Grid Datafarm におけるスケジューリング・複製手法の性能評価

竹房　あつ子[†1]　建部　修見[†2]
松岡　聡[†3,†4]　森田　洋平[†5]

グリッド技術を基盤にした大容量データに対する遍在するアクセスを可能にする技術をデータグリッドと呼び，複数のシステムの設計・実装が行われている．しかしながら，それらは実験段階にあり，データグリッドアーキテクチャの設計方針の妥当性や性能に関する議論は不十分である．本稿では，Bricks グリッドシミュレータにデータグリッドシステムに対する拡張を行い，Grid Datafarm アーキテクチャに基づくデータグリッドモデルにおける高エネルギー物理アプリケーションジョブの性能について比較・調査した．データグリッドモデルでは，Central モデルと Tier モデルを比較し，Tier モデルでは様々なスケジューリングと複製手法を適用し，2007 年に開始される CERN の高エネルギー物理実験を想定してその性能を評価した．評価では，Central で効率よく処理できること，Tier ではバックグラウンドに複製を作る手法を用いると効率よく処理でき，1 サイトの性能が Central より低い構成でも Central よりよい性能を示すことが分かった．

Performance Analysis of Scheduling and Replication Algorithms on Grid Datafarm Architecture

Atsuko Takefusa,[†1] Osamu Tatebe,[†2] Satoshi Matsuoka[†3,†4]
and Youhei Morita [†5]

Data Grid is a Grid environment for ubiquitous access and analysis of large-scale data. Due to its early research status, the performance of petabyte-scale Data Grid models in a realistic data processing setting have not been well investigated. By enhancing our Bricks Grid simulator to be able to simulate Data Grid scenarios, we investigate and compare the performance of different Data Grid models in the Grid Datafarm architecture, mainly categorized into the central and the tier models but with varying scheduling and replication strategies, under realistic assumptions of job processing for the CERN LHC experiments. Our results show the central model is efficient but the tier model with greater amount of resources and speculative class of background replication policies is quite effective and achieves higher performance while each tier being smaller than the central model.

1. はじめに

グリッド技術を基盤にした大容量データに対する遍在するアクセスを可能にする技術をデータグリッドと呼び，高エネルギー物理学，天文学，ヒトゲノム等の研究分野で重要視されている．一つの例として，CERN で行われる大規模素粒子加速器実験 (Large Hadron Collider (LHC)) があり，Grid Datafarm[1),2)]，EU DataGrid[3)]，GriPhyN[4)] プロジェクトなどでは LHC 実験をターゲットとしてデータグリッドシステムの設計・実装が行われている．LHC 実験では，2007 年より 4 つの実験グループ，粒子検出器により毎年ペタバイト規模の観測データが生成される．数千人の物理学者が素粒子物理データ解析において協力・競争をするため，グリッド技術を用いた世界規模のデータ解析環境の構築を目指している．このようなペタバイトに及ぶデータ解析では大容量のディスクおよび計算資源を必要となるため，MONARC(Models of Network Analysis at Reginal Centres)[5)] プロジェクトでは各国に地域センタを配置する，地球規模の多階層データグリッドモデルを提案している．このモデルでは，0 層センタを CERN に，1 層センタとしてヨーロッパ，アメリカ，アジアに，2 層センタとして各国に，3 層センタは各大学・研究所に置かれる．

このような大規模データインテンシブコンピューティングでは，実験器具，計算機，ディスク，研究者，データ，そしてアプリケーションが世界的に分散している．これらの資源への高速，安全，効率的で信頼性のあるア

†1 お茶の水女子大学 Ochanomizu University
†2 産業技術総合研究所 National Institute of AIST
†3 東京工業大学 Tokyo Institute of Technology
†4 国立情報学研究所 National Institute of Informatics
†5 高エネルギー加速器研究機構 High Energy Accelerator Research Organization

クセスが必要不可欠であり，グリッド，クラスタ，ネットワーク技術がその達成の鍵となる．一方，これらのデータグリッドシステムは現在開発・実験段階にあり，提案されているデータグリッドシステムアーキテクチャの設計方針の妥当性や実用性，実アプリケーションを想定した性能評価に関する議論は不十分である．

本稿では，Bricks グリッドシミュレータに対しデータグリッドのためのディスクシミュレータの拡張と複製機構を組み込み，データグリッドシステムモデルとその性能について Grid Datafarm アーキテクチャを想定して高エネルギー物理アプリケーションジョブの性能を比較・調査した．256 ノードの Prest III クラスタ上で LHC 実験アプリケーションを想定した 1 年間分の Bricks シミュレーションを約 800 回実行し，1 つのサイトで集中的にデータ解析を行う Central モデルと MONARC で提案されている Tier モデルを比較した．また，Tier モデルでは様々なスケジューリングと複製アルゴリズムを提案・適用し，2007 年の LHC 実験を想定してその性能をシミュレーションにより評価した．

2. Grid Datafarm アーキテクチャ

大規模データインテンシブコンピューティングでは，資源が世界的に分散しているため，資源への高速，安全，効率的で信頼性のあるアクセスが必要不可欠である．これらの大規模データインテンシブアプリケーションのデータは殆ど更新されることがなく，write-once read-many モードでアクセスされる傾向がある．よって，世界規模のペタスケールデータを効率的に共有するには，ファイルの複製生成が負荷分散，アクセスバンド幅，耐故障性において非常に効果的である．

