自律運用管理ミドルウェアを高速化する資源情報キャッシュ制御方式

町田 文雄[†] 小倉 章嗣[†] 西村 祥治[†] 川戸 正裕[†] 前野 義晴[†]

情報システムの運用管理にかかるコストが増加しており、自律的な運用管理を実現する自律運用管理ミドルウェアへの期待が高まっている。自律運用管理ミドルウェアで自律的な負荷分散制御や障害対処を実現するためには、管理対象の情報を正確かつ高速に参照するための資源情報管理機能が必要である。本稿では、自律運用管理ミドルウェアによる問題解決を高速化するため、資源情報管理機能で資源情報をキャッシュし、更新処理を制御するキャッシュ制御方式を提案する。提案方式は自律制御の実行状態に適応して更新制御する点を特徴とし、不要な更新処理を削減することで管理サーバにかかる負荷を削減する。実験により、管理サーバの負荷を増加させることなく、管理対象システムで発生した性能問題を解決するのに要する時間を40%短縮できることを確かめた。

Self-Scheduling Cache Controller for Autonomous Management Middleware

Fumio Machida,[†] Shoji Ogura,[†] Shoji Nishimura,[†]

Masahiro Kawato[†] and Yoshiharu Maeno[†]

The concept of autonomous management middleware aims to reduce the cost for information systems management through automating the management operations. A resource information management function, which provides resource information quickly and properly, is essential to implement the autonomous management middleware. To enhance the performance of the autonomous management middleware, we suggest a cache control framework in resource information management function adapting the behavior of autonomous management middleware. This framework shortens the processing time of failure recovery or load balancing without increasing the load of middleware. We have implemented this framework and evaluated the functionality and the performance.

1. 背景

企業やデータセンタで稼動する情報システムは多様化、複雑化しており、その運用管理にかかるコストが問題となっている。人手による運用管理のコストを削減するため、管理対象で発生した問題に対して自律的に対処する自律運用管理ミドルウェアの実現が課題とされていた。このような運用管理の自動化や自律化を目指した研究は、ポリシ制御によるネットワーク管理の分野で進んでいる[2][3]。近年は、ポリシ制御によりサーバやストレージ、アプリケーションを含む統合運用管理を自動化する研究にも発展している [5]。我々はポリシ制御機構を用いた自律運用管理ミドルウェアのプロトタイプである Polimatica [1]の開発により、これらの技術検証を進めてきた。

自律運用管理ミドルウェアにおける課題の一つに、問題が発生してからそれを解決するまでの対処時間の短縮化が挙げられる。管理対象システムの可用性や性能目標を維持するため、管理対象システムで発生した障害や負荷変動には素早く対処する必要がある。ミドルウェアによる対処時間を短くするためには、特に、管理対象となる資源情報を素早く、的確に把握することが重要である。

本稿では、自律運用管理ミドルウェアの対処時間を短縮化させる資源情報管理機能を設計する。分散

† 日本電気株式会社 インターネットシステム研究所
Internet Systems Research Laboratories, NEC Corporation

した資源情報の参照にかかる時間を短縮するため、資源情報のキャッシュ機能を取り入れ、キャッシュした情報を低負荷で更新制御するキャッシュ制御方式を提案する。提案方式は自律運用管理ミドルウェアの実行状態に適応してキャッシュ更新処理を行う点を特徴とする。提案方式と単純なキャッシュ制御方式を比較し、提案方式が低負荷で対処時間を短縮できることを実験によって確認した。

本論文の構成を以下に示す。2章では自律運用管理ミドルウェアの構成と動作を示し、対処時間を短縮させるための課題点を明確にする。3章で課題を解決する資源情報のキャッシュ制御方式について述べる。4章ではデータセンタを想定した実験システムで提案方式を評価した結果を示す。5章で関連研究をまとめ、6章で結論を述べる。

2. 自律運用管理の課題

本章では自律運用管理ミドルウェアの構成と動作を説明し、資源情報管理機能の設計における課題を示す。

2.1 自律運用管理ミドルウェアの構成

ポリシ制御を中心とした自律運用管理ミドルウェアの構成を図 1に示す。この構成は OAA (Observe-Analyze-Act) ループ[4]としても知られており、ポリシ制御による管理ミドルウェアの構成として一般的であると考えられる。

