2003－HPC－95　(27)
2003／8／5

メガスケール環境シミュレータ Anastasia における詳細シミュレーション

鈴木雄大[†]　柴田俊介[†]
大野和彦[†,☆]　中島浩[†]

我々は 100 万台規模のプロセッサを用いたメガスケールコンピューティングについての研究を進めている．同時に，その環境であるメガスケール環境を仮想的に提供するシミュレータ Anastasia を提案しており，本稿ではその Anastasia における詳細シミュレーションについて述べる．

詳細シミュレーションでは実行シミュレーション方式をとり，高精度なメガスケール環境の再現を目指している．具体的には，シミュレーション対象のアプリケーションを実行し，その情報をもとにメガスケール環境を再現する．さらに，再現したメガスケール環境をアプリケーションにフィードバックすることで，高精度なシミュレーションが可能になる．

モンテカルロ法で π を計算するアプリケーションを使用してシミュレータの精度を評価したところ，平均で 7.7%の誤差となり精度の高い詳細シミュレーションを行えることを示した．

Detailed simulation in Megascale simulator Anastasia

Takeo Suzuki,[†] Syunsuke Shibata,[†] Kazuhiko Ohno[†,☆]
and Hiroshi Nakashima[†]

We are pursuing a research project on megascale computing in which more than one million processors are involved. Simultaneously, we propose a simulator named Anastasia, which provides a virtual megascale environment. This paper describes detailed simulation in megascale simulator Anastasia.

In detailed simulation, we aim accurate megascale environment's behaviour by emulation. By running the object of simulation, we retrieve the information to reproduce the megascale environment. Furthermore, it is possible to get more accurate simulation by feedbacking the reproduced megascale environment into the application.

To evaluate the validity of the time management and scheduling, we build a prototype of Anastasia and measured its timing accuracy with a simple parallel application executed on a virtual PC cluster. The result shows that the average time error between the virtual and real cluster is only 7.7%.

1. はじめに

現在，ゲノム情報解析や環境・気象・災害シミュレーションといった巨大な計算能力が要求される分野においては 1Pflops 以上の性能が期待されている．そのような要求に答えるべく，我々の研究グループではハードウェア/ソフトウェア協調による低電圧化技術や，大規模並列タスクの実行モデル構築・利用技術などのコモディティ技術を活用することで，100 万台規模のプロセッサを用いたメガスケールコンピューティングの実現を目指している．

メガスケールコンピューティングでは，プロセッサをいくつかのサイトに分散して配置することを想定しており，このような環境において並列処理を効率良く行うためにはタスクの配置や実行順序を決定するタスクスケジューラが重要である．このタスクスケジューラはメガスケール環境のプロセッサやネットワークの状況，タスクの挙動や性質等を考慮し，最適なタスクの配置を決定することが求められる．しかし，メガスケール環境は多数のプロセッサとネットワークで構成されるため，性質が不均一・複雑になってしまう．そこで，シミュレータで様々なメガスケール環境を構築し，その上でタスクスケジューラの開発・性能評価を行いたいという要求がある．また，実際の環境でタス

† 豊橋技術科学大学
Toyohashi University of Technology
☆ 現在，三重大学
Presently with Mie University

クを実行することなく，タスクの挙動や性質等の情報を取得したいという要求もある．メガスケール環境では計算機のダウンやネットワークの切断などが頻繁に起こると予想でき，故障時の対応も必要である．そこで，このような複雑な環境の詳細なシミュレータが必要になる．

そこで我々の研究グループではメガスケールシミュレータ Anastasia の開発を進めている．Anastasia では仮想メガスケール環境を構築し，タスクスケジューラの性能評価やタスクの挙動・性質等の情報を取得することを目指している．現在はその第一段階として，小規模な環境での詳細なシミュレーションの実現を目指しており，本稿ではタスクの挙動を正確にシミュレーションするイベントスケジューラ等について述べる．

以後，2 章で Anastasia の概要，3 章でイベントスケジューラについて述べ，4 章で予備評価を行う．5 章で関連研究について述べ，最後に 6 章でまとめと今後の課題について述べる．