また，最適なファイル複製および計算ノードの選択，出力および一時的なファイル領域の割り当て方法等，スケジューリングは効率的なジョブの実行のために必要不可欠である．ファイル複製生成手法においても，どのファイルをいつ，どこに複製を生成するか，またディスク領域の不足を避けるためにどの複製をいつ削除するかなど，データの分散が効率的なジョブ実行の鍵となる．

一方，データサイズの増加により，データアクセスバンド幅と CPU パワーが効率的なデータ処理には必要不可欠である．すなわち，ペタバイトのデータ処理を可能にするデータグリッドでは高速ファイル転送や効率的な複製管理だけでなく，高速データアクセスや高速データ処理を実現しなければならない．TB/ 秒規模のバンド幅でさえ，ペタバイト規模のデータの処理には不十分である．一般に，TB/ 秒におよぶバンド幅はグリッドのような分散計算環境では実現不可能と思われるが，大規模データインテンシブコンピューティングではデータアクセスの局所性を利用することにより，TB/ 秒におよぶバンド幅を実現することができると考えられる．

データアクセスの局所性により，計算ノード群と独立

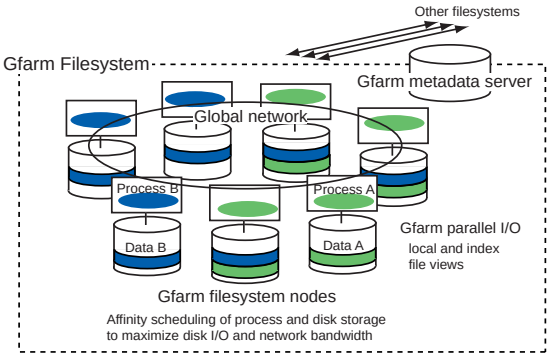

図 1 Gfarm ファイルシステム．

したネットワーク越しの I/O アクセスは効率的な実行の妨げとなる．一般に，データグリッドにおけるデータ処理システムではペタバイト規模のデータを扱うため，計算に必要なデータを HPSS 等の高性能ストレージシステムに格納し，適宜計算ホストにロードして処理する．しかしながら，計算ノードとデータが密に結合していること，すなわち，owner-computes，または move-the-computation-to-data 手法を適用した方が，データ並列アプリケーションをスケーラブルかつグリッド上でより効率的に処理することができる．

Grid Datafarm システムでは，データアクセス局所性のあるデータインテンシブアプリケーションのデータアクセスバンド幅を向上させることを目的とし，計算ノードとデータを融合させたアーキテクチャを提案している．Grid Datafarm では，グリッド上の数千，数万ものディスク・計算ノードを Gfarm ファイルシステムと呼ばれる 1 つのファイルシステムイメージとして管理する．図 1に Gfarm ファイルシステムを示す．Gfarm ファイルシステムでは，ペタバイト規模のグローバル並列ファイルシステムを提供する．クラスタノードのディスクに分割・格納されたデータに対し，owner-computes ルールでプロセスをスケジュールし，並列実行する．これにより，数万ノードにおよぶグリッド上のクラスタを利用し，スケーラブルな I/O バンド幅，並列処理を実現可能とする．

本稿の評価では，データグリッドシステムにとして Grid Datafarm アーキテクチャを想定する．

3. シミュレーションモデル

データグリッドに対して CERN LHC 実験をアプリケーションモデルとして評価する．

3.1 データグリッドアプリケーションモデル: CERN LHC 実験

LHC 実験では，粒子の衝突実験から測定されるペタバイト規模のデータ（イベント）を収集し，以下の段階的な処理が（）内の頻度で行われる[5]．

Large: **RAW→ESD:** RAW データを再構成し，

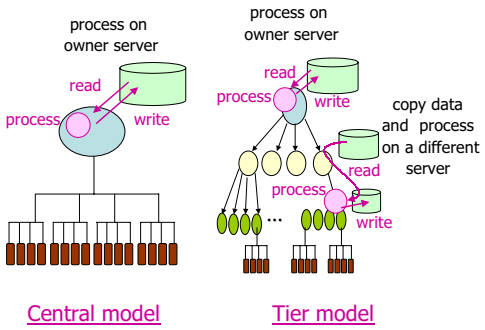

図 2 Central モデル（左）と Tier モデル（右）．

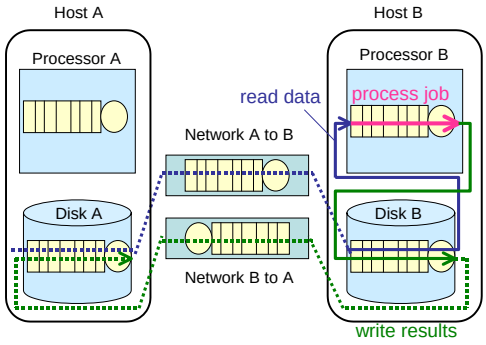

図 3 データグリッドコンピューティングにおける Bricks シミュレーションの流れ．

ESD(Event Summary Data) を生成する（2-4/ 年）

Medium: **ESD→AOD:** ESD を用い，AOD(Analysis Object Data) を生成する（1/ 月）

Small: **AOD→TAG:** AOD を用い，TAG データを生成する（1/4 時間）

LHC 実験での典型的なジョブ Large，Medium，Small は，いずれも数百万もの物理イベント処理の集合からなり，それぞれのイベント処理は独立なためイベント単位で並列データ処理が可能である．各ジョブはデータグリッドシステム上で次のように処理される．

