社団法人　情報処理学会　研究報告
IPSJ SIG Technical Report

2005－ARC－162 （10）
2005－HPC－101 （10）
2005／3／7

グリッド環境におけるスーパースケジューラ連携手法の検討

秋岡明香[†1]　竹房あつ子[†2]　中田秀基[†3,†4]
松岡聡[†4,†1]　三浦謙一[†1]

グリッド環境において効果的な負荷分散を実現するために、従来からスーパースケジューラの利用が提案されてきた。しかし、グリッド環境にスーパースケジューラが複数存在する場合の挙動は充分な検討がされていない。本稿では、スーパースケジューラの連携手法とその効果について、グリッドシミュレータを用いて検討した。その結果、複数のスーパースケジューラが単独で機能するよりも、階層構造や分散ネットワークを構成して協調することで、アプリケーションの実行待ち時間を短縮し、グリッド計算資源の利用効率を高めることを確認した。さらに、スーパースケジューラが階層構造を構成する場合と分散ネットワークを構成する場合について、利点と欠点を確認した。

A Study on the Effect of Cooperative Superschedulers on the Computational Grid

Sayaka Akioka,[†1] Atsuko Takefusa,[†2] Hidemoto Nakada,[†3]
Satoshi Matsukoka[†4,†1] and Kenichi Miura[†1]

In this paper, we evaluated the effect of the cooperative superschedulers on the computational Grid with a Grid simulator. Many studies have proposed to utilize superschedulers in order to achieve effective load balancing and improve the resource utilization. On the other hand, there is no deep consideration on the affects caused by superschedulers in different ways of cooperation. Through the simulation, we got many results to support the effectiveness of superschedulers in cooperation. Cooperative superschedulers shorten the waiting times of applications, and accelerate utilization of the resources. In addition to those results, we discuss on pros and cons of two major cooperative styles: tiers and distributed network.

1. はじめに

近年、地理的に分散している複数の計算資源を相互に高速なネットワークで接続し、これらを一つの計算基盤とみなして使用するグリッドコンピューティングに関する研究が盛んである。グリッド技術を用いることで、単体の大型並列計算機や計算機クラスタシステム等では実現することが難しい、非常に大規模な計算基盤を構成することが可能である。この際、異なる管理組織の計算資源を動的に連携させ、巨大な仮想計算機を形成することも可能であり、グリッド技術により従来よりも柔軟かつ容易に膨大な計算資源を利用することが可能となる。

一方で、この大規模な計算基盤を効果的に使用するためには、実行要求アプリケーションを適切な計算資源に割り当てるためのスケジューラが必要になるが、グリッド環境では膨大な数の計算資源を対象としてアプリケーション配置を決定する必要があるため、単一のスケジューラによりすべての要求アプリケーションを処理することは現実的ではない。そこで、計算機クラスタシステムなどの比較的小規模な単位で（ローカル）スケジューラがアプリケーション配置を行ない、計算資源を管理する組織などの中規模な単位でスーパースケジューラがローカルスケジューラをとりまとめ、計算資源の管理を行なう方法が提案されてきた[1)]。このような構成では、スーパースケジューラが複数の中規模計算基盤を対象として、より適切なアプリケーション割当て先を探し当てることが期待されている。しかし、スーパースケジューラが複数存在する場合の作用ついては、未だ充分な検討がされていない。

†1 国立情報学研究所
National Institute of Informatics
†2 お茶の水女子大学
Ochanomizu University
†3 産業技術総合研究所
National Institute of Advanced Industrial Science and Technology(AIST)
†4 東京工業大学
Tokyo Institute of Technology

そこで本稿では、多数のスーパースケジューラの連携によって、アプリケーションの実行待ち時間やグリッド計算資源の使用効率がどのような影響を受けるかについて、グリッドシミュレータを用いて検討した。その結果、複数のスーパースケジューラが相互に連携を取りつつ負荷分散を行なうことで、アプリケーションの実行待ち時間を短縮し、計算資源の利用効率を向上させることが分かった。

