Vol.2009-HPC-122 No.8
2009/10/9

マルチグリッド環境における効率的な監視システムに関する研究

小山 敦[†1] 長坂 康史[†2] 渡瀬 芳行[†3] 佐々木 節[†3]
岩井 剛[†3] 佐藤 三久[†1,†4] 建部 修見[†1,†4]

現在のグリッドシステムでは膨大な計算資源を扱う．これら計算資源は広域に分散しており，全ての把握が困難である場合が多い．グリッドシステムの拡充に伴い，これらをより安全に運用し，システムの可用性を高めることが重要であるが，ミドルウェアやアプリケーションの改善といったユーザサイドからの努力には限界がある．計算機環境全体を監視する独立したシステムを用いて保守を行うことでロバストで可用性の高いグリッド環境の構築が可能である．我々は先行研究[1)]にて gLite を対象とした監視において有効性が確認された監視システムの対象を現在のグリッド規模に合わせて複数のグリッドミドルウェアの監視に拡大し，そのフレームワークを提案する．現在日本で運用されているマルチグリッド環境である KEK(高エネルギー加速器研究機構)および広島工業大学のグリッド環境を利用し，監視システムの実装・性能評価を行った．本稿ではフレームワークおよびシステムの概要を述べ，性能評価について報告する．

A Study on Efficient Monitoring System in Multi-Grid Environment

Tsutomu Koyama[†1] Yasushi Nagasaka[†2]
Yoshiyuki Watase[†3] Takeshi Sasaki[†3]
Go Iwai[†3] Mitsuhisa Sato[†1,†4] and Osamu Tatebe[†1,†4]

A present grid system treats a huge calculation resource. As for these calculation resources, it mostly distributes to the large area, and the entire grasp is difficult. There is a limit in the effort from the user side of improvement of the middleware and the application though it is important to operate these more safely along with the expansion of the grid system, and to improve the availability of the system. Grid environments with high availability can be constructed robustly by using and maintaining an independent system that observes the entire environment computer. We expand the object of the monitoring system that confirmed effectiveness by the previous work when gLite was observed, to a present grid scale observing two or more grid middlewares, and propose the framework. Mounting and the performance assessment of the monitoring system were done by using grid environments of KEK(High Energy Accelerator Research Organization) and Hiroshima Institute of Technology that are the multi grid environments operated in Japan now. The framework and the outline of the system were described in this text, and we report on the performance assessment.

1. はじめに

近年，グリッド・コンピューティング技術を用いて大規模計算機環境を構築し，そのリソースを用いることによって，様々な研究が活発に行われている．グリッド・コンピューティングを用いることによって，大量のリソースを仮想化・集約し，スーパーコンピュータに匹敵する様な計算能力を実現することが出来る．この大量のリソースを利用する研究分野としては，現在，ヒトゲノムにおける生物情報解析や，天文学における天体情報解析，高速加速器実験における物理解析等が挙げられる．これらの研究では大量のデータを扱い，それを用いた大規模な計算処理を行わなければならないため，大規模な計算リソースを低コストで用意できるグリッド･コンピューティングの利用が有効であるとされ，実用化のための研究が行われている．また，企業においては，災害等により組織内の機器が故障した場合，速やかに別の機器へ接続を切り替えることにより，業務の中断を回避することを目的として，グリッド・コンピューティングによる自律的なリソース管理機能を用いた耐久性の高い業務システムの実用化が進められている．このように，グリッド・コンピューティングはシステムとして多方面にわたり導入あるいは検討がなされている[2)]．

