オペレーションシステム経済化技術
―分散データ駆動型アーキテクチャ―

移動通信ネットワークは多種多様で，数多くのネットワーク設備から構成される．これらのネットワーク設備を監視制御するシステムはOPSと呼ばれている．従来のOPSは，高価な大規模UNIXサーバと市販ミドルウェアを用いて構成されているため，そのTCOの増大を招いていた．そこで，OPSのTCOの大幅な削減を可能とする，分散データ駆動型アーキテクチャ“D3A”を考案した．

秋山 一宜（あきやま かずよし）　昆 孝志（こん たかし）
高橋 和秀（たかはし かずひで）　神宮司 誠（じんぐうじ まこと）

1. まえがき

OPS（OPeration System）[1][2]は，移動通信ネットワークで発生する故障や輻輳の発見と，それに対する回復措置を担っている重要なシステムである．OPSを構成するサーバの故障や内部輻輳などが要因で，ネットワーク設備（NE：Network Element）からの警報がネットワーク管理者に通知されなかった場合，NEの故障や輻輳に対する措置が遅れ，通信サービスの停止やサービス品質の低下が長期化する．このため，OPSには高い信頼性と高い処理能力が要求される．これらの要求条件を満たすために，従来のOPSは，大規模UNIXサーバとRDBMS（Relational Data Base Management System）やCORBA（Common Object Request Broker Architecture）などの市販ミドルウェアを多用して構築されてきた．

しかし，大規模UNIXサーバと市販ミドルウェアの導入コストや保守コストは非常に高額である．また，大規模UNIXサーバを提供するハードウェアベンダの都合（サーバの販売停止や保守停止など）に依存せざるを得ない状態（ベンダロックイン）に陥り，後継機へのサーバリプレースやOSのバージョンアップなどを余儀なくされてきた．その結果，いったん特定のハードウェアベンダのサーバを採用すると，後継機の導入コストだけでなく，OPSアプリケーションの動作検証コストについても高額な投資を伴うこととなる．市販ミドルウェアについても，その販売停止や保守停止により，ハードウェアと同様なベンダロックイン状態に陥る．

一方で，近年サーバ構成技術の動向は大きく変化しており，小規模IA（Intel Architecture）サーバやギガビットイーサネット（GbE：Gigabit Ethernet）の低価格化が進み，市場流通性が増したことから，PCクラスタ[3]，グリッドコンピューティング[4][5]，モバイルエージェント[6][7]など複数のサーバを結合して高い処理能力を得る技術が実用化段階を向かえている．これらの技術分野は，ハイパフォーマンスコンピューティング（HPC：High Performance Computing）と呼ばれている．また，Linuxに代表されるオープンソースソフトウェア（OSS：Open Source Software）が安定化し，重要なシステムにも採用される例が多くなってきた．

本稿では，これらの技術動向を踏まえ，OPSのTCO（Total Costs of Ownership）を大幅に削減することを目的として考案し，実用化した分散データ駆動型アーキテクチャ（D3A：Distributed Data Driven Architecture）[8]～[17]について，その動作メカニズムと処理能力の測定結果について述べる．また，その評価についても従来のOPSと比較しながら述べる．

2. D3Aの基本動作

D3Aは，複数のIAサーバを束ねて高い処理能力を得ることが可能なアーキテクチャである．また，その技術的要素からハードウェアの拡張性やアプリケーション追加の柔軟性に優れ，従来のシステムアーキテクチャと比較して，コスト面においても優位な新技術といえる．

2.1 動作メカニズム

本アーキテクチャを**図1**に示す．本アーキテクチャは，以下に示す7つの要素から構成される．

- 物理エレメント（PE：Physical Element）
- 論理エレメント（LE：Logical Element）
- 論理経路データ（LPD：Logical Path Data）
- LPDドライバ（LPDD：LPD Driver）
- LPDマネージャ（LPDM：LPD Manager）
- 外部アダプタ（EA：External Adaptor）
- エレメント構成マネージャ（ECM：Element Configuration Manager）

ハードウェアの構成要素がPEであるのに対し，ソフトウェア（OPSアプリケーション）の構成要素がLEである．PEは，ラックマウント型IAサーバについては小規模IAサーバそのものに相当し，ブレード型IAサーバについてはエンクロージャーに搭載された各サーバブレードに相当する．ここで，小規模IAサーバとは2CPU（Central Processing Unit）以下のIAサーバとする．LEはOPSアプリケーションを分割した並列実行単位であり，PEに分散配置される．LEは機能分散の単位，つまり分散配置が可能な機能の最小単位といえる．PEは負荷分散の単位，つまりハードウェアの限られたリソースに対して，負荷処理を分散する単位ということもできる．これら複数のLEを組み合わせることにより，1つのOPSアプリケーションを構成する本ア