2. Anastasia の概要

Anastasia ではクラスタ等の比較的安価なシステムを使いメガスケール環境を再現する．詳細シミュレーションではシミュレーション対象のアプリケーションをクラスタ等で実行し，その情報をもとにメガスケール環境での挙動を再現するという実行シミュレーション方式を用いる．この方式はモデルシミュレーション方式に比べ，時間はかかるものの詳細なシミュレーションを行えるという利点がある．

実行シミュレーション方式ではアプリケーションを構成するタスクに対して仮想的なメガスケール環境を提供する必要がある．このため，Anastasia ではタスクの実行情報をもとにメガスケール環境での挙動を再現し，その結果をタスクにフィードバックする．例えば，時刻を取得する gettimeofday は実時刻ではなくメガスケール環境における時刻を返さなければならない．また，タスクが受信を行うために select を呼び出した場合も，実時刻ではなくメガスケール環境で実行したときに受信可能なデータがあるかどうか判断しなければならない．

Anastasia は**図 1** のような構成をとる．各部の役割について以下で述べる．

2.1 シナリオ

Anastasia ではユーザが記述したメガスケール環境の設計図（以後，シナリオと呼ぶ）をもとに仮想メガスケール環境を構築する．このシナリオには，メガスケール環境を構成するプロセッサやネットワークの性能情報，メガスケール環境で実行するタスクの情報などを記述し，後述するイベントスケジューラが再現する．

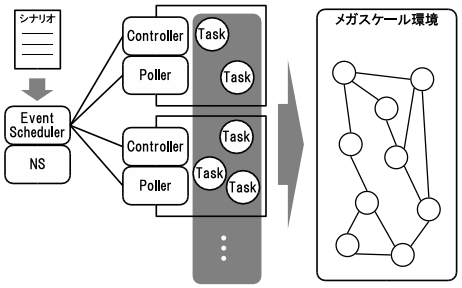

図 1 Anastasia の構成

100 万台規模のプロセッサと，それらを繋ぐネットワークから構成されるメガスケール環境では，プロセッサのダウンやネットワーク性能の変動・切断等が避けられない．そこで，このようなメガスケール環境の動的な変化もシナリオに記述しシミュレーションする．ただし，現在は実装されていない．

これらにより，動的な変化も含めた現実的な環境におけるタスクスケジューラの評価が可能になる．

2.2 アプリケーション

対称とするアプリケーションは C 言語で書かれた TCP/IP ソケットプログラムで，一部のシステムコールを Anastasia で用意した関数に置き換える．これをクラスタ等の小規模な環境で実行し，後述するコントローラが制御することでメガスケール環境上で実行しているかのように動作する．

2.3 コントローラ・ポーラー

アプリケーションの挙動等の情報収集，および仮想メガスケール環境の提供を行う．

コントローラはアプリケーション中に埋め込んだ Anastasia の関数を使い，後述するイベントスケジューラに対する情報の提供，イベントスケジューラが再現する仮想メガスケール環境情報の反映を行う．ただし，後者はイベントスケジューリングが進まないと取得できない場合がある．この場合はタスクの動作を停止し，仮想メガスケール環境情報が取得できるまで待つ．また，タスクで発生した通信の中継も行う．

ポーラーはイベントスケジューラが必要とするときに，タスクが消費した CPU 時間等の情報収集を行う．この作業はタスクを停止することなく行うことができる．

2.4 イベントスケジューラ

イベントスケジューラでは，コントローラ・ポーラーが収集した情報をもとにメガスケール環境の再現を行

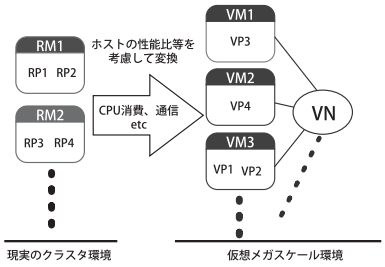

図 2 実環境と仮想環境

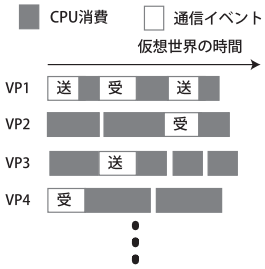

図 3 イベントリスト

う．また，その結果をもとにタスクに対してコントローラ経由で仮想メガスケール環境情報の反映を行う．

イベントスケジューラついては 3 章で詳しく述べる．

3. イベントスケジューラ

メガスケール環境を高精度に再現するためには，タスクの挙動をメガスケール環境に写像する際の精度が重要になる．Anastasia ではタスクで発生した通信等の命令をイベントとして扱い，それらをイベントスケジューラを用いてメガスケール環境に写像する．

