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

2004－HPC－99 (13)
2004／7／30

メガスケールシミュレータAnastasiaにおける高精度タスクモデルシミュレーション

門島正和[†] 鈴木雄大[†]
柴田俊介[†,☆] 中島浩[†]

我々の研究グループでは，100万台規模の汎用プロセッサ群を利用したメガスケールコンピューティングに関する研究を進めている．その中の1つに，メガスケール環境を仮想的に構築し，タスクスケジューリング機構やアプリケーションの検討及び改善を行うためのシミュレータAnastasiaの開発がある．このAnastasiaには，現在までに，実タスクを用いた精度の高い詳細実行モードとタスクモデルを用いた計算量の小さい簡略実行モードが実装されている．

そこで本稿では，これら2つのモードの長所を生かし，高精度なタスクモデルを生成して高速なシミュレーションを行う新たな実行モードの提案と設計を試みた．また，本モードの予備評価として，πの解法及び16-queen問題における実システムとの実行時間の比較を行った．その結果，実システムでは93秒かかる処理を，平均誤差1.1%という十分な精度を持ちながら，1.9秒で処理できることを確認した．

Accurate model-base simulation for megascale system simulator Anastasia

MASAKAZU KADOSHIMA,[†] TAKEO SUZUKI,[†] SYUNSUKE SHIBATA[†,☆] and HIROSHI NAKASHIMA[†]

We are developing a simulator named 'Anastasia' for megascale computing systems as a easily usable development and evaluation tool. Anastasia has two methods to execute applications; 'emulation mode' to capture detailed application behavior based on real executions; and 'model simulation mode' to grasp application behavior quickly using abstracted task models.

In this paper, we propose a new execution mode named 'accurate model simulation mode', in which a user defined model is modified by sample emulations of real task programs. We also evaluate the preciseness and execution speed of the new mode comparing with a real execution. As a result, our simulator completes its job in 1.9 second for a parallel program that takes 93 second in real world. It is also shown that the simulation error is only 1.1 % in average.

1. はじめに

現在，世界中では，地球環境シミュレータやゲノム情報解析器など，最高峰のスーパーコンピュータ20台分以上に匹敵する高度な計算能力の必要な大規模なアプリケーションが数多く提案されている．このようなアプリケーションの実装をより現実的に低いコストで達成するため，我々の研究グループでは，比較的安価とされる汎用プロセッサを100万クラスのオーダでネットワーク結合した大規模並列システムによる分散計算，すなわち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．

大規模並列環境では，多数のプロセッサやネットワークなどが複雑かつ不均質に絡み合っており，各タスクの配置と実行のタイミングがシステム全体の性能に大きく関わる．そのため，効率的な並列コンピューティングを行う際には，タスクのスケジューリング戦略について十分に検討・改善を繰り返すことが重要となる．しかし，現状においてメガスケール環境を実際に構築して検証を行うことは非常に困難である．また，実環境では全く同じ挙動を繰り返し再現することは不可能に近い．そこで，我々は仮想的にメガスケール環境を構築し，アプリケーションやスケジューリングの評価を行うためのシミュレータAnastasiaの開発を進めている．[1)2)]

† 豊橋技術科学大学
Toyohashi University of Technology
☆ 現在，株式会社パナソニックモバイル金沢研究所
Presently with Panasonic Mobile Communications Kanazawa R&D Lab.

