2005－ARC－162 (3)
2005－HPC－101 (3)
2005／3／7

信頼性を考慮したキャンパスグリッド向け自律分散ファイルシステム

渡辺浩二[†]　松澤照男[††]　井口寧[†††]

従来のグリッドの分散ストレージシステムは，転送速度などの性能に特化しており，システムの信頼性や，ディスク利用効率についての議論はされていない．本研究では，一般のユーザー環境を使用して分散ストレージを構築する際に，個々のコンピュータの稼働率を測定する．この数値を元に冗長データの多重度と，データの分散先を動的に変化させる．冗長データにはリードソロモン符号を使用した多重パリティを使い，レプリカ方式よりも総データ量の削減を図る．この２つの手法によって，信頼性がばらばらな環境においてもシステムの信頼性の確保と，ディスクスペースの有効利用が可能な分散ファイルシステムについて提案を行う．

Autonomous distributed filesystem for Campus Grid that considers reliability

Koji Watanabe,[†] Teruo Matsuzawa [††] and Yasushi Inoguchi[††]

We propose the autonomous distributed file system that considers reliability and utilization efficiency of disk. We calculate the system reliability, and use Reed-Solomon code to avoid data loss by multiple disk node failure. The proposed system selects storage nodes that keep distributed data upon reliability of nodes automatically. The reliability of system is calculated by using availability of used nodes. The proposed system divides a file into data blocks and encodes some parity blocs. These created blocks are transferred to storage nodes. Using these methods, proposed system can keep high reliability and save disk space.

1. はじめに

近年，高エネルギー物理やヒトゲノム解析等の大規模データ解析を必要とする分野ではグリッド技術がキーテクロノジーとなっている．また，一般的な使用にはオーバースペックとも言えるPCの性能向上を受け，地球規模でのグリッドだけでなく企業や学校単位でのグリッド (キャンパスグリッド) にも注目が集まっている．キャンパスグリッドでは，各ユーザの利用スタイルやディスクの使用率も異なり，よりリソース管理とジョブスケジューリングが重要だといえる．特にストレージ容量は依然として高上昇率を保っており，世代交代の度に数十 GB ずつ記憶領域が増加している．このため，大きなディスク領域が余っているコンピュータも多数存在している．

grid 上では大量のデータが扱われるために，分散ファイルシステムを構築する手法が注目されている．グリッドを使用したデータストレージ分野での先行研究では，GridDatafarm[1] や Data reservoir[2] がある．これらの研究では，広域に分散したデータを高速に転送が可能で，数千キロ離れた場所への転送もすることができる．しかし，研究の主な対象がデータの保存，転送能力となっていてデータの保存性や信頼性についての議論がほとんどされていない．これは，先行研究でのグリッド構成に使用しているコンピュータが，もともと信頼性が高く，常時稼働が前提に運用されているために，詳しく信頼性を知る必要がないためと考えられる．それに対し，本研究のターゲットなるキャンパスグリッドでは，一般ユーザの使用しているコンピュータや，ネットワークでグリッドを構築するために，ダウンや再起動，ネットワーク障害などが頻繁に起きるので，詳細な信頼性計算が必要であるといえる．また，グリッドで一般的に使用されているレプリカ管理はオーバーヘッドが少なく，冗長性の確保，負荷分散には有利になるが，ディスク容量の利用効率の面で

† 北陸先端科学技術大学院大学 情報科学研究科
Graduate School of Information Science, Japan Advanced Institute of Science and Technology
†† 北陸先端科学技術大学院大学 情報科学センター
Center for Information Science, Japan Advanced Institute of Science and Technology
††† 北陸先端科学技術大学院大学 情報科学センター / 科学技術振興機構 さきがけ研究 21 (機能と構成)
Center for Information Science, Japan Advanced Institute of Science and Technology / PRESTO, Japan Science and Technology Agency

は不利になるといえる．

本研究ではグリッド上にデータを分散配置する際に，あらかじめ算出しておいた稼働率に応じてデータの配布先と，リードソロモン符号[3] で生成した多重パリティの冗長度を動的に変化させる分散ファイルシステムの構築，性能評価を行った．

以降，第 2 章でシステムの構成と実装について，第 3 章ではシステムの信頼性について，第 4 章では性能評価について，第 5 章にまとめを示す．

2. 提案システムの構成と実装

2.1 システムの構成

システムの構成は，ファイルの分割や多重パリティ計算を行うマスタサーバ，データを格納するストレージノード，各ノードのディスク空き容量や空きメモリ状況を把握する資源管理サーバである．各サービスが独立した状態の概念図を示す．