グリッドシステムはその構造上，多数の広域に分散された計算資源を必要とする．並列分散処理を用いた場合でも，グリッド上の計算機の性能が低下した場合，システム全体の処理性能に影響を及ぼす．また，ソフトウェアだけでは回避が困難なシステムのフリーズ，ハードウェアの故障等からの速やかな復旧のためにはグリッドシステムとは独立した汎用型監視システムの導入が有用である．これらの汎用型監視システムではロードアベレージ等のマシン内部の情報をはじめ，グリッドシステム外部の計算機を監視することも可能であり，用途に合わせて逐次新たな監視機能を追加することができるほか，グリッドコマンド等の実行も可能であるなど拡張性の高いものとなっているため，最近のマルチグリッド環境に適している[3)]．

一方でこれら汎用監視システムはそのポータブルな構造のため，負荷が増加する傾

[†1] 筑波大学大学院 システム情報工学研究科
Graduate School of Systems and Information Engineering, University of Tsukuba
[†2] 広島工業大学大学院 工学系研究科
Graduate School of Engineering, University of Hiroshima Institute of Technology
[†3] 高エネルギー加速器研究機構
High Energy Accelerator Research Organization
[†4] 筑波大学 計算科学研究センター
Center for Computational Sciences, University of Tsukuba

向にある．マシン環境によっては監視システムがリソースを消費してしまい，グリッドシステムの処理に支障をきたす可能性もある．

本研究では国内に存在するマルチグリッド環境上で動作する監視システムのためのフレームワークを提案する．さらにシステムの実装・性能評価を行い，実際のマルチグリッド環境で実用可能かどうかを検証する．本稿では提案する監視システムフレームワークについて述べ，その実装および性能評価について報告する．

2. マルチグリッド環境の監視

現在のグリッド環境が 1 章で述べたとおり拡大しているため，新たな形態の監視システムが求められる．先行研究では KEK と広島工業大学間でグリッドミドルウェア gLite を実装したグリッド環境を構築し，それを構成する全ホストを監視するシステムのためのフレームワークを構築し，実装および性能評価を行った．それによれば実運用に耐えうる安定性を備え，グリッド環境の監視に有用であることを確認している．我々は異種のグリッドミドルウェアを実装した複数の組織が連携して計算資源を共有する環境をマルチグリッド環境と呼ぶ.本研究では gLite に対する監視システムから，マルチグリッド環境のための監視システムへと焦点を移し，そのためのフレームワークを提案する．

2.1 フレームワークに必要な機能

本章では監視システムのフレームワークを提案するにあたり，監視システムにおいて必要となる機能について考察し，以下の 5 つを挙げた．

(1) 監視機能

グリッドシステムを監視するためには，広域に分散された計算機の情報を同時に見ることができなければならない．したがって監視システムではネットワーク上での各計算機のパフォーマンスおよび状態異常をリアルタイムで監視する．

(2) 診断機能

グリッドシステムでは多数の計算機を一度に監視するため，そのデータも大量である．それらのデータを確実に検査するため診断機能を設ける．閾値などの条件によって自動的にマシンの状態を判断させることができる．

(3) 通知機能

グリッドシステムに異常が発生した場合およびマシンのパフォーマンスが評価基準による異状条件を満たした場合にユーザに向けて情報を通知する機能が必要である．通知機能によってユーザはリアルタイムの監視から解放され利便性が向上する．

(4) 可視化機能

簡単な通知機能では音と文字によってユーザに異常を知らせることができるが，そのどちらも状況によっては確認が困難なこともある上，大量のデータから必要な情報を確認するには不向きである．そこで画像，あるいはグラフ等の統計データを生成し提示することで，ユーザのより迅速な理解を助ける．

(5) 記録機能

監視システムを導入する大きな利点の一つにデータの保存がある.監視データ自体，あるいは診断記録，通知記録をデータベースに格納することによって計算機環境全体の動作傾向を把握することができ,グリッドシステムの管理に役立てることができる．

