによる大規模データ処理プログラムのフレームワークをもつ．HDFSでは，大規模なデータファイルを一定の大きさのブロックに分けて管理し，そのブロックを保持するサーバ上で分散並列に実行されるのが，MapReduceプログラムである．従来のデータベースによる大規模データ処理では，大規模データを蓄積するストレージと，データ処理量に制約があるデータベースエンジンに分離されていた．処理としては，ストレージから少しずつデータを転送して，データベースエンジンで処理し，ストレージに転送するという一連の作業を繰り返す必要がある．Hadoopでは各サーバが自身で保持するブロックを処理することを基本としているため，データの頻繁な転送が発生せずに高速に処理できる．

Hadoopの構成を**図1**に示す．Hadoopのシステムは，マスターノードとスレーブノードからなる．スレーブノードは実際のデータ蓄積やデータ処理を実施するノードで，多数存在する．ブロックの保持を行うデータノードプロセスと，MapReduceプログラムの実行を行うタスクトラッカープロセスが，各スレーブノードで動作する．マスターノードはスレーブノードを管理するノードで，主にネームノードおよびジョブトラッカーから構成される．ネームノードは，スレーブノード上に分散されたブロックの配置を管理する．ジョブトラッカーは，MapReduceプログラムの実行を制御し，タスクトラッカーへのMapReduceプログラムの開始・終了を制御する．これら一連のMapReduceプログラムの実行はジョブと定義される．

1つのブロックは，基本的に異なる複数台のデータノードで複製が保持される．あるデータノードが故障した場合，ネームノードの指示により，該当のデータノード上に存在したブロックの複製が他のデータノードに作成され，自動的に一定数の複製状態を維持する．これにより，複数のスレーブノードが断続的に停止した場合でも，データの損失を防ぐ

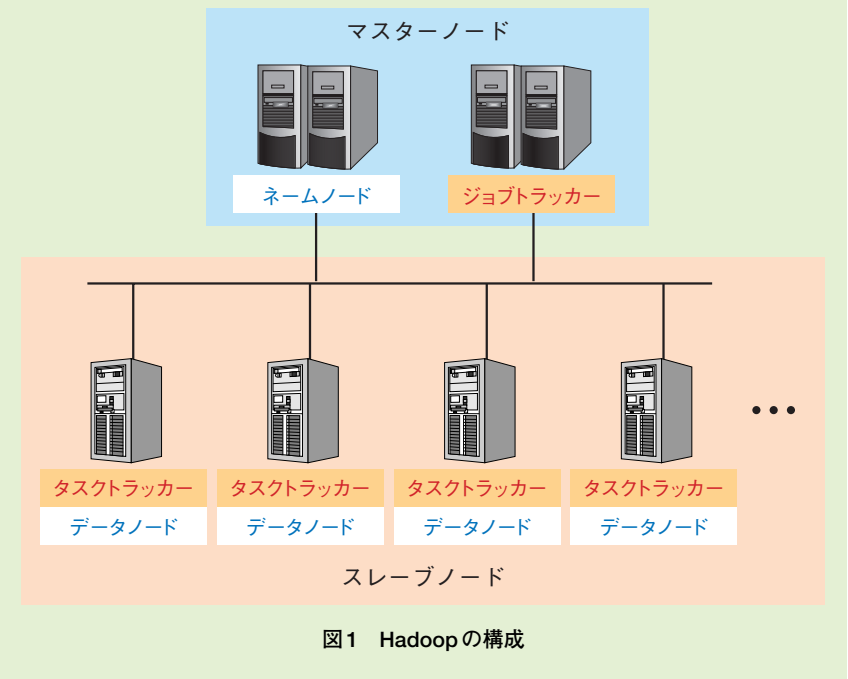

図1　Hadoopの構成

ことができる．また，ジョブトラッカーは，ジョブの処理対象のブロックをもつスレーブノードのタスクトラッカーに，MapReduceプログラムの実行を指示し，その実行状況を監視する．もし指示したタスクトラッカーでの実行が失敗した場合は，他のタスクトラッカーへMapReduceプログラムの実行を再指示する．また，ジョブトラッカーは，複数ジョブが投入された場合の優先順位などを制御する．これらの機能により，スレーブノードが故障し，多少台数が減ったとしても，システムの動作は継続する．