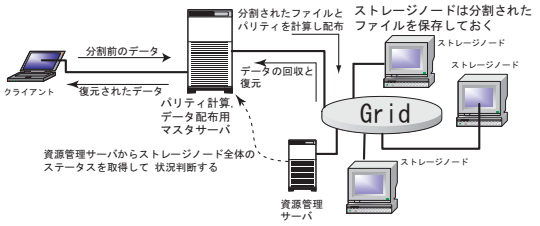

図 1 システムの概念図

マスタサーバ

マスタサーバは，主に次のような役割をする．

- ストレージノードの生存確認
- 資源情報問い合わせ
- リードソロモン符号のエンコード，デコード
- メタデータの読み書き

データの読み出し時には，GridFTP を使用してストレージノードに分散しているファイルを集めてきて，元のファイルに結合する．このとき，正常でないファイルや，消失してしまっているファイルがあればリードソロモン符号より復元して元のファイルに復元をする．

資源管理サーバ

資源管理サーバでは，信頼性計算と資源情報情報の登録を行う．資源情報の扱いは globus の MDS を使用して管理する．この情報をもとに，パリティの計算を行うノードや，ストレージホストのディスク空き容量を監視する．

ストレージノード

ストレージノードでは，GridFTP のサーバが待機しており，マスタサーバからの転送要求を待つ．また，ストレージノードはパリティの計算ホストになる場合もある．

2.2 システムの実装

実装に使用した言語は C と Perl を使用した．C では，リードソロモン符号や組み合わせ計算の処理に使用した．リードソロモン符号を生成するライブラリ[3] を元に，コードの一部 (ファイル入出力部分など) を改変し使用した．Perl ではジョブ生成用のスクリプト書き出し，ファイルの分散配置の準備，システム信頼性計算などの制御を行っている．

マスタサーバが一台だけでは，故障した場合や負荷の集中が起きることが予想される．このため，本システムでは 2 つの動作モードを使い分けることにした．一つは，マスタサーバが自分自身でパリティ生成などをすべて行う『自己処理モード』と，マスタサーバがグリッド内の他のマシンにパリティ生成などを依頼し，そこから GridFTP の第三者転送で分散配置を行う，『外部処理モード』がある． 外部処理モードは，マスタサーバが扱う元ファイルサイズと，空き物理メモリ容量を比較し，メモリ不足でスワップがおきそうな場合には，グリッドの中を検索し，メモリに空きのあるホストに処理を依頼する．この手法をとることで，パリティのオーバーヘッドを背負うホストが分散される．しかし，どのホストからもアクセスできるディレクトリに元ファイルをおいておく必要があり，現状では nfs でマウントされたディレクトリにファイルをおいておき，それを処理担当になったホストが読み出すことになっている．

3. システムの信頼性

本システムでは，システムの信頼性に応じて多重パリティの冗長度を変化させる．信頼性が高ければ多重度をさげ，冗長データを減らしオーバーヘッドも少なくて済む．システムの信頼性が低い場合には，多重度を上げてデータの保護を優先する．このシステムの信頼性算出の基となるのが，使用するストレージノードの稼働率である．個々のストレージの稼働率は，資源管理サーバが測定，算出を行う．資源管理サーバが一定時間ごとにストレージノードの生存確認を行うプログラムを実行し，順番にチェックする．uptime と downtime の積算状況を更新する．この情報より各ストレージノードの MTBF と MTTR を算出し，システムの信頼性 R_{system} は $R_{system} = \frac{MTBF}{MTBF+MTTR}$ として求める．また，システムの信頼性モデルは簡単に表現できるように m-out-of-n の並列システムとし

て計算した．ここで問題となるのが，それぞれ異なる稼働率のサブシステムから構成される並列システムの場合，故障の起きるパターンをすべて計算する必要があるために構成するサブシステムが多数になった場合 (数百台の規模であっても) に，計算の爆発が起きてしまう．この問題の解決方法として，ある程度の稼働率でクラス分けをし，その代表値に統一してシステム信頼性を計算をする．代表の稼働率が a ならば，

$$R_{system} = \sum_{i=0}^{n-m} ({}_nC_{n-i} \cdot a^{n-i} \cdot (1-a)^i)$$

と簡素な式で計算することが可能となる．

クラス分けには稼働率に応じて３つのクラスと，使用不可のクラスに分ける．クラス分けはノードの稼働率 R_{node} を基に行う．

- $99.99\% \leq R_{node}$ (Highest priority として使用)
- $99.9\% \leq R_{node} < 99.99\%$ (Higher priority として使用)
- $99\% \leq R_{node} < 99.9\%$ (Normal priority として使用)
- $R_{node} < 99\%$ (稼働率が上がるまで使用しない)