3. 提案フレームワーク

3.1 概要

2 章で述べた機能に基づき，グリッド環境を監視するシステムのためのフレームワークを検討した．図 1 に本フレームワークの構成を示す．本フレームワークは監視，診断，通知，可視化，記録，管理の 6 つの機能からなり，全体として一つのアプリケーションとして動作する．図 1 では各機能がそれぞれ隣接する機能に対してデータあるいは処理の流れを持つことを示している．図 1 について概要を説明する．管理機能はグリッド環境に関する情報を持ち，監視機能はグリッド環境からデータを取り込み記録，診断，可視化機能へ渡す．診断機能は監視機能からのデータを通知，記録，可視化機能へ渡す．通知機能は管理機能へデータを渡す．可視化機能は可視化データを管理機能へ渡す．記録機能は監視，診断データをデータベースに格納する．最終的に管理機能が Web ブラウザ上で全てのデータを公開するという流れになる．

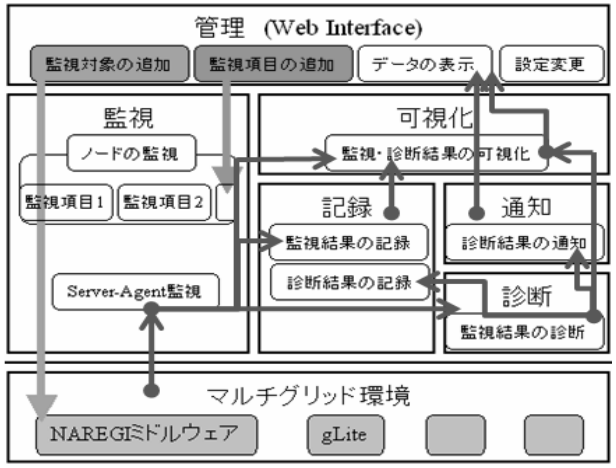

図 1 監視システムのフレームワーク

3.2 フレームワークの機能

3.2.1 監視機能

本研究では単一ではなく複数のミドルウェアを監視できるポータブルなシステムを目標としているため，サーバ・ホスト間の通信には Server-Agent 通信を用いた．図 2 に概要を示す．グリッド環境では PKI が用いられるため，セキュリティが厳しく外部からのプッシュタイプの命令を容易には受け付けない．グリッド環境におけるリソースの需要増大にスムーズに対応するためにはあらかじめオールインワンで機能を詰め込んだエージェントプログラムを配置するのが最も効率が良い．また，本研究では監視のための設定に XML を用いた．XML はスケーラブルかつポータブルな特性があるため，本フレームワークに最適であると考えられる．

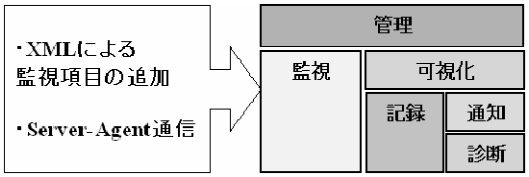

図 2　監視機能

3.2.2 診断機能

診断機能では Agent から送信されるデータを閾値に基づいて診断する．設定には XML を使用する図 3 に概要を示す.

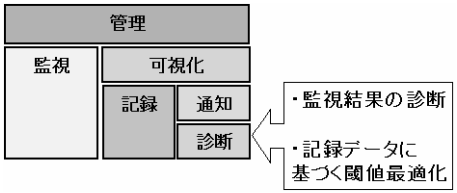

図 3　診断機能

3.2.3 通知機能

クライアントマシンにおける異常発生，監視データの検査情報を通知する．図 4 に概要を示す．宛先アドレスや通知法などの設定は XML で行う.

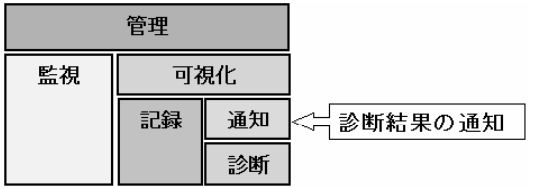

図 4　通知機能

3.2.4 可視化機能

通常の監視及び診断結果のグラフィックによる識別表示を行う. 図 5 に概要を示す. また XML による設定を用いて監視結果および診断結果をグラフとして表示する.