以下でイベントスケジューラについて詳しく述べる．

3.1 シミュレーションモデル

メガスケール環境は**図 2** に示すようにプロセッサを表す Virtual Machine 群（以下 VM）とそれら結ぶ Virtual Network（以下 VN）で構成される．各 VM はプロセスを表す Virtual Process 群（以下 VP）から構成される．また，各 VP は**図 3** に示すようなイベントのリストからなり，時刻によってプロセスの挙動や状態が変化する様子を再現する．

3.2 対象イベント

イベントスケジューラは以下に挙げるイベントをメガスケール環境に写像することで仮想メガスケール環境を再現する．

- ソケット通信
  listen, connect, accept, write, read, select
- 情報取得
  gettimeofday
- CPU 時間消費

現在，Anastasia ではソケット通信を用いたタスクをシミュレーション対象にしている．このため，イベントスケジューラでは listen や connect などの，ソケット通信で使用される命令をイベントスケジューリングの対象とする．これらを仮想メガスケール環境に写像することで，接続が確立する時刻等を正確に再現することができる．また，情報を取得する命令のうち，gettimeofday のようにその命令が発生した時刻が結果に影響するものもイベントスケジューリングの対象にしている (結果に影響しないものはコントローラで処理している)．

これらのイベントをメガスケール環境に写像する際には，タスクが消費した CPU 時間情報を利用して行う．Anastasia では式 1 を用いて，タスクが消費した実 CPU 時間から仮想メガスケール環境で消費する CPU 時間を算出する．

$$\text{仮想 CPU 消費時間} = rate \times \text{実 CPU 消費時間}$$
$$\text{ただし，}\quad rate = \frac{\text{実ノード性能}}{\text{仮想 CPU 性能}} \qquad (1)$$

read 命令のように，通信相手と接続されているか，データが到着しているか，といった先行するイベントに依存するものもある．このような場合はタスクが消費した CPU 時間等の情報に加えて，read 命令は対応する write 命令が送信しなければ終了できないというような依存関係も考慮してメガスケール環境へ写像する．

3.3 スケジューリング方法

イベントスケジューラはイベントの開始・終了などによりアプリケーションの状態が変わる時刻をイベントスケジューリング時刻の候補とする．この候補のうち最も早い時刻までを仮想メガスケール環境に写像し，その結果をアプリケーションにフィードバックする．この手順を繰り返すことによってメガスケール環境を再現する．これについて以下で詳しく述べる．

図 4 にタスクで通信イベントが発生してからフィードバックを行うまでの一連の流れを示す． タスクは，それを実行するクラスタとシミュレーション対象であるプロセッサの性能比が一定でないため，図 4 の実際の環境のように進み具合にばらつきが生じる．そのため，実行しているタスクプロセスでのイベント発生順にイベントスケジューリングを行った場合，進みの遅いタスクからメガスケール環境における過去のイベントが発生してしまう恐れがある．過去のイベントが未来のイベントに影響を与える可能性があるため，仮想

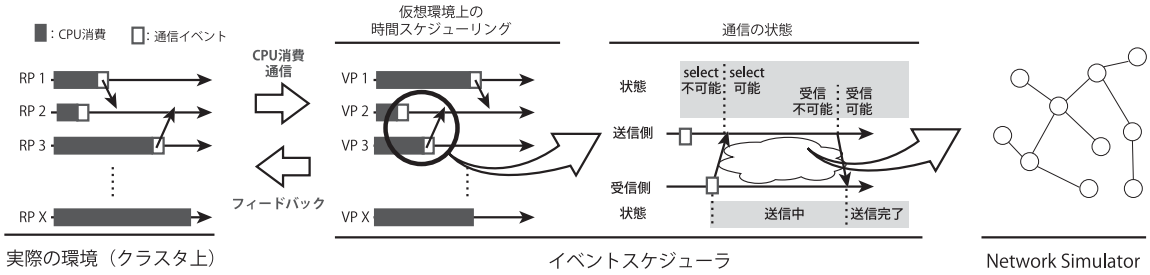

図 4 イベントスケジューリングの流れ

メガスケール環境を巻き戻してイベントスケジューリングをやりなおす必要がある．実行シミュレーション方式での巻き戻しは非常に複雑・困難で，巻き戻しにかかるコストも非常に大きい．そこで，Anastasia では巻き戻しが発生しないよう，メガスケール環境における発生順で写像を行う．