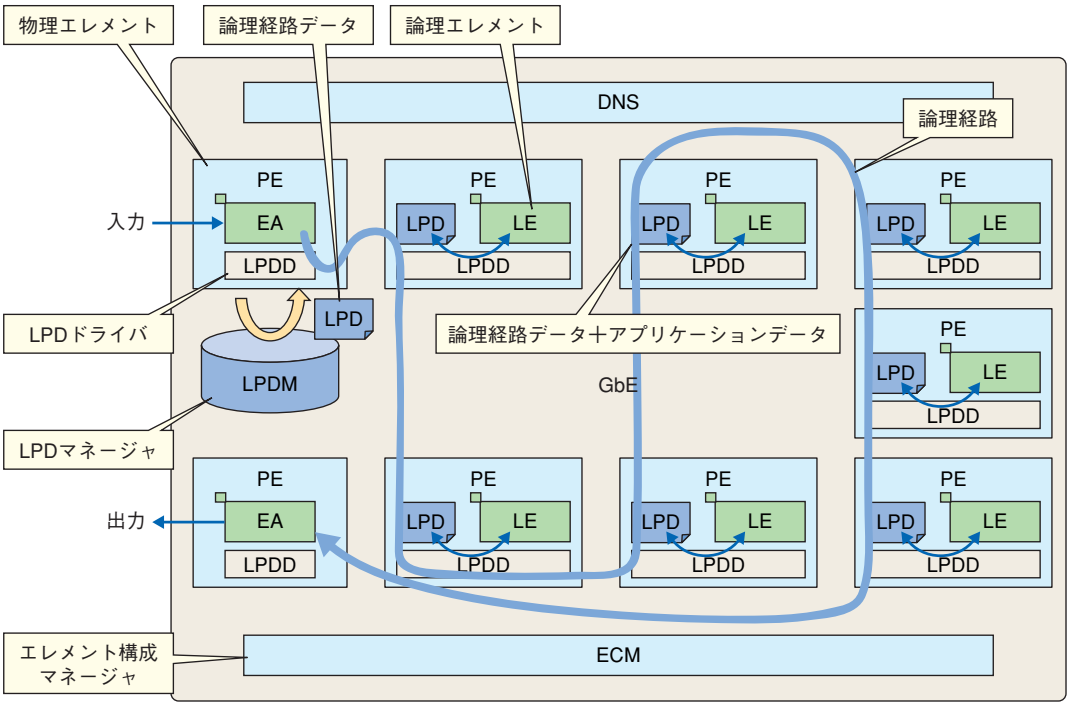

図1　D3A

ーキテクチャを使用すれば，高い処理能力を得ることが可能である．また，動作環境としては，3GHz以上の処理速度のCPUを搭載した非常に高性能なPEが，高速なGbEで結合されるため，LE間は高速な通信が可能である．

LPDに基づく論理経路制御の様子を**図2**に示す．LPDは，論理経路データ部とアプリケーションデータ部の2つの部分から構成され，XML（eXtensible Markup Language）により記述される．各LEは，前段のLEからLPDを受信すると，自分が処理すべきアプリケーションデータのみを，LPDから抽出して処理し，その処理結果をアプリケーションデータに再び設定して，次段のLEに送信する．LPDの論理経路に指定された各LEにおいて必要となるデータ（LE間インタフェース）は，アプリケーションデータ部にて定義されるため，各LPDのアプリケーションデータ部は，その論理経路におけるLE間の共通インタフェースであると見なすことも可能である．

LPDDによる物理経路制御の様子を**図3**に示す．LPDDは，LPDの論理経路データ部に基づいて，LE間で論理的に経路制御を行うドライバである．また，LPDDはドメインネームシステム（DNS：Domain Name System）に基づい

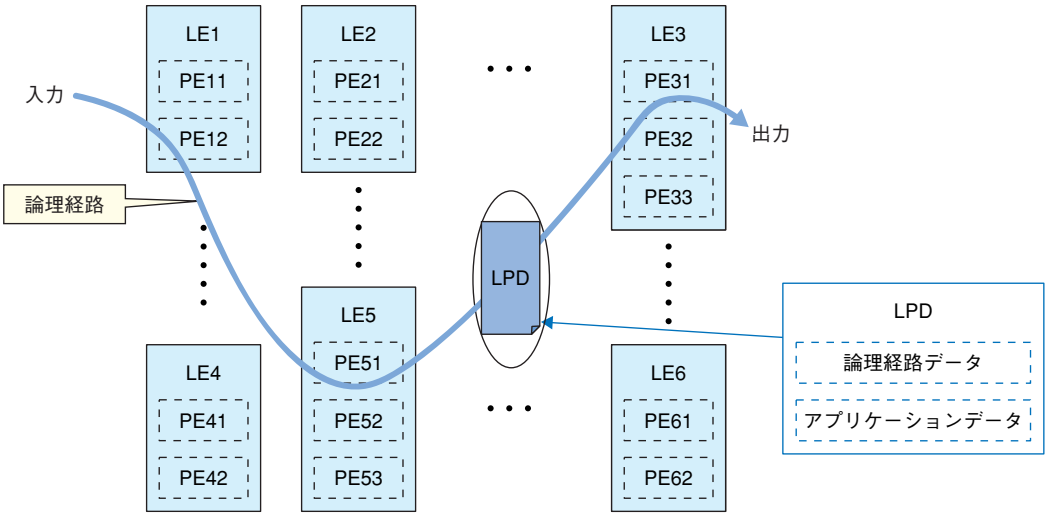

図2　論理経路制御

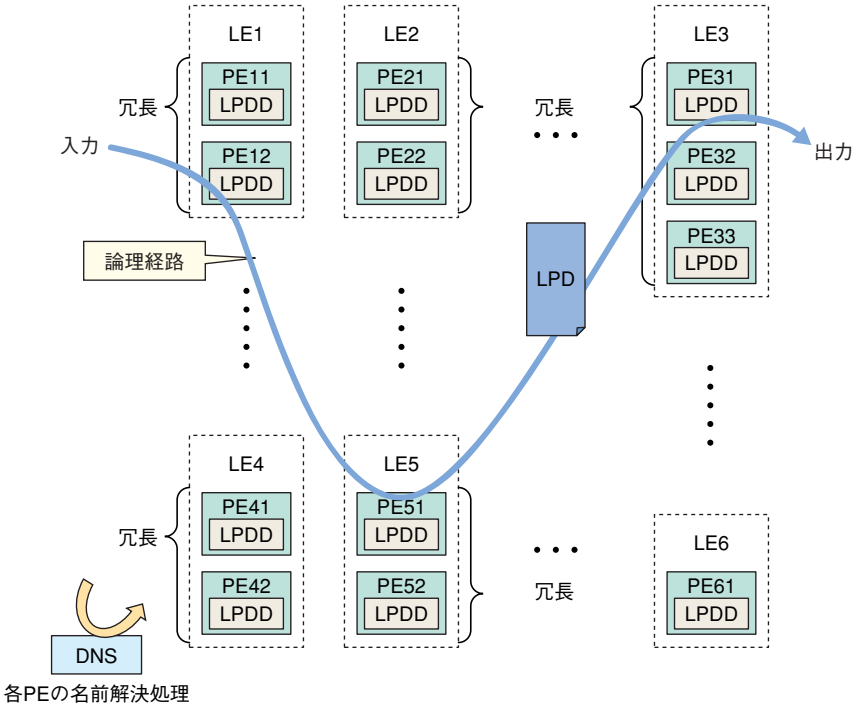

図3　物理経路制御

て，PE間で物理的に経路制御を行う．ここでDNSは，LEの存在するPEの名前解決を行うために用いる．LPDDは，冗長構成を制御する機能やLE/PEの故障を管理する機能などから構成され，本アーキテクチャのプラットフォームとなっている．各PEに分散配置されたLPDDがLPDに基づいてLEを駆動することにより，そのLPDに対応するOPSアプリケーションが実行される．OPSアプリケーションからはPEを意識することがないため，LPDDがPEをLEに仮想化しているといえる．