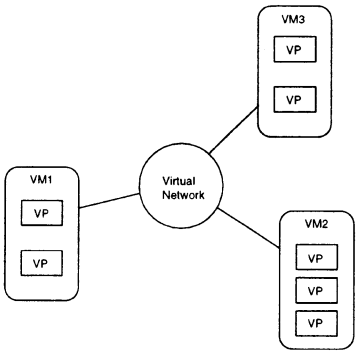

図 1　Anastasia の構成

現在，Anastasia には，実タスクを用いた詳細実行により精度の高いログを取得し，検証を行うエミュレーションモードと，タスクの抽象モデルを用いた簡略実行により全体の計算量を削減して高速化を図るモデルシミュレーションモードの 2 種類の実行モードが実装されている．そこで本稿では，次のステップとして，この両モードの長所を生かし，精度の高いタスクモデルを用いた高速シミュレーションを行うための新たな実行モードを提案する．

以降，2 章で Anastasia 及び各実装モードの概要，3 章で高精度モデルシミュレーションの設計，4 章で同モードの予備評価，5 章で本研究と関連のある Grid シミュレータについて，6 章で今後の展望と結論をそれぞれ述べる．

2. メガスケールシミュレータ Anastasia

Anastasia は，使用する各プロセッサやネットワークの仕様に基づき，仮想的にシステム環境を構築し，アプリケーションの挙動を再現・評価するシミュレータである．その特徴について以下に挙げる．

2.1 システムのモデル化

Anastasia の構成モデルを図 **1** に示す．Anastasia では，このように，計算機を Virtual Machine(以下 VM と表記)，アプリケーション等の実行プロセスを Virtual Process(以下 VP と表記)，ネットワークを Virtual Network に置き換え，システム環境のモデル化を行っている．これにより，1 台の実マシン上で複数のマシンが動作しているように見せかけ，少規模なクラスタでも大きなシステム環境の検証を可能にしている．

2.2 ワークロード

Anastatia のワークロードは、我々の研究グループで開発しているタスク並列スクリプト言語 MegaScript[3] で記述された階層的な並列プログラムである。MegaScript はタスクと呼ぶ逐次または並列の外部プログラムを多数並列実行するための言語であり、タスク間の通信は標準入出力をストリームと呼ぶ通信路を介して行われる。ストリームの入出力端には複数のタスクを接続することができ、入力端ではメッセージがマージされ、出力端ではマルチキャスト配信される。

2.3 イベントスケジューラ

Anastatia のイベントスケジューラは、各タスクが行うストリーム通信のタイミングやバイト数、および CPU 計算時間をイベントとして収集し、それらをターゲットシステムの環境に写像してスケジューリングを行う。後述するモデルシミュレーションモードでは、スケジューラに与えるイベントは以下の 3 種類である。

- input(n) ： ストリームからの n-byte 入力
- output(n) ： ストリームへの n-byte 出力
- compute(t) ： 大きさ（時間）t の計算

上記のうち input と output はスケジューラ内部で TCP/IP の通信に変換され、送受信や select などのイベント列に置換される。またエミュレーションモードでは、TCP/IP レベルの通信イベントが直接収集される。

通信イベントによって生じるターゲットシステムのネットワーク通信における遅延は、ネットワークシミュレータ ns[4] を用いて再現する。イベントスケジューラは ns と協調動作し、ネットワーク内部での通信の進行と計算ノードの計算の進行を調整しつつ、各種イベントのターゲットシステムでの生起時刻を決定する。

2.4 シナリオ

シミュレーションを行うための環境情報，すなわち VM/VP の各仕様や数量，ネットワークの接続状況などに関しては，ユーザが記述するシナリオによって与えられる．この情報に基づき，スケジューラは VM と実マシンの処理速度差の補正や ns のネットワークパラメータの設定などを行う．

また，モデルシミュレーションモードでは，それぞれの計算ノードにおける計算をモデル化した，特殊な記述をシナリオに用いることができる．この記述モデルは図 **2** に示したように，MegaScript のタスク特性を表す記述 (メタプログラム) のサブセット仕様によって表現される．なお，図中の input, output, compute は，前述のイベント情報として直接変換される関数である．また，FOR n⟨block⟩ END は，n 回の繰り返しを表現する制御構文となっている．この他にも，確立 r で条件が真となることを意味する IF ⟨block⟩ END が現在用意されている．

```
def behavior
  input(4)
  FOR n
    compute(100)
  END
  output(4)
end
```