イベントスケジューラは次にイベントの状態が変化する時刻（イベント開始等），もしくは他タスクに与える影響が変化する時刻をイベントスケジューリング時刻の候補とする．例えば write イベントの場合，イベントの途中で通信相手の read・select イベントに影響を与える．まず，最初のデータが受信側に到達したときに select が受信したことを検出可能になり，次に受信側が最後の ack を送信したときに recv が完了する条件が整う．このように，1 つのイベント内でも他のタスクに与える影響が変化する．これらを再現するため，write イベントに関しては select での検出が可能になる時刻，read が完了する条件が整う時刻，write が完了する時刻という 3 つの時刻のイベントスケジューリングを行う必要がある．

図 4 のように write イベントの発生時刻が決定した場合には，ネットワークシミュレータを使って通信遅延を計算し，最初のデータが受信側に到達する時刻を求める．この時刻までの写像を行った場合，受信側のタスクで select が可能な状態に入るので，select イベントが発生していないか調べ，発生していれば select で受信を検出可能になったことをタスクに反映する．

3.4 ネットワークシミュレーション

Anastasia の詳細シミュレーションではネットワークの状態をシミュレーションするために ns(Network Simulator)[1] を用いた．ns はイベントドリブン型のネットワークシミュレータであり，パケットレベルでのシミュレーションを行う．ns では様々なレイヤーのプロトコルが実装されており，Anastasia では TCP のシミュレーションを行う．

ns の利用目的は Anastasia でのシミュレーション中にネットワークのシミュレーションを行い，シミュレーション内の通信の順序を決定することと，後で結果が利用できる様にパケットレベルのトレース結果を出力することである．

Anastasia と ns をパイプで接続し，以下のように利用している．

(1) Anastasia 側でメガスケール環境での開始時刻確定した通信イベントがそろった時点でイベントを ns に与える．
(2) ns のネットワークのシミュレーションをスタートさせる．
(3) ns 内でイベント終了（send と send に対応する recv が終了) した時点で，その結果を Anastasia へ出力する．同時に，その時間で ns は一旦停止する．
(4) Anastasia に返ってきた結果を元にして，次のイベントスケジューリングを行う．

この一連の流れを繰り返すことで，Anastasia の必要とするネットワークシミュレーションをオン・ザ・フライで行う．また，この時のネットワークシミュレーションのパケットレベルのトレース結果も保存されているので，今後の解析などの利用も可能である．

4. 予備評価

予備評価では，アプリケーションを実システムで実行したときの挙動と，Anastasia でシミュレーションした場合の挙動を比較し，イベントスケジューラの精度について評価した．

4.1 評価用アプリケーション

予備評価はモンテカルロ法を用いて π の値を Master-Worker 型で計算するアプリケーションを用いた．Worker は Master からジョブをもらい，これを処理した結果を Master に返すという動作を繰り返す．これを 1Master-16Worker という構成で実行する．

4.2 評価方法

予備評価では，ジョブ内部の挙動を再現できている

表 1 予備評価用システム

	実システム	Anastasia
CPU	PentiumIII 1GHz	
メモリ	256MB	
OS	Vine Linux 2.1.5	
ネットワーク	1Gbps Ethernet	
台数	17	3

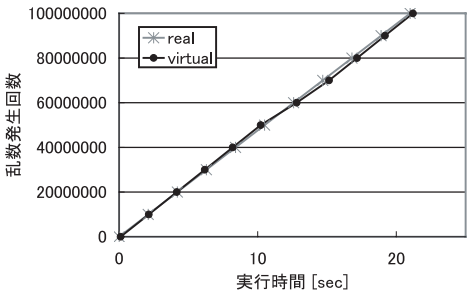

図 5 ジョブ内部の挙動

か，アプリケーション全体の挙動を再現できているか検証する．アプリケーション内にチェックポイントを設け，実システムとシミュレーションのチェックポイント通過時刻を比較する．