以下、第 2 章で、これまでの関連研究についてまとめると同時に、従来研究と本研究との違いを述べる。第 3 章では、本稿で検討したスーパースケジューラの連携形態について説明する。第 4 章でシミュレーション手法の詳細について述べた後、第 5 章でシミュレーションにより得られた結果について考察する。最後に、第 6 章で、シミュレーションにより得られた結果と今後の検討課題をまとめる。

2. 関 連 研 究

Shan らは、グリッド環境において単一のスケジューラが全てのグリッド計算資源を管理する場合と、複数スーパースケジューラが分散ネットワークを構成して相互に連携を取りながら計算資源を管理する場合を、アプリケーションの応答時間、実行待ち時間、及びスーパースケジューラの指示によって別のローカルスケジューラの管理下にある計算資源へ転送されたアプリケーションの数、といった観点で比較している[2]。特に複数スーパースケジューラの場合については、sender-initiated、receiver-initiated[3]、hybrid[4] といった従来から広く知られる手法をグリッド環境で適用した場合について考察している。さらに、単一スケジューラの場合については、従来の集中管理方式の他に、Shan らが提案するアルゴリズムとの比較を行なっている。

Shan らはこれらの手法をシミュレーションにより比較しているが、National Energy Research Scientific Computing Center、Berkeley National Laboratory、及び Lawrence Livermore National Laboratory で 6ヶ月に渡って採取したログと、このログに基づく計算によってアプリケーションの実行時間や到着間隔を設定している。したがって、対象とする計算環境は高速なネットワークで相互接続された 6 台の大型計算機であり、各計算機の性能は大きく異ならないため、均質環境として扱っている。また、アプリケーションは全て各計算機内でのみ実行されるバッチジョブである。

Shan らはグリッド環境の負荷の大小に関わらず、スーパースケジューラが有効に機能する結果は示したものの、アプリケーションの転送コスト、ネットワークコスト、スーパースケジューラの拡張性、耐故障性、非均質環境での検証、複数計算資源を用いるアプリケーションの扱いといった重要な問題を考慮しておらず、今後の課題としている。

本稿では、グリッドシミュレータを用いることで、アプリケーションの転送コストを考慮した検討を行なう。さらに、スーパースケジューラが対象とする計算機数を増やし、各計算機性能は異なるものと設定する。つまり、Shan らの研究よりも規模が大きい非均質環境を想定しているため、より現実のグリッド環境に近い想定でシミュレーションを行なう。シミュレーションの詳細については第 3 章以降で述べる。

3. スーパースケジューラの連携手法

スーパースケジューラの連携手法としては様々な形態が考えられるが、本稿では、1) スーパースケジューラの連携がない手法、2) スーパースケジューラが階層構造を構成して連携する手法、3) スーパースケジューラが分散ネットワークを構成して連携する手法、の 3 つの手法について検討する。それぞれの連携手法について以下に詳細を述べる。

なお本稿では、計算機を管理する組織単位でローカルスケジューラを持ち、地域や国といった単位でスーパースケジューラがローカルスケジューラを統括する、といった状況を想定している。

3.1 連 携 な し

各スーパースケジューラは、対象計算資源で発生したアプリケーションのみを、内部の計算資源のみに対して割り当てる。内部計算資源の負荷状態に関わらず、他のスーパースケジューラが管理する計算資源へはアプリケーションを割り当てない。つまり、アプリケーションは地域や国、といった範囲を超えて割り当てられることがない。実際に Grid3[5] では、各スケジューラが資源情報を収集可能な範囲でアプリケーション割当て先を決定しており、スケジューラは連携をとっていない。

3.2 階 層 構 造

スーパースケジューラが図 1 のような階層構造を構成し、他のスーパースケジューラと連携することで、比較的負荷が小さい計算資源を探してアプリケーションを割り当てることにより、負荷の分散を計る。本稿では、3.3 で述べる分散ネットワークを構成して連携する手法との公平な比較のために、次の手順によって対象計算資源で発生したアプリケーションの割り当て先を決定する。

なお以下では、図 1 のスーパースケジューラ SS_1 の対象計算資源でアプリケーションが発生したと仮定する。

(1) 対象計算資源の最短キュー長 SQL_{SS_1} と、スーパースケジューラ SSS_1 が対象としている全ての計算資源の最短キュー長 SQL_{SSS_1} を比較する。$SQL_{SS1} > SQL_{SSS_1}$ の場合には、よりキュー長が短い計算資源が SSS_1 の対象内にあるため、その計算資源にアプリケーションを割り当てる。