ただし，ネームノードやジョブトラッカーなどのマスターノードは，故障した場合に他のノードで代替することができない単一故障点[*5]であり，高信頼化するために，これらのサーバを二重化構成にする必要がある．

2.2 Steviaの設計と構築

図2にSteviaのサーバ構成を示す．単一故障点であるマスターノード群は，DRBD（Distributed Replicated Block Device）®[*6][3]とHeartbeat[*7][4]を用いて二重化している．DRBDにより，2サーバ間でストレージを完全同期させ，Heartbeatで稼働系故障時に待機系を自動的に稼働させることで高信頼化している．監視・運用システムのサーバとしては，モニタリングサーバ，インストールサーバがある．

大規模サーバクラスタであるSteviaには，多数のサーバを接続するためのネットワークが必要である．**図3**に，Steviaのネットワーク構成を示す．ネットワークは，データ処理に利用する業務系と，システム管理に利用する監視系から構成される．業務系は，1カ所の故障でシステム全体が停止しないように，L2SW（Layer2 Switching hub）[*8]，L3SW[*9]，サーバポートを二重化している．一方，サーバの監視は，OSと遠隔制御ボード[*10]の2系統で行っており，OSレベルでの監視は業務系のネットワーク経由で，遠隔制御ボードでの監視は監視系のネットワーク経由で行っている．OSの起動状態は遠隔制御ボード経由でも見ることができ，遠隔制御ボードの状況はOSにログインして取得すること