(1) クライアント計算機でユーザ（物理学者）がジョブをデータグリッドシステムに投入する

(2) データグリッドスケジューラがジョブに対して適切なサーバ群を選択する

(3) 各サーバは割り当てられたタスクを処理する

(4) サーバは指定されたディスクに出力データを送る（クライアントには統計情報のみが返される）

選択された各サーバはそのサーバ上で処理されるタスク（ジョブの一部）に要するデータがローカルディスクにない場合，ネットワーク経由でロードされる．

ジョブの処理全体に要する時間 $T\_response$ は，T_read，$T\_process$，T_write を入力データの読み込み時間，計算サーバでのジョブの処理時間，出力結果の書き出し時間としたとき，以下のように表される．

$$T\_response = T\_read + T\_process + T\_write \quad (1)$$

3.2 データグリッドアーキテクチャ

MONARC[5] では，単一サイトで構築可能な計算・ディスク資源に制限があるという前提で，多階層地域センタモデルを提案した．一方，近年のコモディティ PC およびクラスタリング技術の発展は目覚ましい．Grid Datafarm では，それらを利用し大規模ディスククラスタを設計・構築しており，5) で必要とされている計算資源を単一サイトで確保できる可能性は十分にある．

本研究では Grid Datafarm アーキテクチャを仮定して単一サイトで全てのジョブを処理する Central モデルと，多階層の地域センタでジョブを処理する MONARC 型の Tier モデルを比較する（図 2）．Tier モデルでは効率的なデータ処理のため，適切なユーザジョブの割当てと適切なデータ複製の必要がある．それらスケジューリングおよび複製手法の詳細は 5節で述べる．

4. Bricks のデータグリッド拡張

Bricks グリッドシミュレータ[6] は Java で実装された離散イベントシミュレータであり，典型的なグリッドのスケジューリングモジュール群（Scheduling Unit）と動的なシミュレーショングリッド環境を提供し，様々なスケジューリングアルゴリズムの解析を可能にする[7),8)]．Bricks を用いてデータグリッドアプリケーションの性能を評価するため，次のように Bricks システムを拡張した．

- ローカルディスク I/O オーバヘッドの表現
- 複製マネージャの提供
- 複製カタログの提供
- ディスク管理機構

4.1 ローカルディスク I/O

データグリッドにおける精密なシミュレーションではディスクアクセスは無視できないため，ディスクアクセスの挙動を待ち行列を用いて表現するよう Bricks システムを拡張した．Bricks では図 3のように待ち行列でデータのネットワーク通信遅延，プロセッサでの処理遅延に加え，ディスク I/O の遅延を表現する．図中の実線は，ジョブが必要とするデータが Host B に格納されており，その Host B 上でジョブが処理される場合のワークフローであり，Disk B からデータを読み，Processor B でそのジョブを処理し，結果を Disk B に格納する．一方，破線と実線を合わせたワークフローは，ジョブ処理に要するデータが計算ホストの Host B とは異なる Host A に格納されており，結果も計算ホストとは異なる Host A に格納する場合を示す．この場合，データを Disk A から Network A を介して Disk B に格納した後読み込み，Processor B でジョブを処理して結果を

Disk B, Network B を介して Disk A に格納する．

本シミュレーションでは，Processor と Disk の待ち行列は時分割処理され，Network の待ち行列は FCFS(First-Come-First-Served) で処理される．Network ではデータは指定された論理パケットに分割・転送される．また，図 3で示すように，Disk では read, write 時の遅延とも，1 つの待ち行列により表す．

4.2 複製マネージャと複製カタログ

2節で述べたように，大規模データインテンシブコンピューティングではデータの分散そのものがデータグリッドシステム上での効率的なジョブ実行の鍵となる．よって，データグリッドのための拡張として Bricks Scheduling Unit モジュールで複製マネージャと複製カタログ機構を提供するようにした．

複製マネージャはグリッド上の資源情報を把握し，適切にデータの複製を生成する．複製マネージャの詳細は 5節で述べる．また，複製カタログはデータの複製がグリッド上のどのホストにあるかを把握しており，スケジューラや複製マネージャからデータの所在に関する問い合わせがあると，そのデータ（複製を含める）を格納しているホスト群を通知する．

4.3 ディスク管理機構

データグリッドアプリケーションでは大規模データを扱うため，全てのデータが任意のホストで格納できるわけではない．また，負荷分散，耐故障性，安全性等のためにデータの複製が作られるため，ますますグリッド全体のディスク空間が圧迫される．よって，個々のローカルディスクで格納しているデータを管理するとともに，ディスク空間が圧迫されたときに，負荷分散，冗長性の点で削除しても大きな影響を与えないデータ（ただし，オリジナルと複製は等価であるとする）を適宜削除し，グリッド上での安定したジョブ処理が継続できるようにした．データ削除手法の詳細は 5.4節で述べる．