(2) 手順 (1) でアプリケーション割り当て先が見つからない場合、SQL_{SS_1} と、スーパースケジューラ $SSSS_1$ が対象としている計算資源の最短キュー長 SQL_{SSSS1} を比較する。$SQL_{SS1} > SQL_{SSSS_1}$ の場合には、よりキュー長が短い計算資源が $SSSS_1$ の対象内にあるため、その計算資源にアプリケーションを割り当てる。

(3) 手順 (2) でアプリケーション割り当て先が見つからない場合、手順 (1) 及び (2) と同様に、より上位層のスーパースケジューラが対象としている計算資源の最短キュー長と SQL_{SS_1} を比較し、より最短キュー長が短い計算資源を見つけるか、最上位層のスーパースケジューラに到達するまで続ける。

(4) 最上位層のスーパースケジューラに到達しても SQL_{SS_1} よりも短いキュー長を持つ計算資源が見つからない場合には、対象計算資源にアプリケーションを割り当てる。

つまりこの連携手法では、適切なアプリケーション割り当て先を探すために、手順を追うごとにより広範囲な資源管理を行なうスーパースケジューラへ、問い合わせを行なう。

3.3 分散ネットワーク

スーパースケジューラが図 2 のような分散ネットワークを構成し、他スーパースケジューラが対象とする、より負荷が小さい計算資源を探してアプリケーションを割り当てることで、負荷の分散を計る。以下に、図 2 のスーパースケジューラ SS_1 の対象計算資源でアプリケーションが発生したと仮定し、この連携手法によりアプリケーション割り当て先を決定する手順を述べる。

(1) 対象計算資源の最短キュー長 SQL_{SS_1} と、スーパースケジューラ SSS_1、...、SSS_n が対象としている計算資源の平短キュー長 SQL_{SSS_1}、...、SQL_{SSS_n} を比較する。$SQL_{SS1} > SQL_{SSS_k}$ である $k(1 \leq k \leq n)$ が存在する場合には、SSS_k の計算資源にアプリケーションを割り当てる。

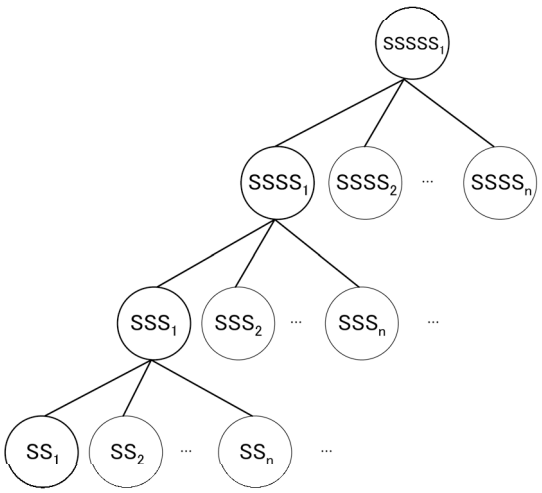

図 1 スーパースケジューラが階層構造を構成して連携する例

(2) 手順 (1) でアプリケーション割り当て先が見つからない場合、SQL_{SS_1} と、スーパースケジューラ $SSSS_1$、...、$SSSS_m$ が対象としている計算資源の最短キュー長 SQL_{SSSS_1}、...、SQL_{SSSS_m} を比較する。$SQL_{SS1} > SQL_{SSSS_k}$ である $k(1 \leq k \leq m)$ が存在する場合には、$SSSS_k$ の計算資源にアプリケーションを割り当てる。

(3) 手順 (2) でアプリケーション割り当て先が見つからない場合、手順 (1) 及び (2) と同様に、より遠方のスーパースケジューラが対象としている計算資源の最短キュー長と SQL_{SS_1} を比較し、対象計算資源よりも最短キュー長が短い計算資源を見つけるか、最末端のスーパースケジューラに到達するまで続ける。