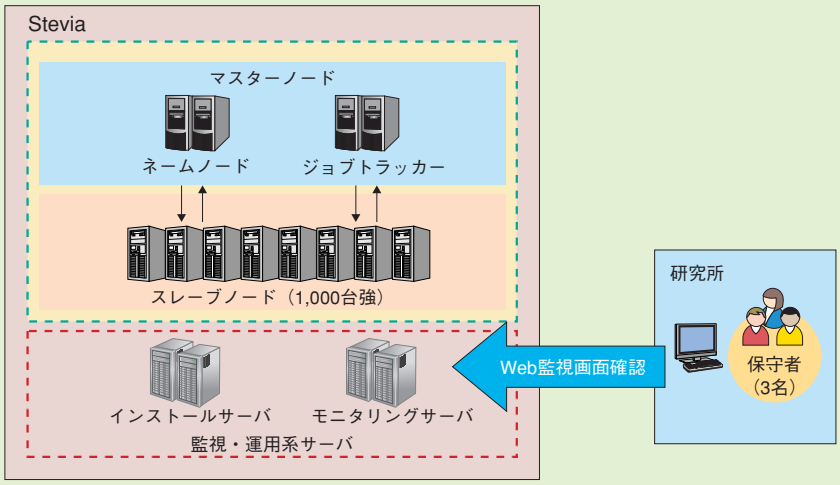

図2　Steviaのサーバ構成

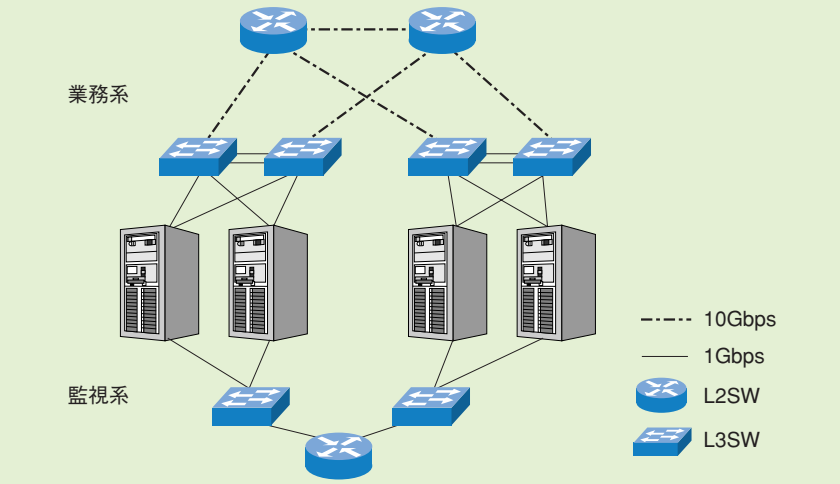

図3　Steviaのネットワーク構成

＊5　**単一故障点**：その箇所が故障するとシステムの全体が停止する箇所．
＊6　**DRBD®**：複数のLinuxサーバ間でディスクパーティションのミラーリングするミドルウェア．DRBDはLINBIT Information Technologies GmbHのオーストラリア，米国およびその他の国々における商標または登録商標．
＊7　**Heartbeat**：高可用性クラスタを構成するためのソフトウェアで，あるサービスを提供するサーバが落ちたときに代替機がそのサービスを引き継ぐための機能を提供する．
＊8　**L2SW**：OSI参照モデルの第2層のデータリンク層のスイッチ．通常はイーサーネットのパケットを転送するスイッチングハブのこと．
＊9　**L3SW**：OSI参照モデルの第3層のネットワーク層のスイッチ．通常はTCP/IPのパケットを転送するルータのこと．

もできるため，監視系のネットワーク自体は一重化とした．各サーバは，ラック単位のL2SWに1Gbpsで接続されている．一方，ラック間は，ラック単位のL2SWと上位のL3SWの間を，10Gbpsで接続している．

Steviaは，ハードウェアの搬入開始から，監視・運用システムも含めて6カ月で構築を完了し，運用を開始した．利用したサーバやネットワーク装置はいずれも一般市販品であり，安価に調達することができた．運用開始当初，SteviaのHDFSの総容量は約1ペタバイトであったが，その後HDFS容量が逼迫したため，スレーブノードのサーバ増設を行った．サーバの増設は，全サーバを停止することなく，故障したサーバを復旧する手順と同じように容易に実行することができた．

3. Steviaの監視・運用

Steviaは，筆者らが常時作業する研究所とは異なる遠隔地に設置している．一方，Steviaとは別に，プログラム開発や試験のために小規模サーバクラスタシステムが必要であり，それらを研究所に設置している．そのため，システムの設置場所の違いを意識することなく，また，サーバ規模に依存しない監視・運用の方式が必要であった．そこで，筆者らはオープンソースソフトウェアを活用し，システム規模に依存しない単一な監視・運用システムを構築した．

Steviaの監視・運用システムは，以下の3つの機能を実現している．

①サーバやネットワーク装置の監視
②サーバやネットワーク装置のリソース（メモリ使用量，CPU使用率など）統計情報の取得
③サーバへのソフトウェアの自動インストール

これらを**表1**に示すオープンソースソフトウェアで実現した．

Nagios®[*11][5]は，サーバやスイッチの死活監視，サーバのログ，プロセス，HTTP[*12]監視，ネットワーク機器などのSNMP（Simple Network Management Protocol）[*13]監視を行い，その状況を保守者に通知する．Nagiosにより Steviaの全装置について網羅的に監視することができたが，通知メッセージ数が多いため，メッセージの選別が必要であった．また，一部のSNMPで警告を発生しない故障などは，実機のランプ確認が必要であった．

Ganglia[*14][6]とCacti[*15][7]は，サーバやネットワーク装置におけるCPU使用率，メモリ使用量，ディスク使用率を取得し，それらを表示する．HDFSの要であるネームノードの動作状況を監視するため，独自モジュールを追加することによりネームノードのJava®[*16]のメモリ使用量やコネクション数を観測できるようにした．