5. Tier モデルのスケジューリングと複製管理

多階層分散データグリッドシステムでは，ユーザのジョブを効率よく処理するために適切なスケジューリングと複製手法を用いなければならない．多くの複製を生成すると効率的な負荷分散が実現でき，応答時間の短縮が望めるが，ディスクサイズの制限やネットワークバンド幅への圧迫により性能に悪影響を与える．

5.1 Grid Datafarm における複製管理

2節で述べたように，Grid Datafarm システムでは拡張ストライピングクラスタファイルシステムを提供し，データグリッドアプリケーションに必要なデータを断片化してメタデータにより管理する．メタデータはファイルステータス，ファイル断片ステータス，ディレクトリ，複製カタログ，ファイルシステムノードステータスからなり，ファイルシステムメタサーバが管理する．これらの情報により，分散データグリッドシステム上で耐故障性，広バンド幅，低レイテンシ，負荷分散の実現が可能となる．本稿のスケジューリング・複製手法は，これらのメタデータを利用することを前提とする．

5.2 オンラインスケジューリングアルゴリズム

スケジューラは発行されるジョブに対して入力データのオーナーである DataSourceHost，ジョブを処理する ComputeHost，結果が格納される DataDestinationHost を適切に選択する．DataSourceHost!=ComputeHost または ComputeHost!=DataDestinationHost の場合，入力／出力データの複製がオンデマンドに生成される．一方，複製マネージャは定期的にグリッド環境情報を収集し，複製の生成，移送，削除をバックグラウンドで管理する．

シミュレーションでは，次のオンラインスケジューリングアルゴリズムを比較・評価する．

Greedy: 処理時間を最短にすることを目指すアルゴリズムであり，MCT(Minimum Completion Time) として知られている[9]．スケジューラは式 (1) に示す応答時間が最短になるように DataSourceHost，ComputeHost，DataDestinationHost を割り当てる．

OwnerComputes: 発行されたジョブの処理に必要な入力データを格納しているホストの中から処理時間が最短となるホストを計算ホストとして選択する．この場合，DataSourceHost，ComputeHost，DataDestinationHost は全て同じホストとなる．

LoadBound-Read: スケジューラは MCT で以下を満たすホストから計算ホストを選択する．

$$Perf_{specified} > Perf_{estimated} \quad (2)$$

$$Perf_{estimated} = Perf/(LoadAvg + 1) \quad (3)$$

$Perf$ はサーバの性能，$LoadAvg$ は負荷平均値，$Perf_{estimated}$ はある時点でのサーバの処理性能の見積もり値，$Perf_{specified}$ はあらかじめ指定した性能値を示す．ジョブは適切なホストから入力データを読み込み，ComputeHost に出力結果を格納する．

LoadBound-Write: スケジューラは以下の $T_{duration}$ を最小にする ComputeHost を選択する．

$$T_{duration} = T_{read} + T_{process} \quad (4)$$

この際，選択された ComputeHost が式 (2) を満たさなければ，式 (3) が最大となるホストに出力データを送信する．これにより，処理能力のあるホストへの負荷の分散を図る．

全てのアルゴリズムにおいて，$Perf_{specified}$ があるホストの性能 $Perf$ より大きい場合，そのホストはスケジューリングの対象から外れる．

5.3 複製アルゴリズム

複製マネージャとして定期的にデータグリッドシステム上の計算ホストの状況を調べ，適宜複製の生成，移送を実施する LoadBound-Replication とジョブ終了後に必ず複製を生成する Aggressive-Replication を用いる．

LoadBound-Replication: 複製マネージャは定期的に全てのホストに対して式 (3) で $Perf_{estimated}$ を算出

する．あるホストの $Perf_{estimated}$ が式 (2) を満たす場合，$Perf_{estimated}$ が最小のホストから最大のホストへと複製を生成して転送する．

この際，複製マネージャは選択されたホスト上のアクセス率 AR が最も高いデータの複製を生成する．

$$AR = N_{accesses}/(T_{current} - T_{stored}) \quad (5)$$

$N_{accesses}$ はデータへの総アクセス数，$T_{current}$ と T_{stored} は現在の時刻とそのデータがディスクに格納された時刻を示す．ここで，データのアクセス数とは，あるファイル（データ）を入力とするジョブが発生すると，そのデータのアクセス回数が 1 増えるものとする．

Aggressive-Replication: 複製マネージャに対してクライアントホストからのジョブ終了の通知があると，複製マネージャはそのジョブが生成したデータの複製を生成し，$Perf_{estimated}$ が最大のホストへ転送する．

5.4 評価でのスケジューリングと複製手法の組合せ

評価では，5.2節で提案した 4 つのスケジューリング手法と 5.3節で提案した 2 つの複製手法を用いた場合とスケジューリング手法のみを用いた場合の計 12 通りの組合せでその性能を調査する．