(4) 最末端のスーパースケジューラに到達しても SQL_{SS_1} よりも短い最短キュー長を持つサイトが見つからない場合には、対象計算資源にアプリケーションを割り当てる。

つまりこの手法では、より適切なアプリケーション割当て先を探すために、近隣の同程度の規模の計算資源を管理するスーパースケジューラへ問い合わせを行なう。

4. シミュレーション

本稿では、第 3 章で述べた連携手法について、Bricks[6] を用いて比較・評価する。Bricks は、グリッド環境でのスケジューリングアルゴリズムとそのフレー

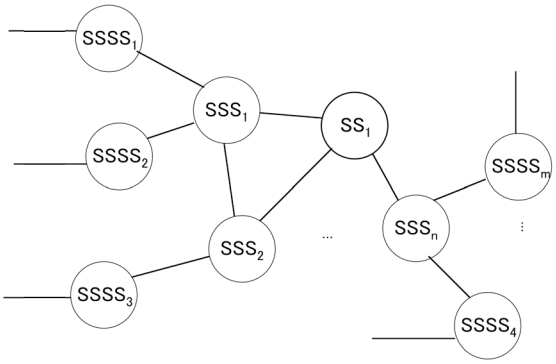

図 2　スーパースケジューラが分散ネットワークを構成して連携する例

ムワークの評価基盤を提供するシステムで、大規模かつ再現性のある評価実験を可能にする。また、様々な性能評価環境やスケジューリングアルゴリズム、及びスケジューリングに関するモジュールを設定できる。

そこで、第 3 に基づいて連携を行なうスーパースケジューラを Bricks モジュールとして実装し、Bricks 内部のグリッド環境を様々に設定して、アプリケーションの実行待ち時間、アプリケーションの実行時間、グリッド環境の利用効率などの側面から比較した。以下にシミュレーション条件の詳細を述べる。

4.1　想定するアプリケーション

想定するアプリケーションは、シングルスレッド／タスクのアプリケーションのみとし、アプリケーション間での通信や同期は発生しないものとする。アプリケーションは、各スーパースケジューラの対象計算資源に指数分布に基づく時間間隔て到着し、各アプリケーションの実行開始時の転送データ量、実行命令数、実行終了後の転送データ量はアーラン分布に基づくように定める。

4.2　スーパースケジューラのネットワークトポロジ

スーパースケジューラのネットワークトポロジについては、GridG[7] により生成する。GridG は、グリッド環境を想定したネットワークトポロジを生成するソフトウエアで、Doar の階層構造生成ツール[10] が生成したグラフを基にして、outdegree power law に従うようにノード間リンクを追加する。近年の研究により、インターネットにおける各種ネットワークトポロジや P2P ネットワークのトポロジはベキ分布に従うことが知られており[8]、GridG によりスーパースケジューラのネットワークトポロジを生成することは順当であると言える。

4.3　その他のグリッド環境

各計算ノードの計算資源や、これらの計算資源を結ぶネットワーク性能については、Kee らが提案するモデル[9] に基づいて設定する。Kee らは、Lawrence Berkeley National Lab、Ganglia demo、U.C. Berkeley Millennium Project や NPACI などの実際の計算基盤を集計し、各種計算資源の性能分布をモデル化する。また、各スーパースケジューラが管理するローカルスケジューラ数、及び各ローカルスケジューラが管理する計算ノード数については、一様乱数で決定する。なお、全ての計算ノードはスーパースケジューラによってスケジューリングされたアプリケーションが占有利用できるものとし、スケジューリングされたアプリケーションは FCFS で実行されるものとする。

表 1　シミュレーション中の計算ノード数及びスーパースケジューラ数

	平均	標準偏差
スーパースケジューラ数（連携なし／分散）	10.4	2.2
スーパースケジューラ数（階層構造）	16.2	3.3
ローカルスケジューラ数	63.6	14.2
計算ノード数	341	73.9

表 2　計算ノードの計算性能と分布

計算性能 [GFlops]	割合 [%]
0.56	20.2
0.60	16.2
0.64	15.3
0.68	19.3
0.72	16.5
0.76	12.4

表 3　シミュレーション中のアプリケーション設定

	平均	標準偏差
実行命令数 [G operations]	187.8	201.8
実行開始時転送データ量 [MB]	101.0	110.6
実行終了時転送データ量 [MB]	94.9	101.4