このチェックのフローチャートを図 2 に示す．

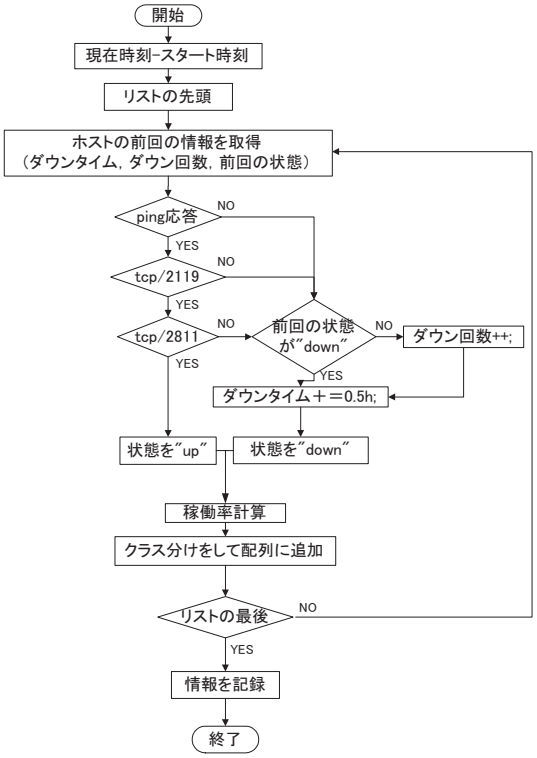

図 2 稼働率チェックのフローチャート (30 分ごとに監視する場合)

データの書き込み時にマスタサーバは，クラス分けされたストレージノードを信頼性の高いクラスから順番に使用する．使用ストレージノードの偏りを防ぐために，プログラムの開始時にクラス内の順番をシャッフルして利用する．そして，使用するノードの分だけ配列から一つずつチェックしながら取り出す．このとき，データの分割サイズは 400MB ずつに分割し，多重パリティの冗長度を２と設定して信頼性計算を行う．もし，要求する信頼性を満たせない場合には，冗長度を１上げてシステムの信頼性計算を再実行する．使用する予定であったクラスで，ストレージノードが不足した場合には，次の信頼性クラスをしようすることにして，再度ストレージノードの選択，システム信頼性計算を実行する．システム全体の信頼性設定は 99.999% とした．

4. システムの性能評価

4.1 実験システムの構成

実験では表 1 に示した機器を使用した． Sun Blade1500 は 8 台使用した．各コンピュターは学内 LAN を通じて GigabitEthernet で接続されている．また，これらのコンピュータ，ネットワークは通常のユーザー環境で行われ，オーナーは通常通りにネットワークやコンピュータを使用している状況で実験を行った．

表 1 使用機器のスペック

機種	Sun Blade 1500	自作
CPU	Ultra SPARC IIIi	Opteron
	1GHz	2.2GHz * 2
メモリ	8GB	512MB
OS	Solaris 8	SuSE Linux 9.0 Pro
NIC	GbEthernet	GbEthernet

また，各種コンピュータが搭載している HDD のアクセス性能は表 2 に示す．このベンチマークは Bonnie[4] を使用した． まず予備実験として，データのストライピング転送における IO 性能向上効果について調査をした． この結果より，GigabitEthernet の LAN 内での転送速度は 740Mbps 程度まではスケーラビリティが確保されることがわかった．netperf での帯域