1 ジョブあたり 10^8 回の乱数生成を行い，ジョブ内部の挙動を調べる際には乱数生成を 10^7 回行うごとにチェックポイントを設けた．アプリケーションの挙動を調べる際にはチェックポイントをジョブ開始・終了時点に設け，32 個のジョブを処理した．

ただし，現時点では listen，connect，accept，select といったイベントスケジューリング機能が実装されていないため，接続が終了した時点を基準として評価する．また，gettimeofday 以外のイベントは成功するものとして扱っており，現時点ではフィードバックしていない．

実システムおよび Anastasia を実行する計算機を**表 1** に示す．Anastasia は 1 台でイベントスケジューラと ns を実行させ，残りの 2 台でアプリケーションを実行するという構成で評価した．

4.3 結　　果

モンテカルロ法で π を求めるアプリケーションを実システムと Anastasia 上で実行した．それぞれの環境でのチェックポイント通過時刻を**図 5** に示す．ただし，図中では 16Worker 中 1Worker のみを表示している． 図 5 より，乱数発生回数の増加の傾きが似ていることが分かる．チェックポイントでの誤差は平均で 7.7%だったが，計算開始直後は誤差が大きいが次第に安定し，±3 ～ 4% の間に収まるという傾向がみれた．これは，イベントのスケジューリング精度を決

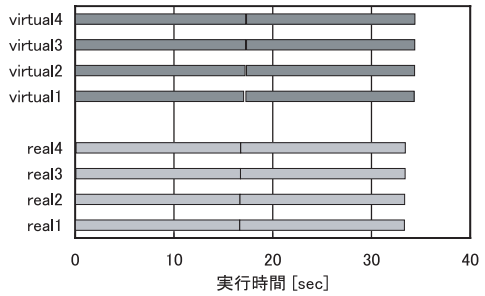

図 6 アプリケーションの挙動

める CPU 消費時間の精度が $10m$ sec と粗いため，アプリケーションを開始した直後はその誤差の影響が大きいと考えられる．

アプリケーションの挙動を**図 6** に示す．この図より，ジョブが終了してから次のジョブが投入されるまでの時間の精度が低いことがわかる．この区間の通信の誤差は 72%であった．誤差の原因については，原因を調査中である．

アプリケーションの挙動を検証した際の実システムでの実行時間は 34.93 秒だった．Master はほとんど計算を行っていないので，アプリケーション全体の計算時間は 550 秒程度（34×16）である．シミュレータではこの計算を 2 台で行うので，タスク間のデータ依存等を考慮しなければ 275 秒程度で終わるはずである．実際には 295.84 秒でシミュレーションが終わった．また，イベントスケジューリングのためにタスクが停止した時間の平均は 11.72 秒となった．これらより，listen 等のイベントスケジューリングを行っていない等の問題点はあるものの，現時点では効率の良いシミュレーションを行えていることを確認した．

5. 関 連 研 究

メガスケール環境と同様に大規模な計算能力を得るためのシステムとして Grid があり，Grid 環境のような分散システム/グローバルコンピューティングをシミュレーションし，スケジューラやタスク配置等を評価する方法がいくつか提案されている．

Grid 評価環境として，MicroGrid[2),3)] が挙げられる．クラスタ計算機上に Globus[4)] ベースの仮想 Grid システムを構築する Grid エミュレータである．Anastasia と同様に実プログラムの関数を置き換えてそのまま実行し，詳細な情報を得るというアプローチである．イベントスケジューリングは Anastasia ようにイベント単位でシミュレーションするのではなく，数 ms ごとのバリア同期によって分散された時刻の同期をとっ

ている．

Grid 環境の基本的なコンポーネントをモデル化してシミュレーションをおこなう方法として，Bricks[5),6)]，Simgrid[7)]，gridsim[8)] が挙げられる．Bricks はグローバルコンピューティングのスケジューリングアルゴリズムの評価システムである．ネットワークトポロジ，計算サーバ，通信モデルをモジュール化し，パラメータを利用して実環境で計測される計算ホストやネットワークの性能や外乱をパラメータとして設定し，グローバルコンピューティングのスケジューリング手法に対する評価を行なうモデルを提案している．