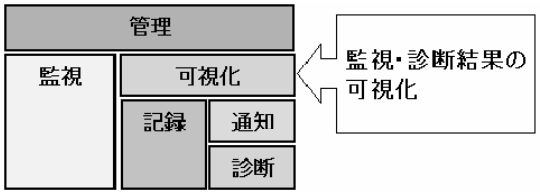

図 5　可視化機能

3.2.1 記録機能

監視・診断機能から送られたデータをデータベースへ格納する. 図 6 に概要を示す. また、診断結果の通知が行われた際に通知記録もデータベースへ保存される。このデータベースは可視化機能へデータを提供する。

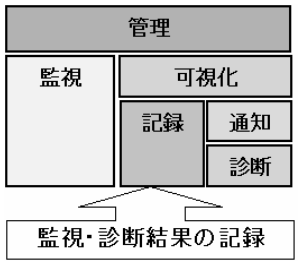

図 6　記録機能

3.2.2 管理機能

管理機能では全機能の表示領域の出力および設定を行う．図 7 に概要を示す．全ての設定は管理ウィンドウ上で行われる. アプリケーションにはポータブルな Web ブラウザを利用する.

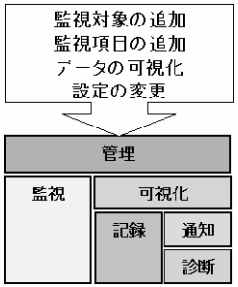

図 7　管理機能

4. フレームワークの実装

本研究では提案フレームワークを実装するに際し，開発コスト，運用実績，スクリプトの充実の観点から，既存の監視ツールを使用した．本稿で使用した監視ツールを表 1(a)に示す．(b)は後述する先行研究において使用されたツールである．

表 1　監視ツール

監視ツール(a)	ZABBIX 1.6
監視ツール(b)	Nagios 3.0a4

ZABBIX[4]は ZABBIX SIA が開発を進める監視ツールで，一般的な監視ツールの機能を実現できる．また標準で XML による設定機能が付属している．そのため ZABBIX を拡張することで本フレームワークの機能を概ね備えられると考え，ZABBIX 上にグリッドシステムのための監視スクリプトを開発し導入することで，提案フレームワークを実装した．

4.1 プラグイン

本研究で開発した NAREGI のための監視用プラグインについて述べる．NAREGI は国立情報学研究所を中心として開発されているグリッドミドルウェアである[5]．

(a)　naregi_gridftp_check.c

NAREGI のファイル転送機能である GridFTP の状態確認を行う．また，データ送信時の RTT(Round Trip Time)を計測する．

(b)　naregi_port_check.c

NAREGI のコンポーネントのポートを監視し，アプリケーションの状態を確認する．

(c)　naregi_gmpi_check.c

NAREGI の MPI 機能の状態を監視する．

4.2 プラグイン以外の追加機能

ZABBIX はデータベースに情報を格納しているが一定間隔の値しか表示できないため，前データの一覧表示を行うログビューア機能を追加した．実行画面を図 8 に示す．図 8 では時刻をキーとしたデータの検索を行っており，監視時のホスト数分の検索結果が表示されている．