表 2 使用した HDD の性能

	Blade1500 (RAID なし)	自作 (RAID1)
read	34	33
write	26.9	30

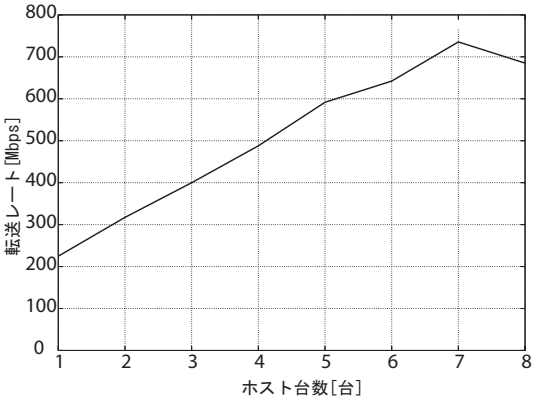

図 3 ホスト数と転送速度の関係

測定においても 750〜800Mbps 程度が上限であったことから，リーズナブルな値である．８並列の場合の性能低下は，ディスクのランダムアクセス性能の限界と転送チャンネル同士の帯域の取り合いなどが要因で起きたものと考えられる．解決方法としては，並列転送の帯域合計が 700Mbps 程度になるように帯域幅を絞ることが考えられる．

4.2 自己処理モードでの性能評価

グリッドを構成し，自作 PC(Opteron*2) における自己処理モードでの性能評価を行った．自己処理モードでは，ローカルディレクトリに保存してあるファイルをグリッドへと分散配置する．プログラム中の，エンコード処理担当ホストの選択プロシージャを無効にして，処理ホストを localhost に固定して使用した．

4.2.1 データの書き込み

データの書き込み時の処理は，配布先ストレージノードの決定，ファイルの分割とリードソロモン符号の生成，GridFTP によるデータの転送を行う．実験結果を図 4 に示す．凡例の『ストレージノード決定』は信頼性計算と生存確認を行い配布先ストレージノードを決定する時間,『データ処理』は元データの読込みから多重パリティの生成と書き出し,『データ転送』は grid への転送の時間である．これは以下の実験でも同様のものとする．実験結果では，ストレージノードの決定は，どの場合でもほぼ一定の時間で終わることがわかる．データの処理には 125MB から 500MB のいずれも 30 秒程度の時間がかかっている．これはデータ処理のジョブ終了検出を globus-job-status がジョブ完了を検出する時間に誤差があるためだと考えられる．ジョブ検出時間自体の誤差と，コマンド実行タイミングによって 10 秒から 20 秒程度の遅れが発生する場合がある．

データ転送は，データ量に対して比例して長くなる．

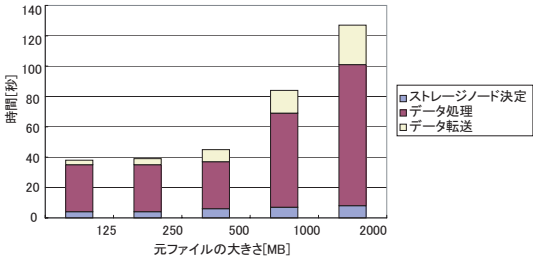

図 4 自己モードでの書き込み性能
(元ファイルの大きさと処理時間の関係)

ファイルサイズに対して転送時間の増加の割合が鈍いのは，転送ファイルサイズが小さいと GridFTP の転送速度があがりきる前に転送が終了するからだと考えられる．

4.2.2 データ読み出し

先の実験で grid 中のストレージノードへと分散配置したファイルを GridFTP でコピーし，元のファイルを復元する実験を行った．読み出し時には，分散配置したファイルが揃っている場合と，grid に異常がある場合について実験を行った．

4.2.2.1 故障が存在しないとき

まず，実験では故障がなく，システムが健全な場合の読み出しについて実験を行った．ファイルサイズは 125MB，250MB,500MB, 1000MB, 2000MB の５つである．これらのファイルのデータブロック，パリティブロックをストレージノードから回収し，もとのファイルを復元した．故障がない場合には，リードソロモン符号の復号は必要ないため，単純に回収したデータブロックを結合するだけでよい．実験結果を図 5 に示す．凡例は先の例に加え,『結合・復元処理』が加わっている．これは，パリティのデコードと書き出し，正常なデータとの結合の時間を含む．

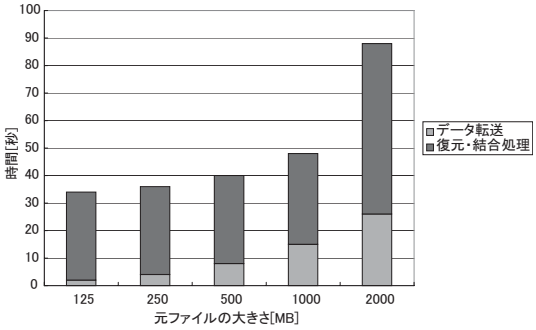

図 5 ファイルの読み出し，復元

この結果では，予備実験でおこなったデコーダの性能とほぼ一致する結果が得られた．データが正しく復

元されているかは，md5sum コマンドでハッシュ値を計算し，元ファイルのハッシュと比較し，同じことを確認した．

4.2.2.2 縮退運転時

次に，ストレージノードに異常が発生した場合について実験を行った．

ファイルの種類は同様に５種類のファイルを使用した．まずこれらのファイルを使用して，復元可能な故障数までファイルの断片を消去し，読み出しのコマンドを実行した．実験結果を図 6 に示す．