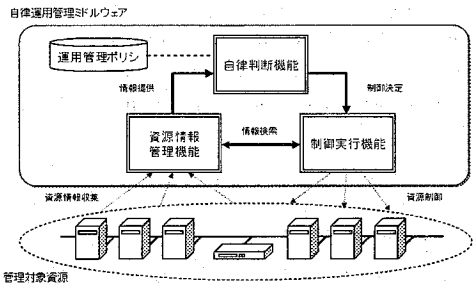

図 1 自律運用管理ミドルウェアの構成

図 1に示す自律運用管理ミドルウェアは資源情報管理機能、自律判断機能、制御実行機能で構成される。資源情報管理機能は管理対象システムを構成する資源の情報を集めて他の機能に提供する役割を果たす。自律判断機能は観測した資源情報と予め登録された運用管理ポリシとを照らし合わせ、システムに与えるべき制御の内容を決定する。この決定に基づいて制御実行機能が各管理対象に応じた制御命令を生成して実行する。

2.2 動作シーケンス

図 1に示す機能構成に基づいた自律運用管理ミドルウェアの基本動作を図 2に示す。

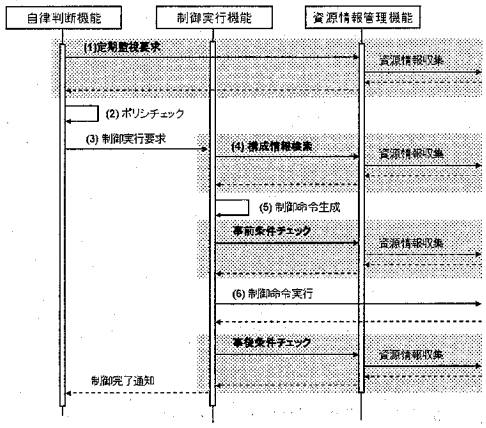

図 2 自律運用管理ミドルウェア動作シーケンス

自律判断機能は資源情報管理機能を利用して管理対象の資源を監視する (1)。ポリシで指定した状態に管理対象が遷移した際、自律判断機能はポリシに基づいて制御要求を生成し (2)、制御実行機能に要求を送る (3)。この要求を受けた制御実行機能は、管理対象の詳細な構成や設定情報を把握するため、資源情報管理機能に問い合わせる (4)。制御実行機能は問い合わせ結果に基づいて各資源に対する制御命令を生成して (5) 実行する (6)。また、制御実行機能は制御の実行前後で管理対象が満たすべき事前条件と事後条件をチェックする。ここで述べる事前条件、事後条件とは、各制御を実行する前後で管理対象が満たすべき条件であり、誤った制御を防止するために設定する条件である。

2.3 資源情報管理機能の性能要件

資源情報管理機能は自律判断機能と制御実行機能から利用される。自律運用管理ミドルウェアによる対処時間を短縮するため、資源情報管理機能の性能に関して以下が要求される。

- 迅速な変化検知

障害や負荷変動が発生してから、資源の状態変化として問題を検知するまでの時間を短くする必要がある。

- 参照処理の高速化

自律的な対処は資源情報を確認しながら実行されるため、資源情報の参照にかかる時間が対処時間に影響を与える。自律的な対処を高速化させるためには参照にかかる時間を短縮する必要がある。

2.4 課題

資源状態の変化を迅速に検知することと資源情報の参照処理を高速化させることは単純な方法では両立が難しい。資源状態の変化を迅速に検知するためには、最新の資源情報を参照する必要がある。しかし、資源に直接アクセスして情報を取得する処理には遅延が伴うため、参照処理時間が増加する可能性がある。特に、複数の資源情報を同時に参照する場合は遅延の影響を受ける可能性が高い。一方、参照処理を高速化するためには、資源情報を自律運用管理ミドルウェア内にキャッシュする方法がある。しかし、資源情報は絶えず変化する可能性があり、キャッシュした情報が必ずしも最新の資源状態を表しているとは限らない。このため、資源状態の変化を即座に検知できるとは限らず、自律制御による対処の遅れを生じてしまう可能性がある。

3. 資源情報のキャッシュ制御方式

2章で指摘した課題は、資源情報をキャッシュして、高頻度でキャッシュを更新することで解決できる可能性がある。しかし、この手法の問題点はキャッシュの更新処理によって管理サーバやネットワークに負荷がかかることである。本章では、自律運用管理ミドルウェアの実行状態に適応してキャッシュを更新するキャッシュ制御方式を提案し、この問題点の解決法を示す。