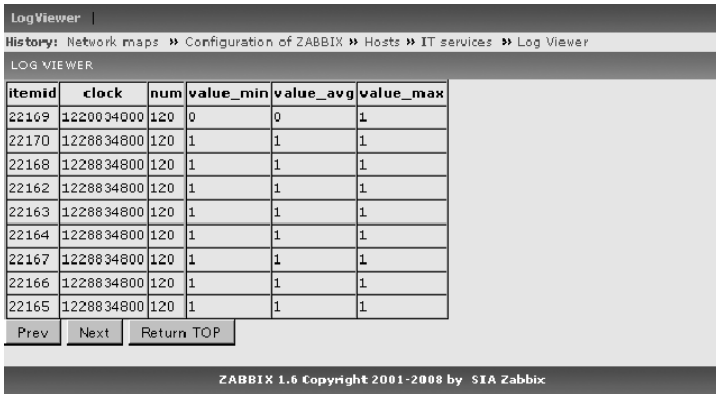

図 8　視覚化機能の追加

5. 性能評価

本章では提案フレームワークを実装したシステムの性能評価について述べる．すでに先行研究ではミドルウェア gLite[6]のための広域グリッド環境監視システムのフレームワークを提案・実装し，その有効性が確認されているが，本研究では複数のグリッドミドルウェアが存在するマルチグリッド環境上で監視システムを実装し，性能について評価した．計算機の構成を図 9 に示す．HIT(広島工業大学）と KEK(高エネルギー加速器研究所)にある NAREGI，gLite マシンを用いた．HIT 側では新たに NAREGIver1.1 をインストールした．両組織のホストは全て NAREGI，gLite それぞれで共通の VO(仮想組織）に所属し，グリッドジョブをやり取りすることができる．HIT では全てが表 2 に示す同一の諸元のマシンを用いて環境を構築している．これらのマシン環境を用いて性能評価を行った．実験では監視システムのサーバは全て HIT 側に置かれた．

表 2　HIT における実験環境

NAREGI	
OS	ScientificLinux 3.0.9
CPU	Pentium4 2.8GHz
Memory	512MB
Kernel	Linux-2.4.34

gLite	
OS	ScientificLinux 3.0.9
CPU	Xeon 3.06GHz
Memory	1024MB
Kernel	Linux-2.6.9-89.0.9.EL

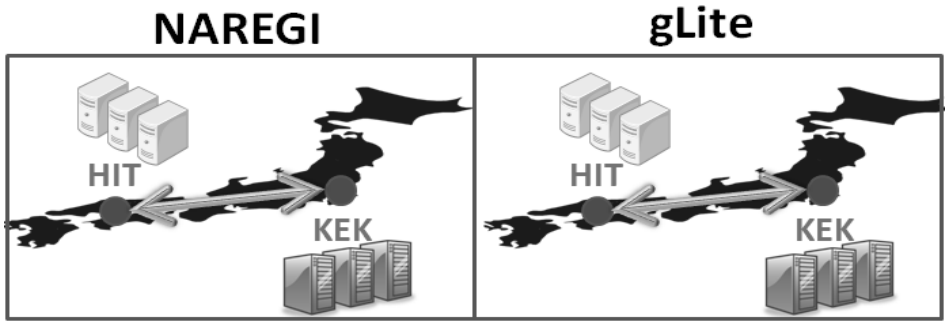

図 9　計算機の構成

5.1 一般的監視項目

一般的な監視項目に関して性能評価を行うため，2 週間システムを連続して稼働し可用性を検証した．評価環境のマシン構成は図 9 にあるマシンを全て利用し，計 42 台にたいして監視を行った．ただし KEK のマシンに関してはセキュリティの関係から全ての項目を実施することはできなかった．評価における一般的な監視項目は，ping による応答監視，port チェックによるアプリケーションの状態監視，CPU のロードアベレージ，OS のバージョン確認の 4 つである．それぞれのタスクは比較的処理が軽いと考えられることや，ping はネットワーク，port はアプリケーションの状態を監視するために必須であることや，CPU 使用率，バージョン確認はリソース管理に欠かせないと考えら得ることからこの 4 つを選択した．この実験では一度もシステムが停止することはなかった．これらの項目では比較的負荷は小さく，CPU への負荷が少ないものと思われる．これらの監視項目は今回の実験中常に動かしていたものであり，監視および診断データはすべてデータベースへ格納される．