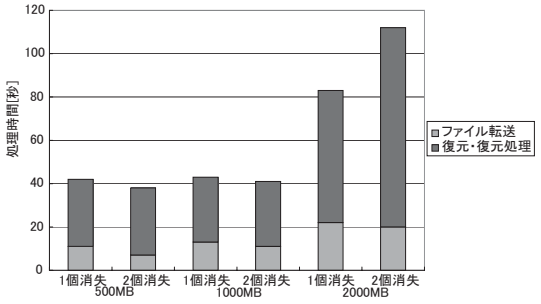

図 6 縮退運転時の性能

先の実験と同様に実行時間が 30 秒以下の場合には，正しくジョブ終了が検出されずに，約 30 秒と検出されてしまっている．grid へと分散配置しておいたファイルが消失してしまっている場合，存在しているファイルだけ転送されるために，転送時間は短くなることが実証された．しかし，消失ファイルが多ければ再生成すべき演算量が増えるため，全体の処理時間は増加してしまうことがわかった．ここでも md5sum コマンドによるハッシュ値の確認を行い正しいデータが再生できていることを確認した．

4.3 外部処理モードでの性能

この節では外部処理モードでの性能を計測した．外部処理モード用の関数を作成し，プログラムに組み込んだ．

4.3.1 データ書き込み

データの書き込み時には，書き込むファイルサイズとプログラム実行マシンの空きメモリを比較しどのホストでリードソロモン符号を生成するかを決める．結果を図 7 に示す．

結果のグラフをみると，分割処理が自己処理モードに比べて大幅に増えていることがわかる．これは，ファイルの読み込み時間が大幅に増えているからだと考えられる．外部処理モードでは，リードソロモン符号の処理ホストとなる可能性のあるホスト全てからアクセス出来る必要があるために，NFS でマウントされたディレクトリに元ファイルを置いている．このために NFS の性能がボトルネックとなり，分割処理が長く時間がかかった．

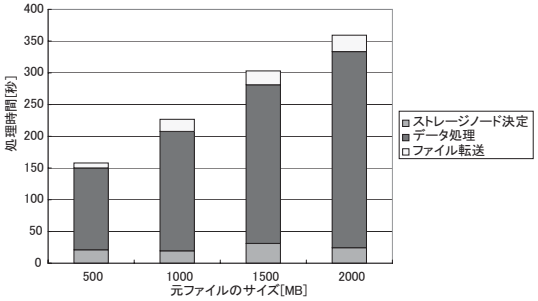

図 7 外部処理モードにおけるファイルサイズと書き込み時間の関係

4.3.2 データ読み出し

次に，grid へ分散配置されているデータを収集し，元のデータを復元する処理を外部処理モードで行った．

4.3.2.1 故障が存在しないとき

この節では，分散配布したファイルが完全に揃っている状況でテストを行った．復元したファイルは nfs 経由でマウントされているサーバーに書き出した．結果のグラフを図 8 に示す．

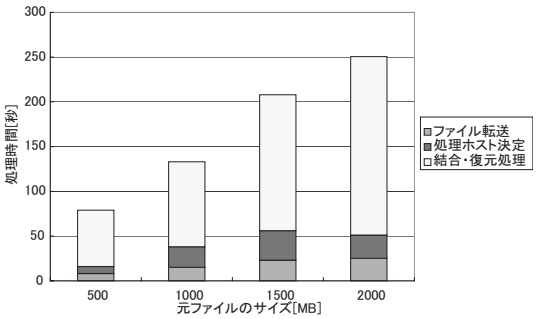

図 8 外部処理モードにおけるファイルサイズと読み出し時間の関係

結果を見ると，GridFTP によるデータ収集の時間と，ファイルの結合復元処理はファイルサイズに比例して長くなることがわかる．nfs の性能がボトルネックとなり全体の処理時間は長くなっている．ファイルサイズの違いによる処理時間の差は予備実験とほぼ同様の結果となり，予測される値に近い結果を得ることができた．

4.3.2.2 縮退運転時

縮退運転では，分散配置をしたファイルを意図的に消去して，正しくファイルが復元出来ているか，その処理にかかる時間について測定を行った．結果を図 9

に示す.