3.1 資源情報管理機能の構成

提案方式を実現する資源情報管理機能の構成を図3に示す。

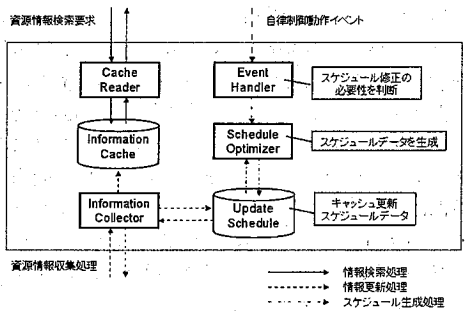

図 3 資源情報管理機能の構成

この構成では資源情報管理機能内で 3 つの処理が独立に動作する。第 1 の処理は情報検索処理で、検索要求を Cache Reader が受け付けて Information Cache から情報を取得して結果を返す。第 2 の処理は情報更新処理で、Information Collector が一定間隔でポーリング処理を開始し、資源にアクセスして取得した情報を Information Cache に格納する。どの情報をいつ更新するかは Update Schedule（キャッシュ更新スケジュール）で定められる。第 3 の処理はスケジュール生成処理で、Schedule Optimizer がキャッシュ更新スケジュールを生成する。この処理はシステム内外で発生するイベントを契機とし、Event Handler がイベントの内容に基づいて Schedule Optimizer を呼び出す。この構成の特徴は、キャッシュの更新手順をキャッシュ更新スケジュールとし、更新処理プログラムそのものから切り離している点である。これにより動的に更新処理の振る舞いを変化させることを可能としている。

3.2 キャッシュ更新スケジュール

Information Collector では、ポーリング処理を開始してからのポーリング処理回数を PC(Polling Count)としてカウントする。キャッシュ更新スケジュールはこの PC を利用し、資源情報毎に次回更新予定の PC を Next PC として指定する。各資源の情報は PC が Next PC と等しくなった際に更新処理を要求する。更新処理後は、更新頻度に基づいて設定された Interval を現在の PC に加え、新しい Next PC として上書きする。各資源に対して Next PC と Interval が指定されたキャッシュ更新スケジュールの例を図 4に示す。

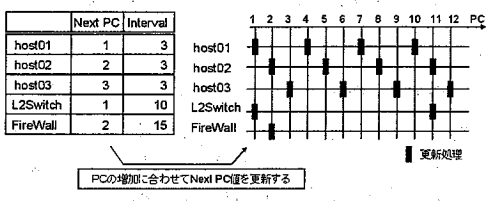

図 4 キャッシュ更新スケジュールの例

この例では、host01 の情報が PC=1,4,7,..の時点で更新される。Next PC 値を資源毎に変えるのは、一時点に更新処理が集中するのを避けるためである。

3.3 キャッシュ更新スケジュールの生成方法

キャッシュ更新スケジュールは各資源の更新頻度や管理サーバの性能を制約条件として設計する。本稿では詳しく扱わないが、制約条件を満たすスケジュールを生成する単純なアルゴリズムがあることを前提とする。本提案方式ではこの制約を満たす範囲で自律制御の実行状態や負荷状況に応じてスケジュールを動的に変更する。図 5に示す具体例で動作を説明する。制御実行機能によってネットワークの構成を変更した場合、ARP テーブルなどに基づく新しいトポロジ情報をキャッシュに反映させる必要があ

る。このとき制御実行機能はL2スイッチやFirewall（FW）などに設定変更を行ったことをイベントとして資源情報管理機能に送る。このイベントをEvent Handlerが解析し、L2スイッチやFWの情報を更新する必要があることを判断する。Schedule Optimizerはこの判断結果と現在のキャッシュ更新スケジュールから、制約条件を満たす範囲でL2スイッチやFWの情報更新処理を追加する。Information Collectorは新しく生成されたスケジュールに基づいてキャッシュ更新処理を開始するため、ネットワーク構成変更後に迅速に制御結果をキャッシュで確認することが可能となる。L2スイッチやFWの情報を常に高頻度で確認する場合と比べ、管理サーバにかかるキャッシュ更新処理の負荷を削減できる。