図 2　ビヘイビア記述

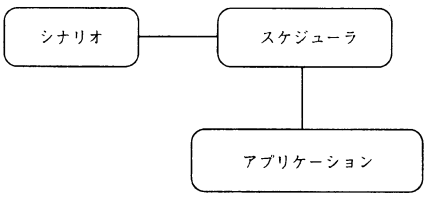

図 3　エミュレーションモードの構成

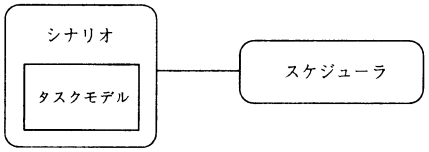

図 4　モデルシミュレーションモードの構成

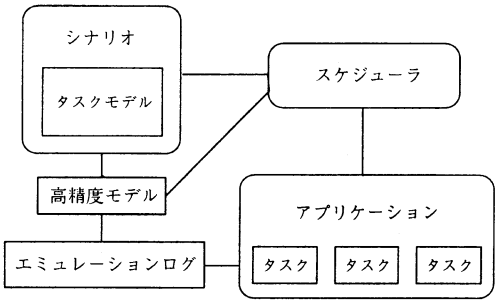

図 5　高精度モデルシミュレーションモードの構成

2.5　実行モード

Anastasia には，様々な検証に対応するため，現在 2 種類の実行モードが実装されている．

2.5.1　エミュレーションモード

タスクの挙動を詳細かつ正確に知るため，現実にタスクの実行を試み，各場面における入出力バイトや計算時間，通信遅延などの情報を取得する．エミュレーションモードの構成を図 3 に示す．

しかし，このモードでは，アプリケーションを実際に動作させることになるため，詳細な挙動が確認できる半面，大規模環境のエミュレーションには膨大な計算量が必要となる．そのため，このまま本シミュレータをメガスケール環境の再現に用いるのは難しい．また，システム全体の流れを大まかに知りたい場合などにも利便性に欠ける．

2.5.2　モデルシミュレーションモード

前述のエミュレーションモードの問題を解消するため，タスクのビヘイビアをモデル化して記述できるようにシナリオを拡張し，それを基に，タスクの簡略実行を行うことのできるモードが実装されている．このモデルシミュレーションモードの構成を図 4 に示す．

シナリオのみを参照して実行を行うため，非常に高速にシミュレーションを行うことができる．しかし，このモードでは，入出力バイトや計算時間などの取得もユーザのモデル記述に頼ることになるため，エミュレーションモードと比較すると，結果の精度は相対的に落ちてしまう．

3.　高精度モデルシミュレーションモード

Anastasia に実装されている 2 つの実行モードは，それぞれ実行精度と計算量に関して，相反する長所と短所を併せ持っていた．ここでは，これら両モードの長所を生かし，より現実とのギャップを抑えた高速シミュレーションを行う，新たな実行モードの設計を試みることにした．

このモードの構成を図 5 に示す．具体的には，まずタスクの種類ごとにサンプルタスクを撰択して，エミュレーションを行う．そして，エミュレーションによって生成したログにより詳細な入出力バイト数や計算時間などを取得して，タスクモデルのビヘイビアにおける不明瞭な部分を明らかにすることでタスクモデル全体の精度を高める．詳細について以下に述べる．

3.1　サンプルタスク

一般に，シミュレーション対象のプログラムは複数の種類のタスクで形成されており，各タスクのインスタンスも複数生成される．そこでまず，タスクの種類ごとに特定のインスタンスを撰択し，それをサンプルタスクとする．次に，タスクモデルに付加された実行ファイル情報に基づき，サンプルタスクを実際に生成してエミュレーション実行を行う．この際の入力パラメータなどは，真のパラメータ集合の中からランダムに選択する．