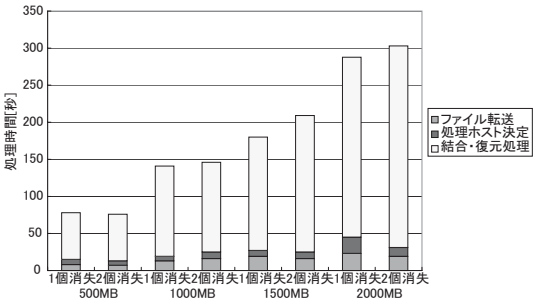

図 9 外部処理モードにおける縮退運転時の性能

縮退運転では，分散配置してあるファイルがいくつか消失しているために，GridFTP によるファイル転送の際の総転送容量は減っていることになるはずである．結果のグラフでは，消失ファイル数が多ければ転送時間が短くなっていることが確認出来た．ここでも，ファイルサイズが小さい場合には，１つのファイルをリードソロモン符号を使用して復元する時間が 5 秒から 10 秒程度と短い．この程度の処理時間の差は，globus がジョブ終了を検出する際の大きなタイムラグによって吸収されてしまう．よって 500MB，1000MB のファイルを復元する際に同じような処理時間の結果となっている箇所があるが，正常な結果だと考えられる．

4.4 データの保守，再生成

いくつかのファイルを grid へ分散配置し，そのファイルたちの健全性を検証するプログラムを作成した．ファイル断片がなくなっていたり，データを保存しているストレージノードが保存時と異なる信頼度クラスに属している場合などにデータを復元し，再度分散配置しなおすようにしてある．

データの確認にかかる時間は，元ファイル１つにつき 15 秒から 20 秒程度の時間を要する．ファイルに異常がある場合には，続いてデータのデコードから再エンコードして分散配置するための時間が必要になる．

4.5 記憶領域利用効率

本研究のシステムでは，ストライピングと多重パリティを使用することで，耐障害性確保と負荷分散を行っている．耐障害性とディスク利用効率を考えた場合，１つのレプリカデータを作成すると，元データと同じだけのディスク容量が必要となる．使用可能なディスク容量を基準にして考えた場合の比較を表 4.5 に示す．

この表には計算例の一部だけを掲載したが，レプリカ方式にくらべて多重パリティ方式を採用したほうが，ディスク使用効率がよいことがわかる．使用するディスク台数が多ければ，ディスク利用効率の更なる向上をすることが可能である．

表 3 ディスク利用効率の比較

	レプリカ	パリティ1 個	パリティ2 個
4 台 15 台使用時に利用可能な%	50(%)	75〜93(%)	50〜87(%)
冗長度	二重化	N+1	N+2

5. おわりに

本論文では，一般ユーザ環境でキャンパスグリッドを構築する際の信頼性に着目し，構成ノードの稼働率に応じて動的にパリティの多重度とデータ分散先を変化させる分散ストレージシステムについて述べた．本システムでは，いくつかのデータを消去したのちデータの再生が可能なことを確かめた．また，実際の運用では偶発的な障害も起きたがデータの再生成プログラムによって再生されたデータが，異なるノードへと再配置され無事であった．

今後の課題としては，一時ファイルを使用せずに処理の高速化を行うことや，データ自体の保全性 (Mean Time to Data Loss) を算出し，より詳細なモデルを構築してシステム全体の信頼性を上げるための改良に取り組みたい．

参 考 文 献

1) 建部 修見，森田 洋平，松岡 聡，関口 智嗣，曽田 哲之，「ペタスケール広域分散データ解析のための Grid Datafarm アーキテクチャ」，ハイパフォーマンスコンピューティングと計算科学シンポジウム HPCS2002 論文集，pp.89-96，2002 年 1 月
2) Hiraki, K. Inaba, M., Tamatsukuri, J., Kurusu R., Ikuta, Y., Hisashi, K. and Jinzaki, A., “ Data Reservoir: Utilization of Multi-Gigabit Backbone Network for Data-Intensive Research, ”to appear Proc. SC2002(CDROM), 2002.
3) GFLIB - C Procedures for Galois Field Arithmetic and Reed-Solomon Coding, http://www.cs.utk.edu/~plank/plank/gflib/
4) Bonnie, http://www.textuality.com/bonnie/