5.2 GridFTP による転送スループット

次にプラグインの性能および負荷について評価する．実験では図 11 のような開発システムを用いて GridFTP によるファイル転送を行い，そのスループットを求めた．用いたマシンは HIT の NAREG 環境上にある GVMS(Grid Virtual Machine Scheduler) と GVME(Grid Virtual Machine Engine)，実行を制御及び監視するサーバからなる．評価ではシステムの通常監視機能を用いて 10～100MB のファイルを各 10 回ずつ送受信するテストを行い，その処理時間を計測した．測定結果は表 3 および図 10 に示すように，各ファイルにおける処理全体の時間に占めるプラグインの割合が極めて小さくなっており，開発した監視プラグインがシステムに及ぼす負荷が少ないことを示している．一方で NAREGI の処理時間が長すぎることに関しては，NAREGI 自体の GVMS における処理端末選択の計算にかかる時間のほか，システム自体がマシンリソースを占有しているためと思われる．評価の過程で何度かシステムがフリーズする現象が見られ，監視システム，ミドルウェア両方が大きくリソースを消費していたものと思われる．

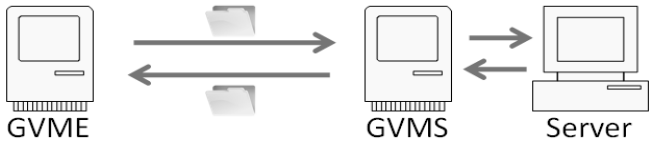

図 11　性能評価におけるマシン構成

表 3　GridFTP による送信時間(a)

容量	10.00	20.00	30.00	40.00	50.00	60.00	70.00	80.00
plugin	2.33	4.38	7.44	9.23	11.99	14.26	16.83	17.97
NAREGI	0.04	0.02	0.10	0.02	0.08	0.02	0.06	0.06

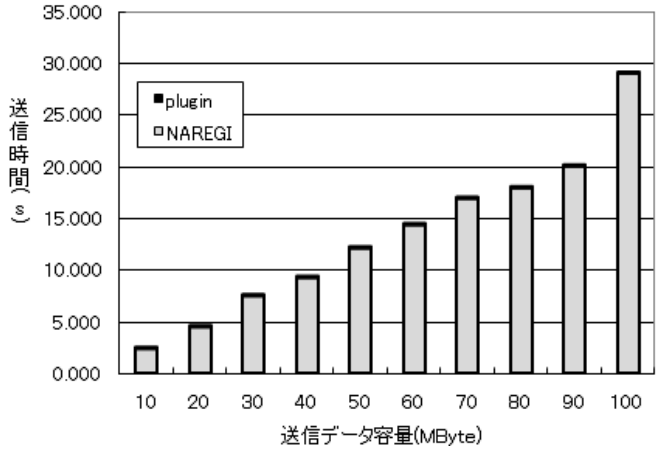

図 10　GridFTP による送信時間(b)

5.3 DB クエリを用いた比較評価

次に先行研究における自動監視システムと本システムの性能を比較評価するため，それぞれデータベースクエリを実行し，応答が帰ってくるまでの経過時間の測定を行った．本システムではエラーにより gLite ベースのプラグインによる計測ができなかったため，NAREGI を用いた性能測定を行った．先行研究では監視ツールとして Nagios(表 1(b)参照)を用いてシステムを実装している．Nagios は Ethan Galstad によって開発されたプラグインタイプの監視ツールであり，機能の拡張が容易である．ZABBIX との相違点は，テキストを用いて設定を行うこと，Server-Agent 通信ではなくプッシュ型通信であることなどがある．評価に際し，gLite のクエリ送信用にプラグインを用意し測定を行った結果が図 11 である．ミドルウェアが異なることから比較することは難しいが，先行研究の結果に対し本システムでは少し乱れが見られるものの 100ms 未満のずれであることから本システムが実用上問題ない性能を有するものと考えられる．しかし，一般的項目に関する性能に関しては概ね本システムが良好な値を示した．これはミドルウェアの差異以前に，Nagios がクエリの実行と同時に様々な処理を行っているため，結果として性能が落ちると考えられる．本システムではプラグインの処理はその他のフレームワークの機能と分離しているため，その分効率が増しているものと思われる．また，本評価においても数度システムがフリーズした．したがって今後の発展の可能性としては，現在のマシンをより性能の高いものにするか，あるいは汎用監視ツールではなくグリッド環境に適したシステムを開発することなどが考えられる．