Simgrid は分散アプリケーションのためのスケジューリングアルゴリズム評価システムであり，makespan と呼ばれるアプリケーション開始から全タスク終了の時間を最小にすることを目指してリソースへのタスクの配置をシミュレーションする．ただし，スケジューリングアルゴリズムはシミュレーション開始前に決定する静的なものであり，将来の変動や動的な変化の評価は行えない．

gridsim も，Simgrid と同様にアプリケーションやタスクをモデリングし，アルゴリズムやタスク配置を評価するためのシミュレーションツールである．Simgrid ではデベロッパがモデルを設計しなければならないが，girdsim は並列アプリケーションのモデルやタスクモデルを多数用意している．

評価環境ではないが，Grid 環境の動的な変化を取り入れたジョブの所用時間算出手法 SKED[9)] がある．問い合わせのあったジョブがその時点で各資源から得る事ができるパフォーマンスの変動を考慮した手法である．ただし，ジョブが割り当て済みでさらに情報が参照できるジョブの将来の変動であり，将来発生するジョブや外乱までは考慮していない．

6. おわりに

本稿では，メガスケールシミュレータ Anastasia における詳細シミュレーションについて述べた．アプリケーションを実行して情報収集を行い，それをもとに仮想メガスケール環境を構築した．さらに結果をアプリケーションに対してフィードバックすることでメガスケール環境の提供を行った．メガスケール環境を構築する際にはイベントスケジューラ，および ns を用いてイベント単位でシミュレーションを進めることで仮想メガスケール環境を構築できることを示した．

今後は，現状でのイベントスケジューラに足りない listen，connect，accept 等のイベントを実装し，イベントスケジューリングの内容および精度を高めていく．

本稿で用いた方式は，アプリケーションの実行レベルでの詳細な情報を得られるシミュレーションである．しかし，メガスケール環境の全てをこの方法でシミュレーションすることは，イベントスケジューラや ns の負荷が増大するため難しい．また，アプリケーションを実行するための環境を用意できない．そこで，タスクやネットワークの詳細な挙動を再現する現在の方式に加えて，これらをモデル化してシミュレーションするモデルシミュレーション方式も取り入れることも検討している．

謝辞 本研究の一部は科学技術振興事業団・戦略的基礎研究事業「低電力化とモデリング技術によるメガスケールコンピューティング」による．

参考文献

1) ns(Network Simulator): http://www.isi.edu/nsnam/ns/.
2) Song, H. J., Liu, X., Jakobsen, D., Bhagwan, R., Zhang, X., Taura, K. and Chien, A. A.: The MicroGrid: a Scientific Tool for Modeling Computational Grids, *Supercomputing* (2000).
3) Xin Liu, H. X. and Chien, A. A.: Network Emulation Tools for Modeling Grid Behavior, *The 3rd IEEE/ACM International Symposium on Cluster Computing and the Grid (CCGrid 2003)* (2003).
4) Globus: . http://www.globus.org/.
5) 中田 秀基 松岡 聡 長嶋 雲兵竹房 あつ子: グローバルコンピューティングシミュレータの概要, 情報処理学会研究報告, Vol. 99, No. 21, pp. 31–36 (1999).
6) Aida, K., Tekefusa, A., Nakada, H., Matsuoka, S., Sekiguchi, S. and Nagashima, U.: Performance Evaluation Model for Scheduling in Global Computing Systems, *The International Journal of High Performance Computing Applications*, Vol. 14, No. 3, pp. 268–279 (2000).
7) Casanova, H.: Simgrid: A Toolkit for the Simulation of Application Scheduling, *Proceedings of the First IEEE/ACM International Symposium on Cluster Computing and the Grid (CCGrid 2001)* (2001).
8) Murshed, M. and Buyya, R.: Using GridSim Toolkit for Supercharging Grid Computing Education (2002).
9) 弓場 敏嗣上田 清詩: Grid 計算環境における予定ガント図を用いたジョブスケジューリング, 並列処理シンポジウム JSPP2002, pp. 201–208 (2002).