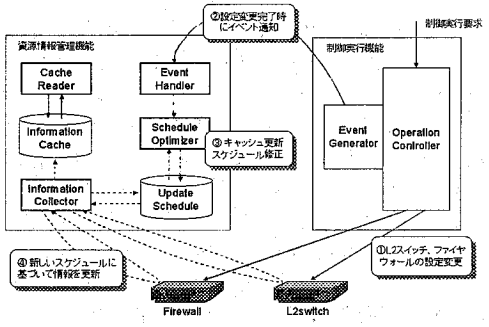

図 5　キャッシュ更新スケジュール書き換えの例

4. 実験と評価

提案したキャッシュ制御方式を自律運用管理ミドルウェアPolimaticaの資源情報管理機能として実装し、実験環境で動作検証と性能評価を行った。

4.1 実験環境

データセンタを想定した図 6に示す実験環境を構築した。

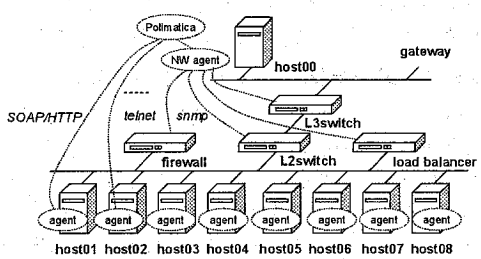

図 6　実験システムの構成

Polimaticaを管理サーバであるhost00にインストールし、資源情報をWebサービスインタフェースで提供するエージェントを管理対象ホストにインストールした。ネットワーク機器の情報は、telnetやsnmpを使う専用のエージェントを用意し、管理サーバ上で動作させた。Polimaticaの各機能はWebサービスとして実装し、自律判断機能、制御実行機能及び資源情報管理機能をSOAP/HTTPで接続させた。

管理サーバはCPU Pentium4 3GHz、メモリ1024MBのLinux(kernel 2.4.18-3smp)PCを使い、管理対象のホストはCPU 2.6GHz、メモリ512MBのPCを使った。OSはアプリケーションに応じてWindows 2000、Windows 2000 Server、Linuxのいずれかをインストールした。

4.2 キャッシュの効果測定

はじめにキャッシュ機能を利用する場合と利用しない場合の検索処理性能を比較した。検索処理性能は制御実行機能が資源情報管理機能に検索処理を要求した際の応答時間で評価した。全ての資源情報（約100KB）を何の変換処理も行わずに取得する際の検索応答時間を測定した。5回の測定結果の平均値を求め、内訳を比較した結果を表 1に示す。

表 1　検索応答時間比較

(sec)	w/o cache	w/ cache
情報収集時間	6.318	0
機能間通信時間	0.750	0.742
合計	7.068	0.742
標準偏差	1.159	0.086

検索処理時間は資源情報をエージェントから集めるための情報収集時間と、検索を要求した制御実行機能との通信時間に分けることができる。情報収集時間はキャッシュの有無に依存し、機能間通信時間は通信データ量に依存する。キャッシュの有無によらず検索結果のデータ量は等しいため、機能間通信時間に差は見られない。しかし、キャッシュがある場合（w/ cache）は情報収集時間が0に等しく、検索応答時間合計はキャッシュがない場合（w/o cache）と比べて約10分の1である。

一回の検索処理で約6 secの性能差があるため、制御実行の過程で複数回検索処理が必要な場合にはキャッシュによる高速化の効果が大きい。

4.3 キャッシュ更新制御方式の評価

提案するキャッシュ更新制御方式の有効性を評価するため、Polimaticaによる自律負荷分散制御の動作検証を行った。動作概要を図 7に示す。

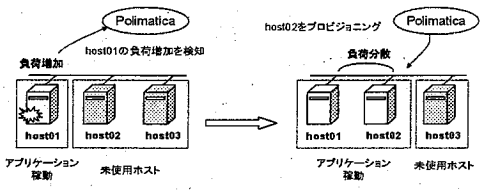

図 7　自律負荷分散制御の動作概要

host01 で Web アプリケーションが動作しており、host02 は始めの状態では未使用のホストであるとする。host01 の負荷が増加した際に Polimatica が負荷の増加を検知し、host02 を Web アプリケーション用のサーバとしてプロビジョニングする。プロビジョニング終了後は host02 によって Web アプリケーション処理が負荷分散されるので、host01 の負荷は軽減されることが期待される。

本測定では、負荷が上昇してから対処を完了するまでの時間を評価した。対処時間は短い方が望ましい。提案するキャッシュ制御方式の比較対象として 5 秒間隔でポーリングを行う短間隔ポーリング更新制御と、40 秒間隔でポーリングを行う長間隔ポーリング更新制御を用意した。提案方式を用いた測定ではポーリング間隔を 1 秒とし、表 2に示すキャッシュ更新スケジュールを初期設定として与えた。また、スケジュール書き換えルールとして表 3に示すルールを Event Handler に実装した。異なるキャッシュ更新制御方式を用いて測定した結果を表 4に示す。結果は 5 回の測定結果の平均値である。