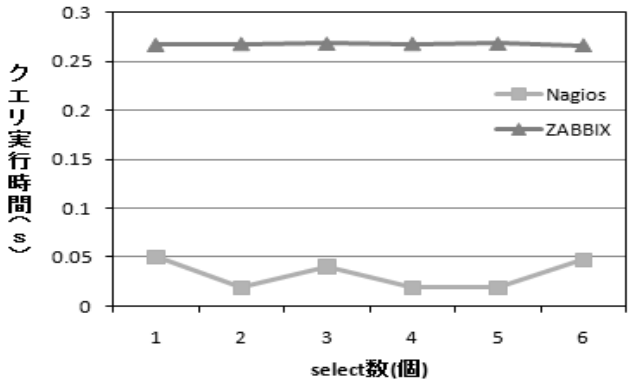

図 11　DB クエリ実行時間

6.　おわりに

本稿では，先行研究にて提案され有効性が検証された gLite グリッド環境のための監視システムから，新たにマルチグリッド環境に適応させたシステムのためのフレームワークを提案した．さらにその性能を評価するための実装では，フレームワークの機能のうち幾つかを既に実現していること，また実際に運用する際の容易さを考慮して既存の監視ツールである ZABBIX を用いた．また ZABBIX には監視や可視化機能などにおいて不十分な点があったため，プラグインや新たな機能などを追加し利便性を向上させた．

性能評価では実際のグリッド環境を利用するため広島工業大学および高エネルギー加速器研究所のグリッドマシン環境を用いてシステムを実装した．一般的な項目に関する監視においてはフリーズすることなく稼働し続け，可用性を示した．また，NAREGI の GVMS, GVME 間での GridFTP によるファイル転送時間に対する評価では監視システムの処理そのものによる負荷が全体に比べて小さく，システムの有用性を確認できたと同時に，グリッドコマンドによる処理の重さ，あるいは大量の監視パケットおよびデータベースの書き込み等によってマシンの能力を上回るタスクが発生した場合に問題があることが分かった．DB クエリによるスループットの評価では，先行研究の実験環境を利用し，本システムとの性能差を検証し，先行研究と比較して十分実用可能な安定性を持つことを確認した．今後の課題としてはロバスト性の確保などがある．

参考文献

1) 藤井峰夫“加速器科学仮想組織におけるグリッド環境自動監視・診断システム”，FIT2007, pp5-6，一般講演論文集 第 4 分冊
2) I. Foster, C. Kesselman, S. Tuecke “The Anatomy of the Grid: Enabling Scalable Virtual Organizations” International Journal of High Performance Computing Applications, Vol.15, No. 3(2001), pp200-222
3) 釘井睦和，廣安知之，三木光範，谷口義樹“P2P フレームワークに対応したグリッド環境のためのモニタリングインターフェース”，研究報告-ハイパフォーマンスコンピューティング（2004-HPC-099）Vol.2004，No.81，2004
4) ZABBIX: ZABBIX SIA
5) K. Miura “Overview of Japanese science Grid project NAREGI” Progress in informatics, Vol.3(2006), pp67-75
6) E. Laure, S. M. FisherA. Frohner,C. Grandi,P. Kunszt,A. Krenek,O. Mulmo,F. Pacini,F. Prelz,J. White,M. Barroso,P. Buncic,F. Hemmer,A. Di Meglio,A. Edlund,“Programming the Grid with gLite”,COMPUTATIONAL METHODS IN SCIENCE AND TECHNOLOGY 12(1),33-45,2006