LPDMはLPDの原本を管理している．また，EAは本アーキテクチャの外縁に配置され，外部とのインタフェースの違いを吸収し，共通インタフェースに変換する機能を有する．外部からの入力を契機に，EAに配置されたLPDDがLPDMの制御により，LPDを構成する論理経路データ部へ，あるテンプレートを読み出し，そのテンプレート内にアプリケーションデータ部を生成する．LPDDによるLPDの読出しを高速化するために，LPDは各PEのLPDDにキャッシュされているが，その原本はLPDMで管理されている．

ECMは，XMLにより記述されたエレメント構成情報ファイル（ECIF：Element Configuration Information File）の原本を管理している．ECIFには，LEとPEの対応関係やLEの冗長構成方式などの構成情報が記述されている．各LEは，初期起動時にECMからECIFを取得し，故障処理時などに参照する．この読出し処理を高速化するために，ECIFは各LEに配備されたLPDDにキャッシュされている．PEの増設や撤去によりECIFが変更された場合，ECMはすべてのLPDDに対して新しいECIFを通知する．ECMは，ECIFに記述されたすべてのLEに対してLPDの送受信を用いてヘルスチェックを行い，未応答のPEを検出するとDNSから該当のエントリを削除することにより，LPDDが故障したPEへ経路制御しないように動作する．

2.2 LPDの経路制御

LPDには論理的に経路制御するためのタグとして，以下の6つの定義があり，基本的にW3C（World Wide Web Consortium）[18]のBPEL4WS（Business Process Execution Language for Web Services）[19]に準拠している．

・直列：<sequence>
・条件分岐：<switch>
・並列分岐：<flow>
・本流と分断された分岐：<drop>
・繰り返し：<for>
・同一のLE（異なるPE）に対する並列分岐：<invokeAll>

また，経路制御の通信方法には非同期型と同期型があり，実現するアプリケーションの内容により，LPDのタグで指定することが可能である．これらのタグ定義のうち代表的な<sequence>タグと<flow>タグについて説明する．

図4は，<sequence>タグによりLE1からLE4までLPDを直列に経路制御していることを示している．非同期型の場合，LPDを受信したLEは，直ちに前段のLEに応答を返送し，自分がLPDを送信した次段のLEからの応答を待ち合わせない．一方，同期型の場合，LPDを受信したLEは次段のLEにLPDを送信し，次段のLEからの応答を待ち合わせて前段のLEに応答を返送する．

図5は，<flow>タグによりLE2からLE3とLE4に並列に分岐するように経路制御していることを示している．非同期型の場合，LE2から分岐したLE3とLE4からの応答を待ち合わせることなくLE1に応答を返送する．同期型の場合，

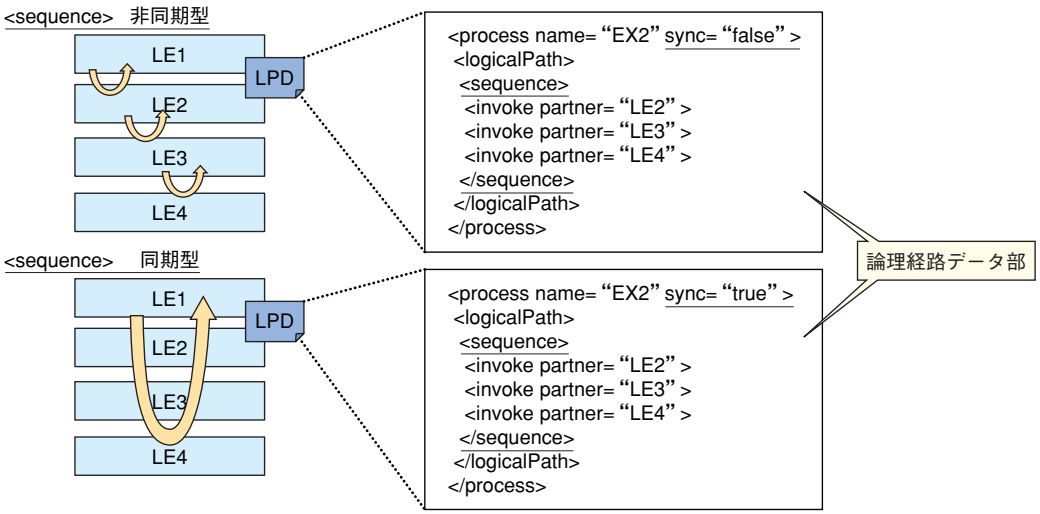

図4　<sequence>タグによる論理経路制御

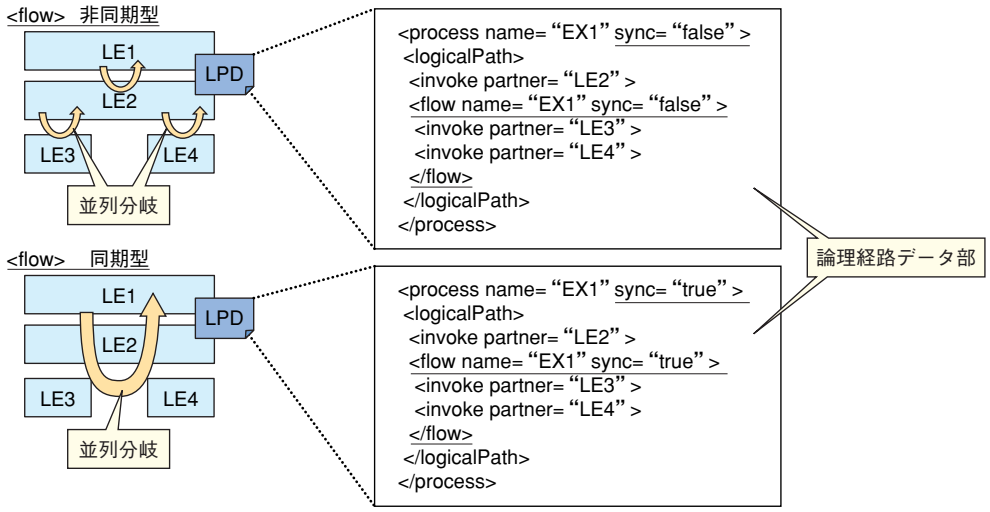

図5　<flow>タグによる論理経路制御

LE2はLE3とLE4からの応答を待ち合わせ，LE1に応答を返送する．

3. D3Aの冗長構成

LEの信頼性を確保するために，PEの冗長構成と負荷分散を実現する必要がある．PEの冗長構成としては，以下の3つの方式がある．それぞれの冗長構成方式を**図6～8**に示す．