3.2　高精度モデルの生成

サンプルタスクはエミュレーションの対象となり，その実行結果は 2.2.1 節で述べたイベント情報の羅列

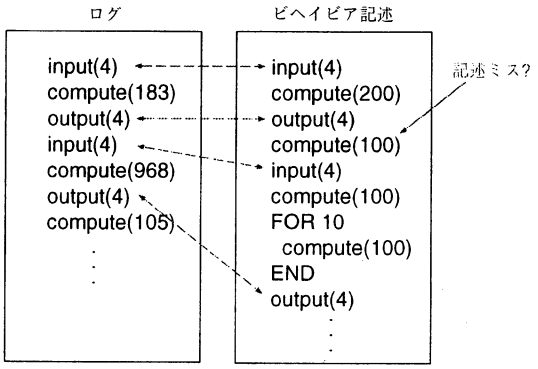

図 6　ログとビヘイビア記述の対応付け

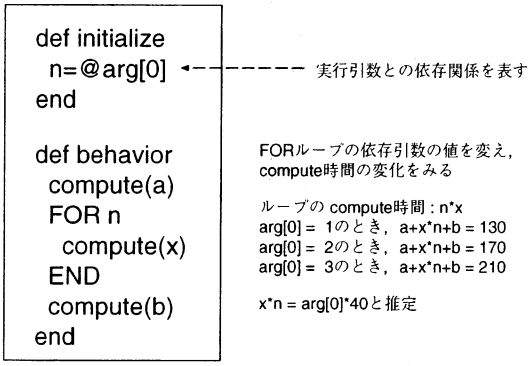

図 7　FOR ループの compute 時間の推定例

として，図 6 左図のようにログファイルへ出力される．そして，ログファイルとシナリオのビヘイビア記述を input / output 命令を指標にして比較し，これを元に各命令の対応を推定する．もし，ログとビヘイビア記述の内容が一致したものであると推測できれば，ビヘイビア記述では不明瞭であった input 時の入力バイト数や compute 時の計算時間などの情報を，対応するログ部分から取得して精緻化することが可能となる．また，どのような解釈を行ってもログとビヘイビア記述が一致しないようであれば，ユーザへ記述の修正を提案する．

3.3　高精度サンプルモデルの応用

次に，サンプルタスクの高精度モデルを基に，同種の全てのタスクインスタンスについて，高精度なモデル化を行いたい．しかし，同じビヘイビアを持っているとはいえ，全てのタスクが同じ挙動を示すことは実質上有り得ない．そこで，タスク実行時の状況を引数によって表し，その実行引数と各ビヘイビアとの依存関係を記述できるようにシナリオを拡張した．これにより，それぞれの細かな挙動の違いを表現できるようになった反面，サンプルモデルを利用したタスクモデルの詳細化には更に工夫が必要となった．以下に，タスク間の相違を予測して詳細化を行う方法を提案する．

3.3.1　サンプルシミュレーション

スケジューリングの検討を行う際，各タスクの compute 時間は最も重要な判断材料の 1 つとなる．同じビヘイビアを持ったタスク同士でも compute 時間に大きく差が生じる場合があるが，これは実行引数に依存した for ループや if 分岐などの制御構文の存在することが主な原因となっている．そのため，各タスクの実行引数が，ビヘイビア内の制御構文の compute 時間にどのような影響をもたらすのか予め把握しておくことが重要となる．

しかし，エミュレーションによって抽出されたログの構成によっては，図 7 のように，1 回の試行で制御構文の compute 時間が算出できない場合がある．本研究では，実行引数を変えながらタスクエミュレーションの試行を数回繰り返し，compute 時間を検証することで，各実行引数において compute 時間がどのように変化するかを推定するサンプルエミュレーションを行うことにした．

4.　評　　価

ここでは，高精度モデルシミュレーションモードの予備評価として，アプリケーション全体をエミュレーションし，そのログを用いてモデルを作成した場合の実際の並列アプリケーションの動作との比較を行う．