5.　シミュレーション結果と考察

第 4 章で述べた方法で仮想グリッド環境を生成し、第 3 章で述べた各連携手法を検討した結果を述べる。

シミュレーション全体を通しての総スーパースケジューラ数、スーパースケジューラひとつ当たりのローカルスケジューラ数、およびローカルスケジューラひとつ当たりの計算ノード数の平均値と分散を表 1 に、各計算ノードの計算性能の分布を表 2 にそれぞれ示す。また、アプリケーションの実行命令数、実行開始時転送データ量、および実行終了時転送データ量の平均値と分散を、表 3 に示す。

表 4、表 5、表 6 に、シミュレーション結果のうち

表 7　アプリケーション配置先決定時のスーパースケジューラ間通信メッセージ数

	分散ネットワーク	階層構造
平均メッセージ数	22.32	74.33

の代表的な例について、シミュレーション全体を通しての計算サイトの負荷分布を集計した結果を示す。表 4 における SS7 や SS8 のように、極度に負荷が集中するサイトが存在する場合に特に、スーパースケジューラが連携をとることで、表 5 や表 6 における SS7 や SS8 に見られるように、負荷が分散し、グリッド環境での計算資源利用率を向上させていることが分かる。このことは、表 4、表 5、表 6 を比較した際に、計算ノードの平均負荷の標準偏差が小さくなっていることからも確認できる。

図 3 には、アプリケーションの実行待ち時間、グリッド環境全体の負荷、及びスーパースケジューラの連携手法の関係を示す。この結果より、グリッド環境全体の負荷の大小に関わらず、スーパースケジューラが相互に連携して負荷分散を行なうことにより、アプリケーションの平均待ち時間を大幅に短縮することが可能であり、スーパースケジューラの連携は効果的な負荷分散を実現する上で有効な手段であることがわかる。さらに、スーパースケジューラの連携手法のうち、階層構造と分散ネットワークにおいて、アプリケーション配置先決定時のスーパースケジューラ間のメッセージ数をまとめた結果を、表 7 に示す。図 3 および表 7 の結果より、アプリケーション実行待ち時間の短縮には、階層構造による連携が分散ネットワークによる連携よりも有効であることが分かる。しかしその一方で、階層構造による連携の場合には、アプリケーション配置先を決定するためにスーパースケジューラ間で交換する平均メッセージ数が、分散ネットワークによる連携の場合と比較して約 3.3 倍となる。本稿では、スーパースケジューラの設置単位として、国や地域を想定しているため、スーパースケジューラの通信コストが負荷分散結果に大きな影響を与えることが予想できる。そのため、今後の課題として、スーパースケジューラの通信コストを考慮して、連携手法ごとの効果について考察する必要がある。

6. まとめと今後の課題

本稿では、グリッド環境において複数のスーパースケジューラが連携することが、アプリケーションの応答時間やグリッド計算資源の利用効率に与える影響について、グリッドシミュレータを用いて検討した。

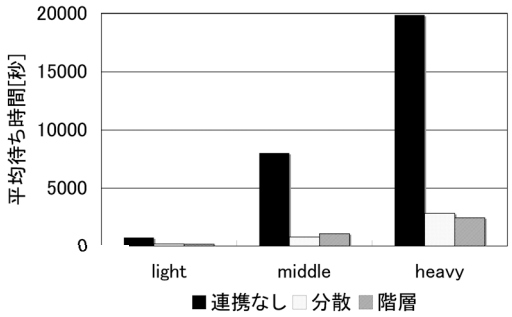

図 3　アプリケーション待ち時間の平均

その結果複数のスーパースケジューラの連携は、アプリケーションの実行待ち時間を短縮し、グリッド計算資源の利用効率を高めることを確認した。特に、スーパースケジューラが階層構造を構成して連携する場合には、アプリケーションの実行待ち時間を大幅に短縮することを確認したが、その一方で階層構造の連携は、分散ネットワークを構成しての連携と比較して、スーパースケジューラ間の通信において、約 3.3 倍のコストを要するという結果も得た。