スケジューリング手法または複製手法により，いくつかのホストでは生成されたデータのオリジナルコピーを管理し，それらのデータの複製が他のホストに転送される．すなわち，データは移動するのではなく，コピーされる．例えば，DataSourceHost と ComputeHost が異なる場合，そのジョブに要するデータが DataSourceHost から ComputeHost のディスクにコピーされる．もし，転送先のディスクの空き領域が不十分だと判明した場合（スケジューラが検出），またはデータグリッドシステム上のホストのうち x% のホストのディスクの使用領域が y% 以上となった場合（複製マネージャが検出），”複製の削除 ”を行う（パラメータ x，y は実行時に指定）．本シミュレーションでは複製の削除のために，以下のアルゴリズムを用いる．

(1) データグリッドシステム上に複製をもつデータのリストを作る

(2) (1) のデータを最後にアクセスされた時刻が古い順に並べる

(3) (2) のリストの最初から N 個のデータを対象にし，式 (6) でデータのアクセス率 AR_{elim} を計算する

$$AR_{elim} = N_{accesses}/(T_{current} - T_{stored})/N_{copies} \quad (6)$$

N_{copies} はあるデータの複製の総数を示す

(4) 以下の条件を満たすまで式 (6) の小さいデータを順に削除する．$Size_{total}$，$Size_{available}$ はディスクの総容量と利用可能容量，$Compactness$ は複製削除の頻度調節パラメータである．

$$Size_{total} \times Compactness > Size_{available} \quad (7)$$

(5) 式 (7) が満たされない場合は，次の N 個のデータに対して (3) 以降のステップを繰り返す

本シミュレーションでは，N を 10 とした．

表 1 シミュレーション環境のパラメータ．

モデル	ディスク容量 [PB]	サイト性能 [MSI95]	サイト内ノード数
Central	2	0.5-1.8	10000
Tier	Tier0(x1): 2	0.6/0.5/0.4	10000
	Tier1(x4): 1	0.3/0.25/0.2	5000
	Tier2(x16): 0.1	0.03/0.025/0.02	500

表 2 LHC 実験のジョブパラメータ．各ジョブのイベント数は全て 1G 個．

Job	計算量 [GSI95*sec]	平均頻度	入力 [TB]	出力 [TB]
Large	1000	1/4[months]	1000	100
Medium	25	1/1[month]	100	10
Small	5	1/4[hours]	10	0.1

表 3 LHC 実験データ RAW(1PB)，ESD(100TB)，AOD(10TB)，TAG(10GB) の平均増加量．

フェイズ	データとその個数
0-3mth	1PB(1)
4-7mth	1PB(2), 100TB(1)
8-11mth	1PB(3), 100TB(2), 10TB(4)
12-15mth	1PB(4), 100TB(3), 10TB(8), 10GB(720)
16-19mth	1PB(5), 100TB(4), 10TB(12), 10GB(1440)
20-23mth	1PB(6), 100TB(5), 10TB(16), 10GB(2160)

複製削除の命令は複製マネージャが発行するが，各サイトでのディスク空き領域の調査，複製削除アルゴリズムの実行はローカルディスクマネージャが行うため，スケーラブルにデータグリッドシステム上のディスク領域管理が可能である．

6. シミュレーションによる評価実験

評価では，ジョブの応答時間を比較する．

6.1 シミュレーションシナリオ

3節で述べたように図 2の 2 つのモデルを比較する．

Central モデル: 全てのジョブリクエストが処理できる十分な計算性能をもつ 1 つのサイトに全てのデータが格納されており，そこで全てのジョブを処理する．安定したジョブ処理が可能となる計算性能は，待ち行列理論より見積もることができる．

Tier モデル: 1 サイトの負荷が増加すると，他の地域センタにデータの複製を生成・転送し，そこでジョブを処理する．Tier モデルでは，12 種類のスケジューリング・複製手法の組合せを用いる．

評価では，データグリッドシステム上に 1 つのデータグリッドスケジューラを想定し，サイトに対してジョブを割り当てるものとする．サイト内ではローカルスケジューラが各ホストにジョブを割り当てる．

表 1にシミュレーション評価実験環境のパラメータを示す．これらのパラメータは GriPhyN のシミュレーション[10] における設定パラメータをもとに決定した．

Tier モデルでは， Tier0 が 1 サイト， Tier1 が 4 サイト， Tier2 が 16 サイトとし， Tier3 にはユーザの各計算機があるものとする． Tier 間の性能比は (Tier0, Tier1, Tier2) = (0.6, 0.3, 0.03), (0.5, 0.25, 0.025), (0.4, 0.2, 0.02) [M SI95(10^6SpecINT95)] の 3 通りとした． Central の最低性能を 0.5[MSI95] としたのは，0.453318[MSI95] より性能が低い場合飽和して想定する LHC ジョブを処理できないことが待ち行列理論で明らかなためである．

WAN とローカル I/O のバンド幅は 2007 年の時点で実現する技術を想定し，それぞれ 10[Gbps] と 100[MB/sec] とした．また，各ジョブは 1 つのサイト内で処理されるものとし， Grid Datafarm システムを想定して各サイトでは並列 I/O，並列処理することにする．一般のクラスタ並列ファイルシステムでは， I/O ノード数を増やすとディスク I/O バンド幅が LAN のバンド幅により制限されるが， Grid Datafarm アーキテクチャではデータアクセス局所性のあるファイルに対し数千ノードのスケーラブルな I/O バンド幅が期待できる．すなわち， Tier0 の各ホストのローカル I/O バンド幅が 100[MB/sec]，ノード数が 10000 の場合， Tier0 での総バンド幅は 1[TB/sec] となる．