・n-ACT（ACTive）構成（選択実行型）
・n-ACT構成（並列実行型）
・n-ACT/m-SBY（StandBY）構成

3.1 n-ACT構成（選択実行型）

本方式では，LPDを受信(①)したPEのLPDDが，ACT状態にあるn個のPEに対して，ラウンドロビン方式で1つのPEを選択し，そのPEにLPDを送信する(③)．この際LPDDは，DNSに対して，LPDにて指定されたLE名により，そのLEが存在するPEのIP（Internet Protocol）アドレスを問い合わせる(②)．また，ECMがECIFに基づいて，PEに対して一定周期でヘルスチェックすることにより，PEの故障を監視する(④)．ECMがPEの故障を検出する(⑤)と，DNSのエントリから，そのPEのIPアドレスを削除する(⑥)．本方式により，PEの負荷分散を実現するとともに，故障したPEを系から切り離すことが可能となる．

3.2 n-ACT構成（並列実行型）

本方式では，LPDを受信(①)したPEのLPDDにて並列指定されているACT状態にあるn個のPEすべてに対して，LPDを送信する(③)．LPDDは，DNSに対して，LPDにて指定されたLE名により，そのLEが存在するすべてのPEのIPアドレスを問い合わせる(②)．また，n-ACT構成（選択実行型）と同様に，ECMがECIFに基づいて，PEに対して一定周期でヘルスチェックすることにより，PEの故障を監視する(④)．ECMがPEの故障を検出する(⑤)と，DNSのエントリから，そのPEのIPアドレスを削除する(⑥)．本方式により，故障したPEを系から切り離すことが可能となる．また，本方式を適用すると，LEはデータをPEに分散配置することで負荷分散を実現する．

3.3 n-ACT/m-SBY構成

本方式は，ACT状態にあるn個のPEとSBY状態にあるm個のPEから構成される冗長方式である．SBY状態にあるPEには通番が付与されており，このうちの最も若い通番を持つPE（代表PE）が，ACT状態にあるすべてのPEに対して一定周期でヘルスチェックをすることにより，PEの故障を監視する(①)．SBY状態にあるPE（代表PE以外）は，自分自身の通番より1つ若い通番を持つPEに対して一定周期でヘルスチェックすることにより，SBY状態にあるPEの故障を監視する(②)．代表PEがACT状態にあるPEの故障を検出すると(③)，ECMに故障したPEを通知(④)し，DNSのエントリから，そのIPアドレスを削除するとともに，自分自身のIPアドレスをACT状態にあるPEとしてDNSのエントリに登録し，故障したPEに成り代わる(⑤)．SBY状態にあるPE（代表PE以外）がSBY状態にあるPE（代表PE）の故障を検出すると，DNSのエントリから，そのIPアドレスを削除するとともに，自分自身のIPアドレスをSBY状態にあるPE（代表PE）としてDNSのエントリに登録し，故障したPEに成り代わる(⑥)．本方式により，故障したPEを系から切り離すことが可能となる．また，本構成を適用すると，LEはデータをPEに分散配置することにより負荷分散を実現する．