4.1　評 価 方 法

評価方法として，モンテカルロ法を用いた π の解法と 16-queen 問題の 2 つの並列アプリケーションについて，共に 1 台の master と 16 台の worker を用い，現実のシステムと高精度モデルシミュレーションモードの動作比較を行った．なお，実システムでは同じ問題を 5 回繰り返し，その平均を動作とした．また，π の解法では worker の試行回数が 5×10^7 回に到達するごとに時刻を出力，16-queen 問題では worker が 5 万回解を見つけるごとに master に通知を行うというチェックポイントを用意して結果をサンプリングし，比較の参考にした．

評価時の実システムと仮想環境のシミュレーション用の計算機のスペックを表 1，システム構成を表 2 に示す．

4.2　結　　果

まず，実システム実行とエミュレーションの全チェックポイントにおける経過時間の平均誤差，最大誤差，及び実行時間比を表 3 に示す．どちらの問題について

表 1 各環境のスペック

	π の解法	16-queen
並列化	1 Master, 16 Worker	
チェックポイント	10 箇所	解 5 万個毎
平均実行時間 [sec]	93	45

表 2 評価システムの構成

	実システム エミュレーション
CPU	Intel Pentium III 1GHz
メモリ	256MB
接続網	1Gbps Ethernet
OS	Vine Linux
コンパイラ	gcc 2.91.66

表 3 エミュレーション時の誤差と実行時間比

	π の解法	16-queen
平均誤差 [%]	1.1	4.1
最大誤差 [%]	1.64	1.2
実行時間比 [%]	1.15	1.4

表 4 実システムと高精度モデルシミュレーションにおける実行時間

	π の解法	16-queen
実システム [sec]]	93	45
高精度モデルシミュレーション [sec]]	1.9	1.91

も平均誤差は 1 秒以下であった．また，5 回の実行の平均を取る際，実システムの結果のぶれは平均 0.8 秒，最大 3 秒となっていたことからも，エミュレーションが高い精度で実現できていることが伺えた．

このエミュレーションによって生成されたログを用い，エミュレーションの挙動を完全な再現したビヘイビアモデルを構築した．この時，実システムとのチェックポイントでの誤差は，エミュレーションを行った場合と変化がなかった．また，実システムと高精度モデルシミュレーションの各問題における実行時間を表 4 に示す．これにより，高速なシミュレーションを実現できることが分かった．

5. 関連研究

メガスケールコンピューティングと同じように大規模な並列環境を利用して，巨大な計算能力を得るためのシステムとして Grid があり，いくつかの Grid シミュレータが提案されている．

5.1 エミュレータ

MicroGrid[5),6)] は globus[7)] や TCP/IP ソケット通信を用いた Grid 用アプリケーションをクラスタに分散して実行するというエミュレーション方式をとっている．MicroGrid では仮想的な Grid リソースや GIS☆を提供しており，この上でアプリケーションを実行することで挙動を高精度に再現している．しかし，Anastasia のエミュレーションモードと同様に実環境とシミュレータにおいてアプリケーションの計算量は変化しないため，スケーラビリティの点で大きな課題がある．

5.2 モデルシミュレータ

Grid 環境で実行するアプリケーションをモデル化してシミュレーションする方法として，Simgrid[8)]，GridSim[9)]，Bricks[10)] が挙げられる．これらはタスクを計算時間や通信量などの値を用いてモデル化することでシミュレーションにかかる計算量を削減しているため，高速に大規模アプリケーションをシミュレーションすることができる．しかし，いずれのシミュレータにおいてもタスクのモデル化はユーザに委ねられており，高精度なモデルを得られるという保証はない．そのため，シミュレーション自体の精度も不安定なものとなってしまう．