3.1節で述べたように，シミュレーションでは実際の LHC 実験での 3 つの異なるレベルの解析ジョブ（表 2）を複数同時に実行する．表 2のデータの粒度・頻度の場合，表 1の Central モデルでは全てのジョブの平均応答時間が待ち行列理論で 38.575-1.337[hours] になると予測できる．また，表 2より LHC 実験の RAW(1PB)， ESD(100TB)， AOD(10TB)， TAG(10GB) のデータは表 3のように増加する．表中の () 内の数値は各フェイズでの複製を含めない各データの総数を示す．

表 3に時間の経過に対する実験データの総数の変化の平均値を示す．表中のフェイズ欄にある mth は月を表す．評価では，表 3の 8mth から 23mth 終了までの 1 年分のシミュレーションを 800 回程度行った．シミュレーションの実行には，東京工業大学の Presto III クラスタ（Dual Athlon MP 1900+， 768MB Memory， 256 nodes）を用いた．全シミュレーションの開始時には全てのデータ（1PBx1， 100TBx2， 10TBx4）は Tier0 に格納しておく．また， 1PB の RAW データは HPSS のような異なるディスク領域に格納されることを想定し，各シミュレーションの間 RAW データはシミュレーション環境中のディスク領域で増加しないものとする．

GriPhyN のシミュレーション[10] では，データアクセスに関して時間的，地域的，空間的局所性を挙げている．本シミュレーションでは，ランダムアクセス（局所性なし）と時間的局所性（最近アクセスされたデータは再びアクセスされやすい）を持つアクセスパターンを想定した☆． LHC 実験では新しく加速器から得られた観測情報データや興味深い解析結果が得られるデータに対して頻繁にアクセスが発生する傾向があるため，時間的局所性を適用する．一方，地域的，空間的局所性の重要性が明らかでないため本シミュレーションでは用いない．

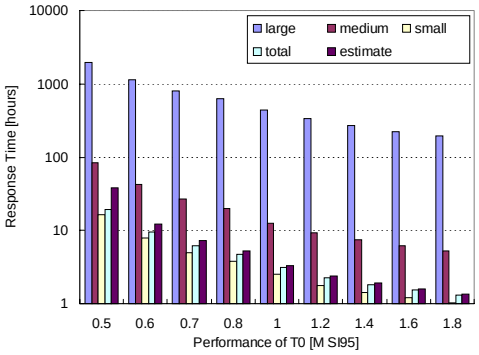

図 4 Central の異なる処理性能における応答時間の比較． Tier0 の性能は 0.5-1.8[MSI95].

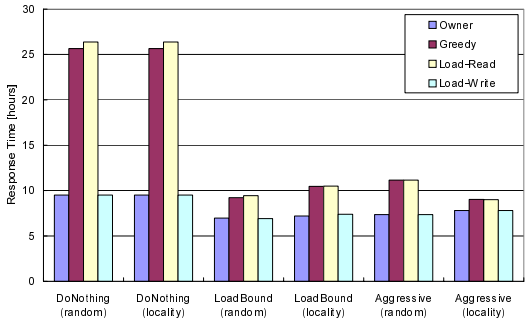

図 5 Tier で異なるスケジューリング・複製手法を用いたときの応答時間． (Tier0, Tier1, Tier2) = (0.6, 0.3, 0.03)[MSI95].

6.2 シミュレーションによる評価結果

図 4， 5， 6， 7に Central と Tier の実験結果を示す．これらは，各システムモデル，時間的局所性のある／ないアクセスパターン，スケジューリング・複製手法の組合せに対してそれぞれ 1 年分のシミュレーションを 10 回行い，その総平均応答時間を算出したものである．シミュレーション中に発行されたジョブ Large， Medium， Small の総数はそれぞれ 30， 102， 21693 であった．

図 4では Central での平均応答時間を示す．グラフ中の large， medium， small は表 2のジョブ Large， Medium， Small であり， total は全ジョブの平均応答時間， estimate は待ち行列理論で算出した平均応答時

☆ 時間的局所性を表現するため，ジョブが必要とするデータを新しく生成された順序で配列に入れ，次のようにデータを選択して新しいデータがアクセスされる可能性を高くした．
$index = rand \times n$
ここで、 $index$ は選択するデータの配列上のインデックス、 $rand$ は平均 0，標準偏差が 0.25 となる正規乱数列， n は総データ数（複製は数えない）を示す．

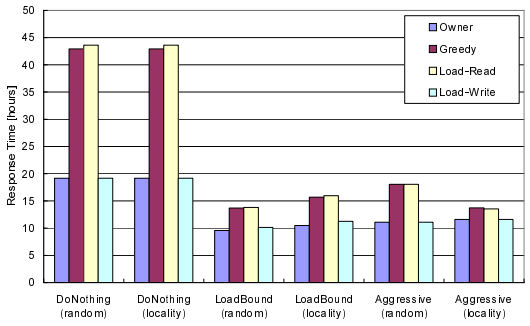

図 6 Tier で異なるスケジューリング・複製手法を用いたときの応答時間．(Tier0, Tier1, Tier2) = (0.5, 0.25, 0.025)[MSI95].