表 2　情報更新スケジュールの初期設定

	Next PC	Interval		Next PC	Interval
host01	0	4	host07	5	15
host02	0	15	host08	6	15
host03	1	15	L2switch	0	30
host04	2	15	L3switch	1	30
host05	3	15	Firewall	2	30
host06	4	15	Balancer	3	30

表 3　スケジュール書き換えルール

発生イベント	対処
host のネットワーク設定変更	変更されたホストの情報を即座に更新する
VLAN の設定変更	L2switch の情報を 1 分間高頻度で更新する
Firewall の設定変更	Firewall の情報を 1 分間高頻度で更新する
Load Balancer の設定変更	Load Balancer の情報を 1 分間高頻度で更新する
※ プロビジョニングで設定変更される資源に対してルールを記述する	

表 4　負荷増加に対する対処時間比較

(sec)	短間隔	長間隔	提案方式
負荷上昇検知時間	7	29	8
事後条件チェック時間	1	9	2
その他	36	30	30
対処時間合計	44	68	40
標準偏差	6.6	13.3	1.7

負荷上昇の検知にかかる時間や事後条件チェックにかかる時間はキャッシュの更新頻度に依存するため、ポーリング間隔の短い方が対処時間は短い。提案方式は監視の対象となる情報や事後条件チェックの対象となる情報を集中的に更新することで、短間隔ポーリングによる更新制御の場合と同程度の対処時間を実現している。対処時間は長間隔ポーリングによる更新制御の場合と比較して約 40%短縮される。

次に、測定中の管理サーバ host00 にかかる負荷を評価した結果を表 5に示す。この測定では vmstat を 1 秒間隔で実行し、CPU 使用率のユーザ時間とシステム時間の平均値を比較した。

表 5　情報収集負荷の比較

(%)	短間隔	長間隔	提案方式
平均ユーザ CPU 使用率	42.3	22.6	21.5
平均システム CPU 使用率	4.6	0.9	1.1

短間隔ポーリングでは頻繁にキャッシュを更新するため、管理ミドルウェアにかかる負荷が大きい。提案方式は不用な情報更新処理を行わないため、長間隔ポーリングと同程度の負荷で運用できる。

5.　関連研究

ポリシ制御によるネットワーク制御の研究はネットワークの品質保証やアクセス制御機構に応用されている [3]。近年では、ポリシ制御をシステム運用管理全体に適用して運用管理の自動化を目指す研究が進んでいる [5]。特に分散した環境で個々のコンポーネントが自律的に振舞うアーキテクチャ [6]やポリシそのものを対象としたポリシリファインの研究 [7][8]や、ポリシ衝突回避の研究 [9][10]などが注目されている。これらの研究では基本となるアーキテクチャや設計指針を示しているが、実装や評価を通して得られる自律運用管理システム自体の性能問題を扱っていない。これらの新しい手法を現実の運用管理システムに適用する上では、性能に関する考慮が必要である。我々はプロトタイプ実装を通して資源

情報管理機能に性能ボトルネックが生じることを確認し、その解決手法を提案した。

システム運用管理における資源情報管理機能ではSNMP[11]が使われることが多いが、UDP ベースであるため信頼性やセキュリティの面で課題がある。近年では WBEM [12]や WSDM [13]などの資源管理技術の標準化が進められており、提案方式もこれらの標準に対応していくことを検討している。また、クラスタや計算グリッドの研究分野では性能や拡張性に重点を置いた資源情報管理システムが多く提案されており、文献 [15]が詳しい。MDS [16]は Globus Alliance [14]で実装が進められている資源情報管理システムで、MDS2 では有効期限つきのキャッシュ機能によって高速化を実現している。現在、最新版である MDS4 では push 型と pull 型の情報収集機能を備えているが、他のコンポーネントの動作状況に適応して収集パターンを制御する機能は提供されていない。グリッドの監視システムでシステムの負荷状況によって情報の収集パターンを変化させる手法[17][18]も提案されているが、制御処理の実行状態に適応して必要な更新処理を追加する機能や、更新頻度を変化させる機能は提供されていない。