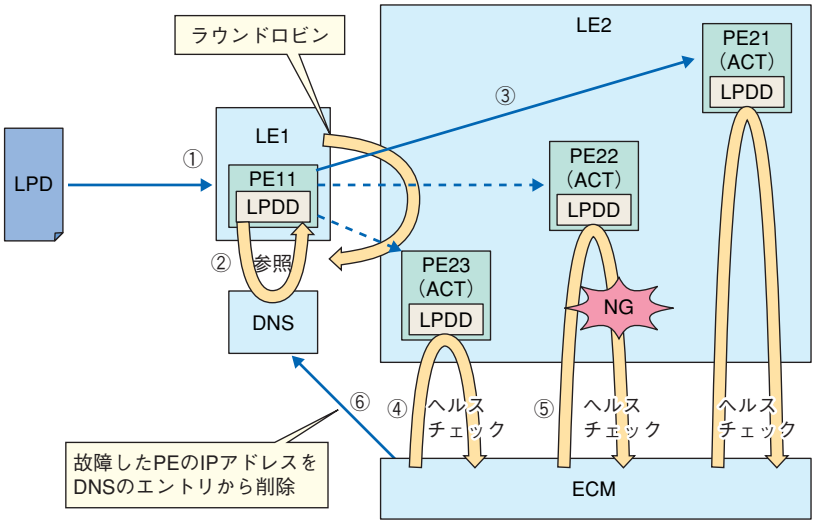

図6　n-ACT構成（選択実行型）

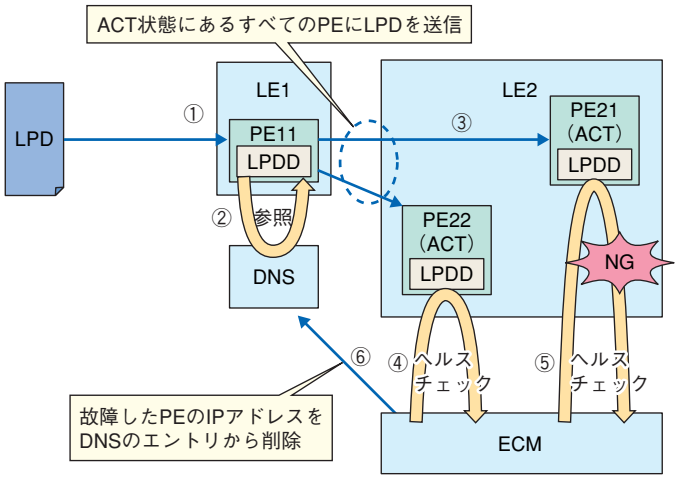

図7　n-ACT構成（並列実行型）

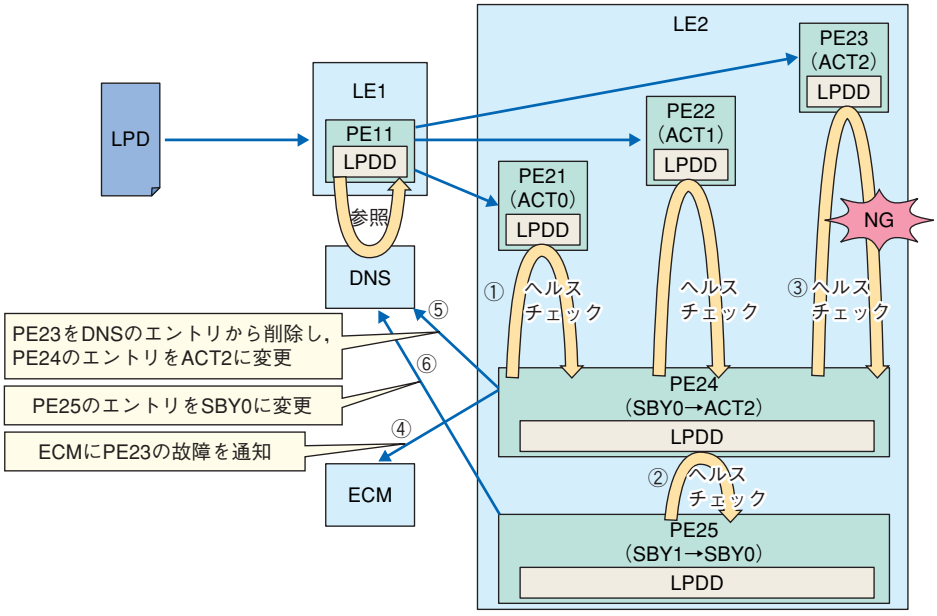

図8　n-ACT/m-SBY構成

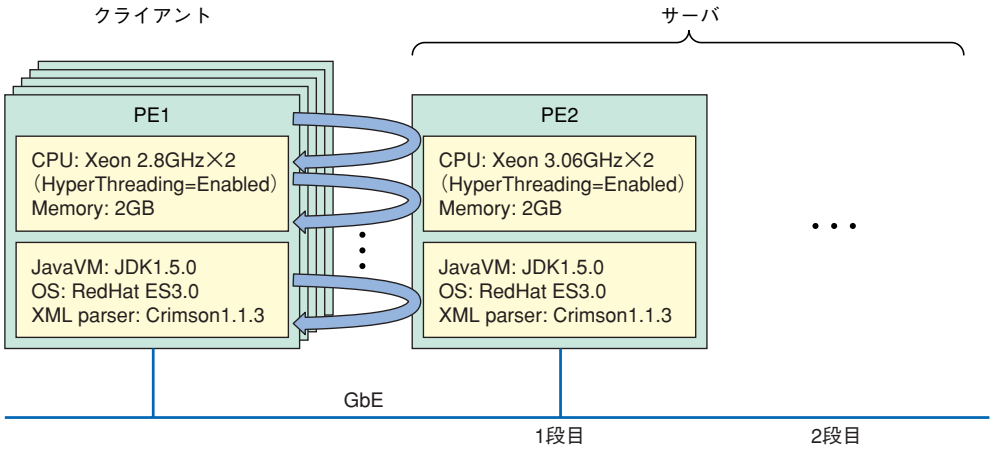

図9　測定環境と測定方法

なお，DNSはプライマリ／セカンダリ構成で動作させている．

4. D3Aの基本処理能力の測定

本アーキテクチャにおいては，LPDD内でLPDのXMLパース処理とLE間通信処理が頻繁に発生するため，その処理能力を明確にしておく必要がある．ここでは，D3Aの基本動作に関してタグ定義ごとにLPDDの処理能力を測定する．

4.1 測定環境と測定方法

測定環境と測定方法を**図9**に示す．ここでは，LPDの送信元のLEをクライアントとし，LPDの送信先のLEをサーバと呼ぶこととする．複数のクライアントから1台のサーバに対して，並列にLPDを送信した場合の処理能力を測定する．クライアントからサーバに対してLPDを合計1000回送信し，クライアントが受信する単位時間当りの応答の平均値mps（message per second）を求めた．ただし，JavaVM（Virtual Machine）によるJava[*1]クラスのロード時間が実験結果に影響しないように1回目の結果は除外した．また，LPDのデータサイズを10kBとした．なお，図9に示すとおり，今回の測定環境は3章で述べた冗長構成を採用していない．

4.2 測定結果と考察

測定結果を**図10**に示す．<sequence>タグによる経路制御においては，LEの段数が1段の場合，非同期型で約1800mpsもの処理能力がある．また，本タグについては，非同期型通信でも同期型通信でも，LEの段数が多くなるにつれてその処理能力が低くなるが，約500mps～600mpsに収れんする．<flow>タグによる経路制御においては，非同期型通信の場合，約200mps～300mps程度，同期型通信の場合，約150mps～200mps程度の処理能力であった．<flow>タグ同期型通信については，並列分岐において次段のLEの処理を待ち合わせるため，並列分岐数が多くなると性能劣化が大きくなると考えられる．

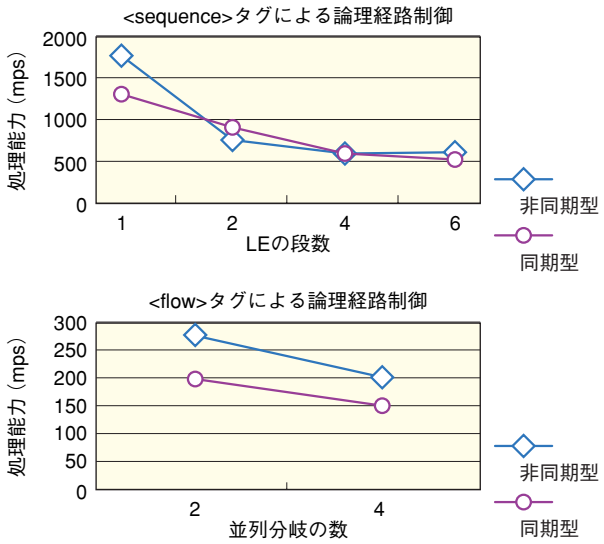

図10　測定結果

5. D3Aシステムの利点

5.1 コスト

OPSを構成するハードウェア／ミドルウェア製品やアプリケーションの構造により，一概に比較することはできないが，OPSの重要な性能指標値である警報受信能力（1秒間に受信可能な警報400件）に基づき評価した．所要の処理能力を達成するために必要なTCOについて，従来のOPSと本アーキテクチャを用いて構成したOPSを比較した結果を**図11**に示す．この比較には，ハードウェアの導入費と保