Simgrid は分散アプリケーション向けスケジューリングアルゴリズムの評価システムで，タスク中の計算・通信部分の依存関係を有向非周期グラフで表し，このグラフをタスクモデルとして利用することで高速なシミュレーションを実現する．ネットワークのトポロジなどの情報を抽出するためのツール ENV[11)] を利用しており，これにより高精度なネットワークシミュレーションを実現している．また，分散スケジューリング手法やルーティングテーブルの変化が与える影響もシミュレーション可能となっている．ただし，スケジューリングアルゴリズムは静的なものであるため，環境が変化した場合についての評価は行えない．

GridSim も Simgrid と同様にアプリケーションをモデル化し，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 ではユーザ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やタスクのモデルを多数用意している．

Bricks はシミュレーション環境やスケジューリングシステムなどをモジュール化し，このモジュールを置き換えることで，様々なシステムの機能試験を実施することができる．現在はホストやネットワークの性能，変動などの指定や NWS☆☆モジュールの組み込みが可能となっており，これにより様々な場面におけるスケジューリング手法の評価が可能となっている．

6. おわりに

本稿では，正確な挙動を持つタスクモデルを生成し，Anastasia の新しい実行モードとして，高精度モデル

☆ Grid Information Service

☆☆ Network Weather Service

シミュレーションモードを提案し，設計を試みた．また，実システムと同モードによるアプリケーションの実行結果を比較し，その有用性を検証するための予備評価を行った．

これにより，Anastasia で並列マシンにおけるメガスケール環境の詳細な検討・改善を効率的に行うことの可能性を示すことができた．

さらに，他の関連研究の観点からも，本研究内容が有用であることを示した．

しかし，このモードではモデルの解釈次第で性能が大きく左右されるため，プログラム構造の様々なパターンに対するログとの一致解釈の方法やサンプルシミュレーションの試行回数の判断基準など，それぞれの場面において，どのような処置が適切となるのか，検討すべき点は数多く残されている．これらの対処については，今後十分な議論が必要になると考えられる．

謝辞 本研究の一部は，科学技術振興機構・戦略的想像研究推進事業「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) 柴田俊介, 大野和彦, 中島浩: 大規模分散計算環境シミュレータの設計と実装, 情報処理学会研究報告, 2002-HPC-91, pp. 173–178 (2002).
2) 鈴木雄大, 柴田俊介, 大野和彦, 中島浩: メガスケール環境シミュレータ Anasitasia における詳細シミュレーション, 情報処理学会研究報告, 2003-HPC-95, pp. 155–160 (2003).
3) 大塚保紀, 大野和彦, 中島浩: タスク並列スクリプト言語 MegaScript によるタスクモデル記述, 情報処理学会研究報告, 2003-HPC-95, pp. 113–118 (2003).
4) Breslau, L., Estrin, D., Fall, K., Floyd, S., Heidemann, J., Helmy, A., Huang, P., McCanne, S., Varadhan, K., Xu, Y. and Yu, H.: Advances in Network Simualtion, *Computer*, Vol. 33, No. 5, pp. 59–67 (2000).
5) 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).
6) Liu, X. and Chien, A.: Traffic-based Load Balance for Scalable Network Emulation, *in Proceedings of the ACM Conference on High Performance Computing and Networking, SC2003* (2003).
7) Foster, I. and Kesselman, C.: Globus: A Metacomputing Infrastructure Toolkit, *The International Journal of Supercomputer Applications and High Performance Computing*, Vol. 11, No. 2, pp. 115–128 (1997).
8) 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), May 15-18, 2001, Brisbane, Australia.* (2001).
9) Buyya, R. and Murshed, M.: GridSim: A Toolkit for the Modeling and Simulation of Distributed Resource Management and Scheduling for Grid Computing, *The Journal of Concurrency and Computation: Practice and Experience (CCPE), Wiley Press, May 2002 (to appear)* (2002).
10) 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).
11) Shao, G., Berman, F. and Wolski, R.: Using effective network views to promote distributed application performance, *International Conference on Parallel and Distributed Processing Techniques and Applications* (1999).