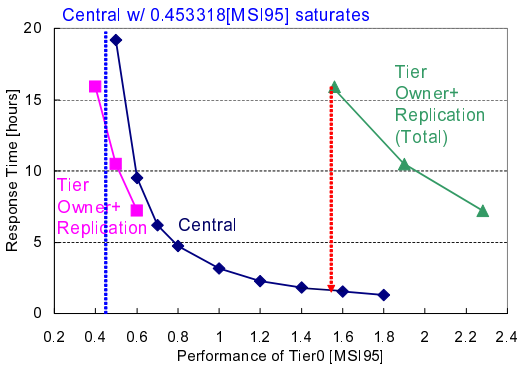

図 7 Central と Tier（OwnerComputes と LoadBound-Replication を適用）の応答時間の比較．

間の見積値を示す．グラフの x 軸には Tier0 サイトの総計算性能を示し，y 軸にはログスケールで平均応答時間を [hours] で示す．図 4より，サイトの総性能の低下に伴い応答時間が急激に増加していくことが分かる．

図 5，6は Tier で異なるスケジューリングと複製手法を適用したときの Large，Medium，Small の総平均応答時間を表している．図 5，6の各 Tier サイトでの総計算性能はそれぞれ (Tier0, Tier1, Tier2) = (0.6, 0.3, 0.03)，(0.5, 0.25, 0.025)[MSI95] である．x 軸には複製手法とデータアクセスパターンを示しており，DoNothing は複製マネージャで複製を作らない場合，LoadBound は LoadBound-Replication，Aggressive は Aggressive-Replication を適用した結果である．() 内の random，locality はアクセスパターンに時間的局所性がない／ある場合を示す．Owner，Greedy，Load-Read，Load-Write はそれぞれスケジューリング手法 OwnerComputes，Greedy，LoadBound-Read，LoadBound-Write を示す．

図 5，6とも，Greedy と LoadBound-Read を用いた場合に平均応答時間が増大している．これは，LHC 実験解析ジョブの入力データはテラバイトからペタバイトに及ぶため，負荷分散のために入力データをオンデマンドにコピーするとそのオーバーヘッドが大きいからである．一方，各スケジューリング手法に LoadBound-Replication および Aggressive-Replication を適用した場合，いずれのスケジューリング手法を用いた場合も性能が著しく向上していることがわかる．これはバックグラウンド複製手法により，コピーのオーバーヘッドが応答時間に含まれないためである．よって，入出力データサイズが非常に大きいデータグリッドアプリケーションではバックグラウンドでデータの複製を作る手法が有効であることが分かる．OwnerComputes では，Aggressive-Replication より LoadBound-Read との組合せの方がよい性能を示した．これは Aggressive-Replication では時間の経過によりディスク領域がより圧迫され，性能の低いサーバ群に複製が作られたことに起因する．データアクセスの局所性の比較では，全てのケースで局所性がある場合の方がやや悪い性能を示した．

図 7は Central と Tier の平均応答時間の比較結果である．Tier では，最も性能のよかった OwnerComputes と LoadBound-Replication の結果を用いた．グラフの x 軸にはグリッドシステム上の総計算性能，y 軸には平均応答時間を示す．1.2[MSI95] は 2.8GHz Pentium4 プロセッサ約 10000 個分の性能と等価である．図中の三角で示した Tier（Total）は Tier で tier0，tier1，tier2 の計算性能の和を x 軸にとった場合の結果であり，図中正方形で示したものは Tier で Tier0 の計算性能を x 軸に取った場合の結果である．

図 4で示したように，Central は Tier0 サイトの総計算性能が高くなるにつれ性能が向上するが，計算要求に対して総計算性能が十分にないと応答時間が急激に増大し，0.453318[MSI95] で飽和する．よって，Central では十分な資源があればよい性能が期待できるが，電力的，空間的，経済的な要因等で 1 つのサイトに配置できる計算・ディスク資源に制限がある場合，性能低下が著しい．一方，Tier では Tier0 の性能が 0.6，0.5[MSI95] の場合のシステム全体の総計算性能はそれぞれ 2.28，1.9[MSI95] であり，Tier（Total）と Central を比較すると Tier でバックグラウンド複製手法を用いた場合でもその性能差は著しい．1 サイトに十分な計算性能があればシステムの安定性を維持してジョブを効率よく処理できるが，Tier では Central より各 Tier の総性能が低く構成できる．すなわち，Tier0 の性能が 0.4[MSI95] の場合のように，Tier0 の総性能が Central での処理限界より小さい場合も，OwnerComputes と LoadBound-Replication のように適切なスケジューリング・複製手法を用いていれば安定したジョブ処理が可能である．また，耐故障性の面でも，データの複製が複数のサイトに分散される方が好ましい．