＊1　Java：米Sun Microsystems社が提唱しているネットワークに特化したオブジェクト指向型開発環境．

守費，およびミドルウェアの導入費と保守費を含んでいるが，アプリケーションの開発費は含まれていない．保守費については5年間で計算している．また，従来のOPSのTCOを100％として正規化している．図11に示すとおり，TCOを約80％削減できる見通しを得た．

5.2 ベンダロックインからの脱却

本アーキテクチャは，低価格な小規模IAサーバとOSSを積極的に活用することを前提としている．また，アプリケーション（LE/EA）はJavaVM上に構築しており，アプリケーションからLinuxのシステムコールやライブラリを直接に使用することを禁止している．これにより，IAサーバに搭載されたCPUのマシン語との互換性にかかわらず，どの機種のIAサーバ上でも動作可能となる．さらに，LE間の通信に用いるLPDをXMLにより記述するため，LE間の通信はハードウェアやOSに依存しない．したがって，PEの異機種結合が非常に容易となり，本アーキテクチャに基づいて開発されたOPSアプリケーションは，異機種混在環境での動作が可能である．

これまでに，異なるCPUを搭載した，異なるハードウェアベンダのIAサーバ10機種について，本アーキテクチャのプラットフォームであるLPDDの動作検証を実施した結果，問題となる現象は検出されなかった[13]．

また，本システムでは市販ミドルウェアを一切採用していないことから，ミドルウェアベンダの戦略に束縛されることはない．基本的にOSSのバージョンアップに追従する必要はないが，ハードウェアのリプレースを契機にJavaVMのバージョンアップを伴うケースが想定される．このような場合でも，JavaVMにはアプリケーションに対する上位互換性があるため，基本的に下位バージョンのJavaVMにて開発されたアプリケーションは，上位バージョンのJavaVM上で動作可能といえる．つまり，JavaVMに上位互換性があるため，ハードウェアベンダに束縛されることはない．本システムにおいて，唯一採用しているLinux上のOSSとして，DBMS（DataBase Management System）であるPostgreSQL[*2]があるが，データベースへのアクセスをJDBC（Java DataBase Connectivity）に制限していることから，ハードウェアに依存しない．

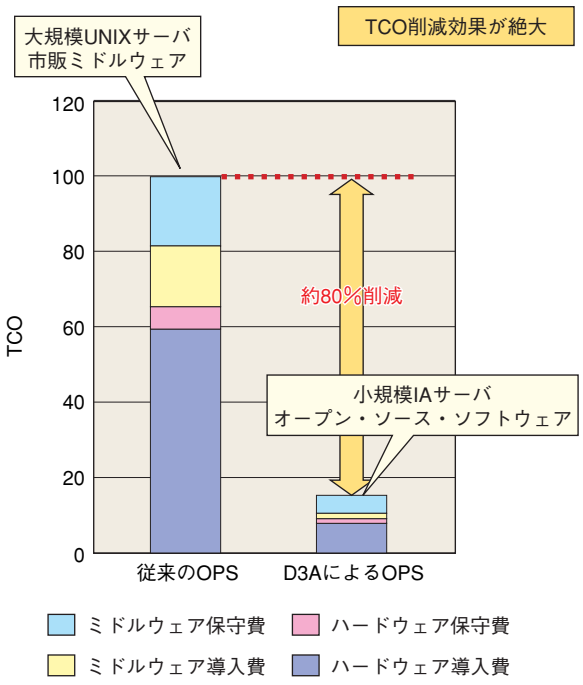

図11　TCO削減効果

5.3 ハードウェアの拡張性

従来のOPSは，大規模UNIXサーバから構成されているため，一部のハードウェアリソース（例えば，CPU）が最大実装されている場合，他のリソース（例えばメモリやディスク）を増設可能であるにもかかわらず，そのリソース（ここではCPU）をそれ以上増設できないために，サーバそのものを増設しなければならないケースが散見される．このような場合，サーバそのものの導入だけでなく，市販ミドルウェアの導入も必要となるため，多大な投資を強いられる．一方，本アーキテクチャでは，特定のLEの処理能力不足に対して，そのLEを配備したPE（小規模IAサーバ）のみを増設するだけで対応可能である．つまり，大規模UNIXサーバと比較して，小規模IAサーバは低価格であり，きめ細かな増設が可能である．

また，小規模IAサーバを増設する際，異なるベンダのIAサーバを増設することも可能である．小規模IAサーバはPCをベースとしているため，市場流通性が高く，小規模IAサーバを販売するハードウェアベンダの数は多い．つまり，増設する際に最も低価格のハードウェアを選択することが可能である．このように，増設するハードウェアの選択肢が広いことは重要なポイントである．

5.4 アプリケーション機能追加の柔軟性

従来のOPSアプリケーションは，CやC++などのコンパイル言語で実装されているため，新たな機能追加を行う場合，基本設計／機能設計／詳細設計の各設計工程，ソースプログラム編集／コンパイル／リンク／データ編集の製造工程，単体試験／結合試験／対向試験／総合試験の各試験

＊2　PostgreSQL：カリフォルニア大学バークレー校で開発されたデータベースシステム．

工程がすべて必要となる．このように，すべての開発工程を踏まなければならないため，機能追加には高額な投資が必要となる．文献[20][21]において提案されている方法でも，コマンド仕様に関する軽微な変更に2週間程度を要する．一方，本アーキテクチャでは，LEの処理順序や処理条件については，LPDにおいてXMLによりデータレベルで記述される．これにより，LPDの変更のみで機能追加が可能な場合は，設計工程を基本設計のみに，製造工程をデータ編集のみに，試験工程を結合試験からプログラム試験のみに短縮することが可能であると考えている．したがって，このような場合は迅速かつ柔軟に機能追加が行える．