KickStart[*17][8]は，自動インストール機能を提供する．KickStartのレポジトリ[*18]を常に最新の状態に維持することで，スレーブノードの復旧と追加の稼働を削減した．Steviaの場合，スレーブノード1台へのインストールを約18分で完了する．データノードについては，同時に50台のサーバへインストールを実行することができることから，全サーバを一気に変更する場合の作業時間を短縮することができた．

Steviaにおけるハードウェア故障の大多数はスレーブノードで発生しており，Hadoopの動作には影響しないため，緊急の対応は不要であった．故障が発生すると，その状況を解析し，ベンダに修理を依頼する．故障解析とベンダとの連絡業務のために，毎月0.5人月の稼動がかかることが計測により判明した．修理はハードウェア故障の都度実施する必要はなく，まとめて月に1～2度の頻度で実施した．

Steviaの運用開始以来，Hadoopクラスタとして機能を提供した時間をもとにした稼動率は99.5％であり，稼動サーバ数は常に1,000台超を維持してきた．

4. 大規模MapReduceプログラムの最適化

Hadoopでの大規模データ処理は，MapReduceのフレームワークに基

表1　保守運用へのオープンソースソフトウェアの利用

種　別	使用ソフト
監視	Nagios
統計情報取得	Ganglia
	Cacti
自動インストール	KickStart

＊10 **遠隔制御ボード**：OSの状態に関係なく，遠隔地から電源のオン・オフやハードウェアリセット，コンソール画面表示，キーボードやマウスによるコンピュータの操作を可能とする制御ボード．
＊11 **Nagios®**：UNIXコンピュータおよびネットワークサービスの監視のアプリケーション．Nagios®は，Nagios Enterprise が所有するサービスマーク，商標，および登録商標．
＊12 **HTTP**：WebブラウザとWebサーバの間で，HTML（HyperText Markup Language）などのコンテンツの送受信に用いられる通信プロトコル．
＊13 **SNMP**：TCP/IPネットワークにおいて，ルータやコンピュータなど，ネットワークに接続されたネットワーク機器を監視・制御するためのプロトコル．
＊14 **Ganglia**：複数のサーバのCPUやメモリの状態を収集し，Web経由で監視することができるリアルタイム監視ツール．
＊15 **Cacti**：Cactiは Cacti Group, Inc.の登録商標．SNMPのデータをグラフ化するツール．

づくプログラムにより実現される．MapReduceは，MapperとReducerという2つのプログラムにより構成される．Mapperは，各スレーブノードに蓄積されたブロックを読み込み，指定された条件に合致するデータのみをKeyとValueのペアで取り出し，Keyに基づいて指定されるReducerへデータを転送する（シャッフル処理）．各Mapperはこれらの処理を並列して実行できる．そして，ReducerはMapperから転送されたデータを読み込み，Keyごとにそのを Valueをカウントするなどの計算を行う．各Reducerもこれらの処理を並列して実行できる．

例えば，特定の日時に東京都で発生した携帯電話ネットワークの制御信号数をカウントするプログラムを考える．Mapperは特定の日時のすべての信号をチェックし，東京都という条件に合致した信号のみを出力する．さらに，Reducerは，Mapperからの出力をカウントすることで目的を達成する．このとき，1台のスレーブノードのみでMapperを動作させると膨大な処理時間が必要となるが，それを多数のスレーブノードで並列に動作させることにより，短時間で処理を終えることができる．例えば，大きな仕事を1人で実行するよりも，複数人で手分けすることにより，早く完成することができるようなものである．ここでもしMapperが各都道府県別にKeyを仕分けて，都道府県ごとに信号をカウントするReducerにデータを送るプログラムを作れば，47個のReducerの並列実行で，47都道府県別の信号数をカウントすることができる．このようにして，MapReduceプログラムは大規模データを処理する．そして，複数のMapReduceプログラムを直列に繋げて実行することにより，より複雑な処理を実行する大規模プログラムを実現する．