7. 関 連 研 究

グリッドで公平な評価環境を提供するために，以下の 2 つのアプローチがある．

1つめのアプローチはグリッドエミュレーションである．MicroGrid[11] はグリッドエミュレータであり，クラスタ計算機上にグリッドのデファクトスタンダードであるツールキット Globus ベースの仮想グリッドシステムを構築する．現在のところディスク資源に対する仮想化のサポートはない．また，既存のアプリケーションやスケジューラ等の評価実験が可能であるが，大規模なシステムおよびアプリケーションを想定した評価実験には莫大な計算時間と計算資源，制御コストが必要となる．

2つめのアプローチはグリッドシミュレーションである．ここではディスクシミュレーションをサポートしている MONARC シミュレーションツール[5]，ChicSim[12] について触れる．

MONARC シミュレーションツールは Java で実装されたオブジェクト指向離散イベントシミュレータである．柔軟なシミュレーションのため，プロセス指向アプローチをとりスレッド化されたオブジェクトで構成されている．データモデルとして，HEP で一般に用いられているオブジェクトデータデザインである Objectivity/DB アーキテクチャを採用している．データベースサーバコンポーネントはデータベースへのオブジェクトのアクセスのため，クライアントーサーバメカニズムをシミュレートする．しかしながら，現時点では本シミュレーションツールを用いたスケジューリングや複製手法の評価実験は行われていない．

ChicSim も GriPhyN[4] プロジェクトにおいて CERN LHC 実験をターゲットにしたスケジューリング，複製手法のシミュレーションを行うためのシミュレーションツールである．ChicSim も離散イベントシミュレータであり、C ベースの並列シミュレーション言語 Parsec[13] で構築されている．文献[12] では，外部スケジューラ，ローカルスケジューラ，データセットスケジューラからなるデータグリッドシステムモデルを提案し，いくつかの外部スケジューラとデータセットスケジューラを組合せてその性能を調査している．シミュレートしたアプリケーションのジョブサイズ，データサイズが 2007 年に開始される LHC 実験よりも小さいものを想定（データサイズは 0.5-2GB）している．また，シミュレーションの間，オリジナルのデータセットは増加しない，アーキテクチャとしては従来のグリッドを想定している，すなわち Grid Datafarm が提案している計算とデータの融合させることを前提としていない点で本研究と異なる．

8. まとめと今後の課題

本稿では，Bricks グリッドシミュレータにデータグリッドシステムに対する拡張を行い，Grid Datafarm アーキテクチャを想定してデータグリッドシステムモデルとその性能についてシミュレーションで比較・評価した．評価では，単一サイトで集中的にデータ解析を行う Central モデルと MONARC の Tier モデルを比較し，本シミュレーションで想定した計算環境で現在想定されている LHC 実験ジョブ処理が可能であることを示した．また，十分な計算性能を保持できれば Central で効率よくデータグリッドジョブを処理可能であるが，1サイトの性能に制限がある場合でも，Tier モデルで適切なスケジューリング・複製手法を適用すれば，安定してジョブを処理することができることが分かった．

より効率的なスケジューリング・複製アルゴリズムを提案し，スケーラブルかつ様々な環境を想定した評価を行うことが今後の課題である．

参 考 文 献

1) Grid Datafarm: `http://datafarm.apgrid.org/`.
2) Tatebe, O., Morita, Y., Matsuoka, S., Soda, N. and Sekiguchi, S.: Grid Datafarm Architecture for Petascale Data Intensive Computing, *CCGrid2002*, pp. 102–110 (2002).
3) EU DataGrid: `http://www.eu-datagrid.org/`.
4) GriPhyN: `http://www.griphyn.org/`.
5) Aderholz, M. and et al: Models of Networked Analysis at Regional Centres for LHC Experiments, Monarc phase 2 report (2000).
6) Bricks: `http://ninf.is.titech.ac.jp/bricks/`.
7) Takefusa, A., Matsuoka, S., Nakada, H., Aida, K. and Nagashima, U.: Overview of a Performance Evaluation System for Global Computing Scheduling Algorithms, *Proc. of HPDC-8*, pp. 97–104 (1999).
8) Takefusa, A., Casanova, H., Matsuoka, S. and Berman, F.: A Study of Deadline Scheduling for Client-Server Systems on the Computational Grid, *Proc. of HPDC-10*, pp. 406–415 (2001).
9) Maheswaran, M., Ali, S., Siegel, H., Hensgen, D. and Freund, R.: Dynamic Mapping of a Class of Independent Tasks onto Heterogeneous Computing Systems, *Journal of Parallel and Distributed Computing*, Vol. 59, pp. 107–131 (1999).
10) Ranganathan, K. and Foster, I.: Identifying Dynamic Replication Strategies for a High Performance Data Grid, *Grid Computing* (2001).
11) Song, H. J., Liu, X., Jakobsen, D., Bhagwan, R., Zhang, X., Taura, K. and Chien, A.: The MicroGrid: a Scientific Tool for Modeling Computational Grids, *Proceedings of SC2000* (2000).
12) Ranganathan, K. and Foster, I.: Decoupling Computation and Data Scheduling in Distributed Data Intensive Applications, *Proceedings of HPDC-11*, pp. 352–358 (2002).
13) PARSEC: `http://pcl.cs.ucla.edu/projects/parsec/`.