6. 結論

企業やデータセンタにおけるシステム運用管理業務をポリシ制御によって自律的に行う自律運用管理ミドルウェアで、負荷変動や障害発生に対する対処を素早く実施するための資源情報キャッシュ制御方式を提案した。提案した資源情報キャッシュ制御方式は自律運用管理ミドルウェアの実行状態に応じて更新処理を変更する点を特徴とする。評価システムでの検証により、キャッシュ更新処理による管理サーバにかかる負荷を増加させることなく、自律運用管理ミドルウェアによる負荷分散制御の対処時間を40%短縮できることを確認した。提案方式を用いることで管理サーバに負荷をかけることなく、自律運用管理ミドルウェアの処理を高速化することが可能である。

参考文献

[1] Y. Maeno, M.Kawato, S. Nishimura, F. Machida and T. Kamachi, "Polimatica: Abstraction for Customizable Private Virtual Organization in Global Grids", IEEE International Conference on Web Services (ICWS2004), June, 2004, pp. 674-681.

[2] D. Verma, "Simplifying network administration using policy-based management", IEEE Network, Volume 16, Issue 2, 2002, pp. 20-26.

[3] M. Sloman, "Security and management policy specification", IEEE Network, Volume 16, Issue 2, 2002, pp. 10-19.

[4] L. Yin, S. Uttamchandani, J. Palmer, R Katz, G Agha, "AUTOLOOP: Automated Action Selection in the "Observe-Analyze-Act" Loop for Storage Systems", 6th IEEE International Workshop on Policies for Distributed Systems and Networks (POLICY 2005), June 2005, pp. 129-138.

[5] D. Agrawal, S. Calo, J. Giles and D. Verma, "Policy management for networked systems and applications", 9th IFIP/IEEE International Symposium on Integrated Network Management (IM2005), May 2005, pp. 455-468.

[6] S. White, J. Hanson, I. Whalley, D. Chess, J. Kephart, "An Architectural Approach to Autonomic Computing", IEEE International Conference on Autonomic Computing (ICAC2004), May 2004, pp. 2-9.

[7] A. Bandara, E. Lupu, J. Moffett and A Russo, "A Goal-based Approach to Policy Refinement", 5th IEEE International Workshop on Policies for Distributed Systems and Networks (POLICY 2004), June 2004, pp. 229-239.

[8] J. Rubio-Loyola, J. Serrat, M. Charalambides, P. Flegkas, G Pavlou and A. Lafuente, "Using Linear Temporal Model Checking for Goal-Oriented Policy Refinement Frameworks", 6th IEEE International Workshop on Policies for Distributed Systems and Networks (POLICY 2005), June 2005, pp. 181-190.

[9] J. Baliosian and J. Serrat, "Finite State Transducers for Policy Evaluation and Conflict Resolution", 5th IEEE International Workshop on Policies for Distributed Systems and Networks (POLICY 2004), June 2004, pp. 250-.

[10] J. Kephart and W. Walsh, "An Artificial Intelligence Perspective on Autonomic Computing Policies", 5th IEEE International Workshop on Policies for Distributed Systems and Networks (POLICY 2004), June 2004, pp. 3-12.

[11] J. Case, M. Fedor, M. Schoffstall and J. Davin, "Simple Network Management Protocol (SNMP)", RFC 1157, May 1990.

[12] WBEM (Web Based Enterprise Management). http://www.dmtf.org/standards/wbem/

[13] WSDM (Web Service Distributed Management). http://www.oasis-open.org/committees/documents.php?wg_abbrev=wsdm

[14] The Globus Alliance. http://www.globus.org/

[15] M. Gerndt, R. Wismueller, Z. Balaton, G. Gombas, P. Kacsuk, Z. Nemeth, N. Podhorzki, H. Truong, T. Fahringer, M. Bubak, E. Laure and T. Margalef. "Performance Tools for the Grid: State of the Art and Future, Automatic Performance Analisis: Real Tools White Paper, 2004.

[16] K. Czajkowski, S. Fitzgerald, I. Foster, and C. Kesselman, "Grid Information Services for Distributed Resource Sharing", IEEE International Symposium on HighPerformance Distributed Computing (HPDC10), IEEE Press, August 2001.

[17] H. Lim Choi Keung, J. Dyson, S. Jarvis and G. Nudd, "Self-Adaptive and Self-Optimising Resource Monitoring for Dynamic Grid Environments", 2nd International Workshop on Self-Adaptive and Autonomic Computing Systems (SAACS 2004), August 2004, pp. 689-693.

[18] R. Sundaresan, M. Lauria, T. Kurc, S. Parthasarathy and J. Saltzm, "Adaptive Polling of Grid Resource Monitors Using a Slacker Coherence Model", 12th IEEE International Symposium on High Performance Distributed Computing (HPDC'03), June 2003.