複数のMapReduceプログラムを繋げた大規模プログラムでは，スレーブノード全体で各MapReduceプログラムごとに進捗が同期する．すなわち，**図4**に示すように，あるスレーブノードにおいて，担当したMapReduceプログラムの実行が終了しても，他のスレーブノードで同一のMapReduceプログラムの実行が終了しないと，その出力結果を入力とする次のMapReduceプログラムの実行は開始されない．したがって，大規模プログラムの実行時間をトータルで短くするためには，各MapReduceプログラムの並列度をできるだけ高めて実行時間を短くするとともに，各スレーブノードの実行時間をなるべく均一化する必要がある．しかし，処理するデータ量の違いなどで，スレーブノードごとにプログラムの実行時間が大きく違うことは容易に起こる．例えば，前述のように，都道府県別の集計では，keyを都道府県にすると，東京都で発生する信号が非常に多いため，東京都の集計を担当したReducerの実行時間が非常に長くなる．このような場合，より規模の小さな市区町村の単位で集計させた後に都道府県単位で再集計させる2段階処理などの工夫をすることで，全体の処理時間を短縮することが可能である．

前述のように，MapReduceによ

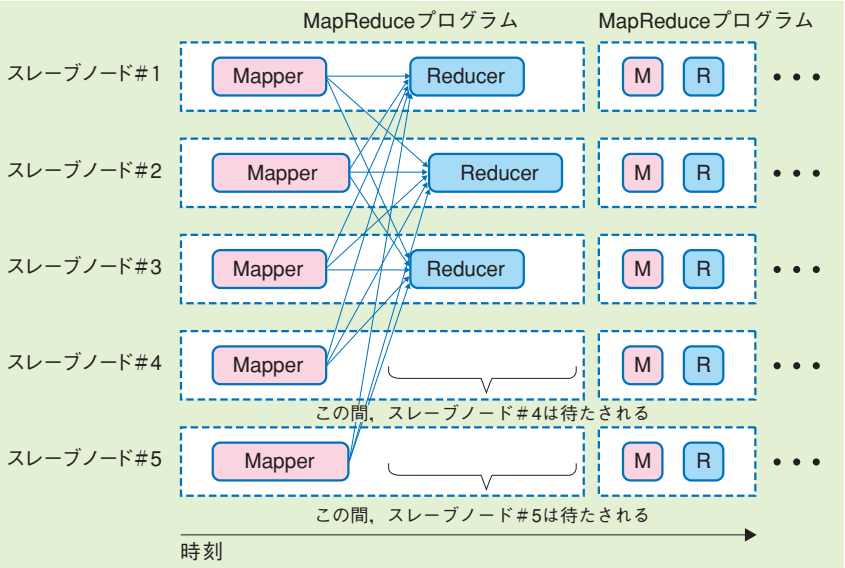

図4　大規模MapReduceプログラムの実行例

＊16 **Java**®：オブジェクト指向のプログラミング言語．Javaにより実装されたアプリケーションは仮想マシン上で実行されるため，異なるプラットフォーム上で動作可能である．OracleとJavaはOracle Corporation およびその子会社，関連会社の米国およびその他の国における登録商標．文中の社名，商品名などは各社の商標または登録商標である場合がある．

＊17 **KickStart**：RedHat系のLinuxでのOS自動インストールの仕組み．

＊18 **レポジトリ**：アプリケーションやシステムの設定情報をまとめて記録するシステム．

る大規模プログラムの開発においては，できるだけ多くのスレーブノードで処理を分担させるとともに，特定のサーバに負荷を集中させないような実行最適化が必須である．しかし，MapReduceによる大規模プログラムの実行最適化は手作業で行う必要があり，その作業自体が難しい．また，スレーブノードの処理を小さくしすぎると，通信のオーバーヘッドが増えたり，keyが分かりにくくなるという別の問題が顕在化する．ドコモは，このような大規模MapReduceプログラム開発における経験を踏まえ，MapReduceプログラムの実行状態を全サーバについてモニタし，実行後に可視化するツール[9]の開発や，実行最適化を自動化[10]する研究を行ってきた．実行最適化のさらなる工夫は，今後も重要な研究テーマの1つである．