5.5 システムの信頼性

従来のOPSでは，特定のハードウェアの故障により，すべての業務が数分程度中断し，その故障が影響を与える範囲はOPS全体となっていた[22][23]．3章で述べた冗長構成により，故障による業務の中断時間は最大のLEでも約90秒程度であり[12]，ハードウェアの故障が影響を与える範囲は，そのLEにのみ限定される．

また，従来のOPSでは，OPSアプリケーションに対する機能追加や変更を行う場合，ロードモジュールの入替えが必要になり，すべての業務が数分程度中断する．本システムでは，LPDとLEを入れ替えるのではなく，新しく追加することにより，業務を中断することなくOPSアプリケーションを変更することが可能である．

6. 関連研究との比較

6.1 グリッドコンピューティング

地理的に分散配置された多数の計算資源を連動させて，スーパーコンピュータを凌ぐハイパフォーマンスを得る技術にグリッドコンピューティングがある．グリッドコンピューティングを医療や物理学の分野で実用化した報告[4][5]がいくつかあるが，商用の大規模なネットワーク管理において実用化された例は見当たらない．最近の技術動向としては，Webサービスとグリッドコンピューティングの技術を融合したモデルとして，OGSA（Open Grid Service Architecture）[24]の標準化活動が盛んである．米アルゴンヌ大学のGlobusプロジェクト[25]は，OGSAを実装したGT3（Globus Toolkit3）[26]を2003年7月に公開した．GT3の実装においては，SOAP（Simple Object Access Protocol）による通信プロトコル処理にApache Axisを用いており，これに大きく依存した実装となっているようである．文献[8]によれば，HTTP/SOAP（Apache Axis）とJavaRMI（Remote Method Invocation）の性能評価から，HTTP/SOAPはJavaRMIに比較し，応答時間については約20倍，処理能力は1/2である．GT3の実装にも，Apache Axisを用いているため，OPSが要求する性能を満たさない．特にSOAPのシリアライズ処理が重いため，条件分岐の数（つまり，シリアライズ処理の数）に比例して処理能力が低下する．OPSを実現するためには2つ以上の分岐処理は必須であり，これらの要求する性能を満たさない．

6.2 モバイルエージェントとの比較

モバイルエージェントをネットワーク管理に適用した報告[6][7]がいくつかあるが，グリッドコンピューティングと同様に，通信キャリアレベルの商用・大規模なネットワーク管理において実用化された例は見当たらない．文献[6][7]においては，NEにモバイルエージェントを送り込んでNEを監視制御する．本稿では，このようなモデルをNE依存型エージェントモデルと呼ぶことにする．NE依存型モバイルエージェントモデルに基づいたネットワーク管理の主な優位点として，NEが警報を多発しない場合，OPS～NE間（つまり，マネージャシステムとエージェントシステム間）の通信トラフィックの削減，およびOPS（つまり，マネージャシステム）負荷の軽減が挙げられている．OPSに，本稿で提案しているアーキテクチャを適用することの優位点について，NE依存型モバイルエージェントモデルとの比較において議論する．まず，第1に，警報を多発するNEが現実に存在しており，このようなNEについて，モバイルエージェントモデルを適用してもOPS～NE間のトラフィックを削減できない．第2に本アーキテクチャは，既存NEへのエージェントシステムの追加や新しいエージェントシステムの導入が不要である．このため，従来のOPSから本モデルにより実現したOPSへのマイグレーションは，NE依存型モバイルエージェントモデルに比較し，容易であると考える．第3に，既存NEにエージェントシステムを搭載できない（特にJavaVM上に構築されたシステムの文献が多いが，モバイルエージェントはJavaVMを持たないNEには搭載できない）場合も存在する．また，特定のモバイルエージェントが搭載可能であることが，新しいNEを導入するための必須条件となり，NEの選択肢が限定されてしまう．第4に，既存NEにエージェントシステムを搭載できない場合は，新しくエージェントシステムを導入する必要があり，NEが設置されているすべてのビルに新しいハードウェアを導入しなければならない．NEが設置されているビルの数やNEの台数が多い場合は，多くのハードウェアを導入する必要があり，導入コストが膨大になるものと予想される．

7. あとがき

本稿では，多数の小規模IAサーバを束ねて高い処理能力を得ることが可能なD3Aについて説明した．本アーキテクチャの優位点を以下にまとめる．

- OPSのTCOを大幅に削減することが可能である．
- ハードウェアベンダやミドルウェアベンダの戦略に影響されにくい．
- OPSのハードウェアを安価な小規模IAサーバ単位で柔軟に拡張可能である．
- OPSアプリケーションへの機能追加に対して，迅速かつ柔軟に対応可能である．
- 従来のOPSと比較して，OPS性能指標値（警報受信能力）が最大で約5倍となる．

本アーキテクチャを適用し，開発したxGSN（serving/gateway General packet radio service Support Node）対応OPSは，すでにサービスを開始し，安定した運用実績をあげている．また，TCO削減効果が非常に大きいことから，既存OPSの全面的なリプレースとなる次期NW-OPSを本アーキテクチャに基づいて開発中である．さらに，本アーキテクチャについては，他の分野のシステムに対しても適用可能性があると考えている[27][28]．

文　献

[1] 大貫，ほか：“オペレーションシステム特集，”本誌，Vol. 8，No. 1，pp. 6-30，Apr. 2000.

[2] 松本，ほか：“モバイルマルチメディアサービス高度化共通基盤（M^{3}In）の開発，”本誌，Vol. 11，No. 2，pp. 70-81，Jul. 2003.

[3] PCクラスタ；http://www.pccluster.org/

[4] 水野（松木）由子，伊達 進，甲斐島 武：“グリッド技術による医用インフラストラクチャ，”信学論（B），Vol. J85-B，No. 8，pp. 1261-1268，Aug. 2002.

[5] 朴 泰祐，佐藤 三久，小沼 賢治，牧野 淳一郎，須佐 元，高橋 大介，梅村 雅之：“HMCS-G：グリッド環境における計算宇宙物理のためのハイブリッド計算システム，”情報処理学会論文誌コンピューティングシステム，Vol. 44，No. SIG11（ACS 3），pp. 1-13，2003.