したがって今後の課題として、スーパースケジューラ間の通信コストを考慮した連携手法の比較を行なうほか、さらに大規模かつ現実的なグリッド環境を想定したシミュレーションを行なうことで、スーパースケジューラの連携手法について検討を進める予定である。

参　考　文　献

1) Jennifer M. Schopf: Ten Actions When Super-Scheduling, GGF Documents, GFD.4 (2001).
2) Hongzhan Shan, Leonid Oliker: Job SuperScheduler Architecture and Performance in Computational Grid Environments, Proceedings of IEEE/ACM Supercomputing 2003 (SC'03) (2003).
3) Derek L. Eager, Edward D. Lazowska, John Zahorjan: A Comparison of Receiver-Initiated and Sender-Initiated Adaptive Load Sharing, Performance Evaluation, Vol.6, 1986.
4) Lionel M. NI, Kai Hwang: Optimal Load Balancing in a Multiple Processor System with Many Job Classes, IEEE Trans. Software Engineering, Vol.SE-11, No.5, 1985.
5) GRID3, http://www.ivdgl.org/grid2003/.
6) 竹房あつ子, 合田憲人, 松岡聡, 中田秀基, 長嶋雲兵: グローバルコンピューティングのスケジューリングのための性能評価システム, 情報処理学会論文誌, Vol. 41, No. 5, pp.1628-1638 (2000).
7) Dong Lu, Peter Dinda: Synthesizing Realistic Computational Grids, Proceedings of

表 4　計算サイトの負荷分布の集計結果（スーパースケジューラ連携なし）

	SS0	SS1	SS2	SS3	SS4	SS5	SS6	SS7	SS8	SS9	SS10	SS11
ローカルスケジューラ数	5	5	9	6	4	5	9	5	10	8	4	4
各ローカルスケジューラ下の計算ノードの平均負荷	0.55	0.35	0.75	3.97	0.95	0.70	0.19	59.2	244.7	0.47	0.17	0.079
各ローカルスケジューラ下の計算ノードの平均負荷の標準偏差	0.69	0.57	0.89	2.95	0.86	0.77	0.45	46.1	167.0	0.81	0.43	0.27

表 5　計算サイトの負荷分布の集計結果（分散ネットワーク）

	SS0	SS1	SS2	SS3	SS4	SS5	SS6	SS7	SS8	SS9	SS10	SS11
ローカルスケジューラ数	5	5	9	6	4	5	9	5	10	8	4	4
各ローカルスケジューラ下の計算ノードの平均負荷	1.22	0.43	0.75	1.60	0.88	0.70	11.3	2.84	31.1	0.48	0.17	0.080
各ローカルスケジューラ下の計算ノードの平均負荷の標準偏差	1.98	0.68	0.90	1.37	0.82	0.77	44.1	2.33	14.0	0.82	0.43	0.27

表 6　計算サイトの負荷分布の集計結果（階層構造）

	SS0	SS1	SS2	SS3	SS4	SS5	SS6	SS7	SS8	SS9	SS10	SS11
ローカルスケジューラ数	5	5	9	6	4	5	9	5	10	8	4	4
各ローカルスケジューラ下の計算ノードの平均負荷	0.55	0.35	0.75	1.61	0.91	1.10	6.65	5.98	33.1	0.47	0.16	0.079
各ローカルスケジューラ下の計算ノードの平均負荷の標準偏差	0.68	0.57	0.89	1.38	0.84	1.41	34.8	10.06	14.8	0.81	0.43	0.27

IEEE/ACM Supercomputing 2003 (SC'03) (2003).
8) Michalis Faloutsos, Petros Faloutsos, Christos Faloutsos: On Power-low Relationships of the Internet Topology, Proceedings of SIGCOMM '99 (1999).
9) Yang-Suk Kee, Henri Casanova, Andrew A. Chien: Realistic Modeling and Synthesis of Resources for Computaional Grids, Proceedings of IEEE/ACM Supercomputing 2004 (SC'04) (2004).
10) Matthew B. Doar: A Better Model for Generating Test Networks, Proceedings of IEEE Global Internet (1996).