5. 今後の課題

Stevia構築当初，大規模データの計算結果のサイズは非常に小さいと見込んでおり，必要十分なHDFS容量を用意したと考えていた．ところが，プログラムの種類が増え，それぞれの計算結果の規模も予想以上に大きいことが判明し，HDFS容量が逼迫する事態が発生した．このため，サーバの増設による容量増加を行ったが，その際にネームノードのスケーラビリティの課題に直面した．このような爆発的なデータ増加に対して，柔軟に対応できるための方策および，生成データの削除ルールが必須である．

Hadoopは基本的にバッチ処理システムであり，大規模なデータを一定間隔で解析することに強みを発揮する．一方，直近のデータをすぐ解析し，その結果をリアルタイムに見たいという要求も多い．このような要求を満足するための1つの方向は，ストリーム型のデータ処理を行うことである．今後は，ストリーム型データ処理方式の検討も加え，新たな「社会の頭脳システム」の構成について検討していく．

ビッグデータ活用の課題の1つは，ビッグデータを集めてくることである．いくらデータがあっても，それらを解析できるように一カ所に集めることができなければ価値はない．しかも，日々発生するデータを発生源から損失なく集めてくることには多くの維持管理の労力が必要である．社会の頭脳システムにおいても，データを継続して自動的に収集するシステムの設計，構築に多くの時間を要してきた．今後は，データの扱いには十分な注意をしながら，コストパフォーマンス良くデータを収集する仕組みの実現に取り組む．

6. あとがき

社会の頭脳システムにおける，大規模データ処理基盤であるSteviaについて解説した．「モバイル空間統計」の作成を支えるSteviaは，ペタバイトクラスの大規模システムであるが，Hadoopの特長を活かして，少人数での維持管理体制のもとで，安定的に運用を継続することができた．今後は，エクサバイトクラスのより大規模なデータをハンドリングできるデータ分析基盤を検討し，携帯電話ネットワークを活用した新たなサービスの実現をめざす．

文　献

[1] 堀越 功：“NTTドコモが巨大マイニング設備構築，” 日経コミュニケーション, pp.30-31, Oct. 2009.

[2] T. White著, 玉川 竜司, 兼田 聖士訳：“Hadoop，” オライリー・ジャパン, Jan. 2010.

[3] DRBD：“DRBD.jp by Thirdware inc.” http://www.drbd.jp

[4] Heartbeat：“Heartbeat - Linux-HA.” http://www.linux-ha.org/wiki/Heartbeat

[5] Nagios：“Nagios - The Industry Standard in IT Infrastructure Monitoring.” http://www.Nagios.org

[6] Ganglia monitoring system: “Ganglia Monitoring System.” http://Ganglia.sourceforge.net

[7] Cacti：“Cacti® - The Complete RRDTool-based Graphing Solution.” http://www.Cacti.net/index.php

[8] KickStart：“Part VI. Advanced Installation and Deployment.” http://www.centos.org/docs/5/html/5.2/Installation_Guide/pt-install-advanced-deployment.html

[9] 川崎 紀宏, 田中 聡, 岡島 一郎：“大規模分散システムにおける並列プログラムの通信と処理状況を同時に表示する可視化ツール，” 信学技報, Vol.110, No.448, NS2010-257, pp.527-532, Feb. 2011.

[10] 中山 誠, 山崎 憲一, 田中 聡：“MapReduceプログラミングモデルに即したReduceフェーズのDataSkew動的緩和方法，” 情処学論, Vol.53, No.3, pp.1189-1203, Mar. 2012.