[6] I. Satoh: “A Framework for Building Reusable Mobile Agents for Network Management，”Proceedings of Network Operations and Managements Symposium（NOMS'2002），IEEE Communication Society, pp. 51-64, Apr. 2002.

[7] D. Gavalas, D. Greenwood, M. Ghanbari and M. O'Mahony: “An Infrastructure for Distributed and Dynamic Network Management based on Mobile Agent Technology，”Proceeding of Conference on Communications（ICC'99），pp. 1326-1366, 1999.

[8] 藤井 邦浩，渡邉 稔也，秋山 一宜，高橋 和秀，谷川 延広：“分散データ駆動型アーキテクチャの実装と性能評価，”信学技報，Vol. 103，No. 43，pp. 1-6，May 2003.

[9] K. Akiyama, T. Watanabe, K. Takahashi and N. Tanigawa: “A Distributed DataDriven Architecture for Operations Support Systems，”NOMS2004.

[10] 佐藤 豪一，高橋 和秀，谷川 延広：“組織階層型モバイルエージェントモデルに基づいたOSSの実用化，”SACSIS2004，May 2004.

[11] 佐藤 豪一，昆 孝志，秋山 一宜，高橋 和秀，神宮司 誠：“分散データ駆動型アーキテクチャによるOSSの実用化，”FIT2004，LC-003，Sep. 2004.

[12] 金子 伊織，牧野 雄至，高橋 和秀，神宮司 誠：“分散データ駆動型アーキテクチャの信頼性評価，”信学技報，Vol. 104，No. 164，pp. 25-30，Jul. 2003.

[13] 西田 幸一，藤嶋 貴志，久川 誠，高橋 和秀，神宮司 誠：“分散データ駆動型アーキテクチャの性能評価と移植性評価，”信学技報，Vol. 104，No. 565，pp. 7-12，Jan. 2005.

[14] 高橋 和秀，昆 孝志，秋山 一宜，神宮司 誠：“組織階層型モバイルエージェントモデルに基づいたネットワーク管理システムの実用化，”信学論（D），Vol. J88-D1，No. 2，pp. 401-420，Feb. 2005.

[15] 佐藤 豪一，久川 誠，高橋 和秀，牧野 雄至，神宮司 誠：“分散データ駆動型アーキテクチャを用いたトラヒックコントロールシステムの設計，”信学技報，Vol. 104，No. 707，pp. 13-18，Mar. 2005.

[16] 藤井 邦浩，久川 誠，藤嶋 貴志，高橋 和秀，牧野 雄至，神宮司 誠：“分散データ駆動型アーキテクチャによるトラフィック管理システムの実現，”信学技報，Vol. 104，No. 707，pp. 7-12，Mar. 2005.

[17] 高橋 和秀，牧野 雄至，神宮司 誠：“分散データ駆動型アーキテクチャによる大規模OSSの設計，”信学技報，Vol. 104，No. 707，pp. 43-48，Mar. 2005.

[18] “W3C”; http://www.w3.org

[19] “Business Process Execution Language for Web Services”; http://www-106.ibm.com/developerworks/library/ws-bpel

[20] 瀬社家 光，木村 伸宏，藤原 健：“オペレーションシステムにおける輻輳制御の検討，”信学技報，Vol. 99，No. 34, 35，pp. 7-12，Apr. 1999.

[21] 木村 伸宏，瀬社家 光：“分散OSSにおける高信頼化の検討，”信学技報，Vol. 100，No. 451, 452, 453, 454，pp. 29-34，Nov. 2000.

[22] 稲森 久由，上田 清，須永 宏：“大規模通信ノードシステムに対応したTMN構築支援技術の検討，”信学論（B），Vol. J84-B，No. 3，pp. 461-477，Mar. 2001.

[23] H. Tojo, H. Ikeda, T. Goto and I. Yoda: “Development Method for the Operation Scenario Program using Script Language，”IEICE, Japan, Vol. J82-B, No. 5, pp. 877-885, May 1999.

[24] OGSA; http://www.gridforum.org/mail_archive/ogsa-wg/Archive/msg00172.html

[25] Globus Alliance; http://www.globus.org/

[26] “GT3”; http://www-unix.globus.org/toolkit/about.html

[27] 秋山 一宜，藤井 邦浩，金内 正臣，渡邉 稔也，高橋 和秀，谷川 延広：“IPサービスを実現するための分散データ駆動型アーキテクチャの提案，”信学技報，Vol. 102，No. 706，pp. 1-6，Mar. 2003.

[28] 秋山 一宜，ほか：“分散データ駆動型アーキテクチャによる広告配信サービスの試作，”IEICE Technical Report，TM2003-114，pp. 19-24，Mar. 2004.

用 語 一 覧

BPEL4WS：Business Process Execution Language for Web Services
CORBA：Common Object Request Broker Architecture
CPU：Central Processing Unit
D3A：Distributed Data Driven Architecture
（分散データ駆動型アーキテクチャ）
DBMS：DataBase Management System
DNS：Domain Name System（ドメインネームシステム）
EA：External Adaptor（外部アダプタ）
ECIF：Element Configuration Information File
（エレメント構成情報ファイル）
ECM：Element Configuration Manager（エレメント構成マネージャ）
GbE：Gigabit Ethernet（ギガビットイーサネット）
GT3：Globus Toolkit3
HPC：High Performance Computing
（ハイパフォーマンスコンピューティング）
IA：Intel Architecture
IP：Internet Protocol
JDBC：Java DataBase Connectivity
LE：Logical Element（論理エレメント）
LPD：Logical Path Data（論理経路データ）
LPDD：LPD Driver（LPDドライバ）
LPDM：LPD Manager（LPDマネージャ）
mps：message per second
NE：Network Element（ネットワーク設備）
OGSA：Open Grid Service Architecture
OPS：OPeration System
OSS：Open Source Software（オープンソースソフトウェア）
PE：Physical Element（物理エレメント）
RDBMS：Relational DataBase Management System
RMI：Remote Method Invocation
SOAP：Simple Object Access Protocol
TCO：Total Costs of Ownership
W3C：World Wide Web Consortium
xGSN：serving/gateway General packet radio service Support Node
XML：eXtensible Markup Language