計算機アーキテクチャ 132－6
システムソフトウェアとオペレーティング・システム 80－6
ハイパフォーマンスコンピューティング 75－6
（1999．3．4）

グローバルコンピューティングシミュレータの概要

竹房 あつ子[†1] 合田 憲人[†2] 中田 秀基[†3]
松岡 聡[†2] 長嶋 雲兵[†4]

グローバルコンピューティングシステムが複数提案される一方，グローバルコンピューティングシステムのスケジューリング手法に対する考察が不十分である．これは大規模かつ再現性のある評価実験が困難であることに起因する．本稿ではグローバルコンピューティングシステムのスケジューリングアルゴリズムとそのフレームワークのための評価基盤を提供するシミュレータBricksを提案する．Bricksでは様々なシミュレーション環境やスケジュールアルゴリズムおよびスケジューリングに関するモジュールを設定可能である．また，これらのモジュールを既存グローバルコンピューティングシステムモジュールに置き換えることで，Bricks上での既存システムの機能試験を実施することができる．他システムのBricksへの組み込み例としてリソース情報の予測システムであるNWSを用いて本システムの評価実験を行ったところ，Bricksが実環境と同等の挙動を示すことを確認した．さらに，NWSがBricks上で正常に動作したことから，Bricksが既存の外部モジュールに対して機能試験環境を提供できることを示した．

Overview of a Global Computing Simulator

ATSUKO TAKEFUSA,[†1] KENTO AIDA,[†2] HIDEMOTO NAKADA,[†3]
SATOSHI MATSUOKA[†2] and UMPEI NAGASHIMA [†4]

While there have been several proposals of high performance global computing systems, scheduling schemes for the systems have not been well investigated. The reason is difficulties of evaluation by large-scale benchmarks with reproducible results. This paper describes an overview of the Bricks simulator that evaluates scheduling schemes on a typical high-performance global computing system. Bricks can simulate various behaviors of global computing systems, especially the behavior of networks and resource scheduling algorithms. Moreover, Bricks is componentalized such that not only its constituents could be replaced to simulate various different system algorithms, but also allows incorporation of existing global computing components via its foreign interface. To test the validity of the latter characteristics, we incorporated the NWS system, which monitors and forecasts global computing systems behavior. Experiments were conducted by running NWS under a real environment versus the simulated environment given the observed parameters of the real environment. Under both environments, NWS behaved similarly, making quite comparative forecasts.

1. はじめに

グローバルコンピューティングは広域ネットワーク上に分散した計算／情報リソースを活用して大規模計算を実現する計算技術であり，近年これを目的としたシステムが複数提案されている[1)]．各システムではグローバルコンピューティングを効率的に行うために，計算リソースの基本性能，利用状況およびリソース間のネットワークの状況をモニタし，それらの情報をもとに遠隔ユーザの要求するタスクを適切に割り当てるスケジューリングフレームワークを実装している．

グローバルコンピューティングシステムにおける各スケジューリング手法の公平な比較と，その特性の調査を行うためには，様々な

- ネットワークのトポロジ，バンド幅，混雑度，変動
- 計算リソースのアーキテクチャ，性能，負荷，変動

等を想定した再現性のある大規模環境での評価が求められる．一方，現在行われている実環境でのスケジューリング手法の評価では評価環境の規模が制約され，再現性もないことにより，他の研究者により開発されたスケジューリングアルゴリズムとの比較が困難である．また，グローバルコンピューティングリソースのモニタ・予測情報はスケジューリングに大きく影響するにも関わらず，開発されたスケジューリングに関するモジュール

†1 お茶の水女子大学
Ochanomizu University
†2 東京工業大学
Tokyo Institute of Technology
†3 電子技術総合研究所
Electrotechnical Laboratory
†4 物質工学工業技術研究所
National Institute of Materials and Chemical Research

の実環境上での機能試験の実施コストが高いという問題がある．

本稿では，グローバルコンピューティングシステムのスケジューリング手法およびそのフレームワークの評価基盤を提供するグローバルコンピューティングシミュレータ Bricks を提案する．Bricks は様々なネットワークトポロジ，計算サーバアーキテクチャ，通信モデルおよびスケジューリングフレームワークの各モジュールを設定可能にする Bricks 環境設定スクリプトをユーザに提供し，(1) スケジューリングアルゴリズムの性能評価と，(2) グローバルコンピューティングシステム開発者が開発するスケジューリングのためのプログラムモジュールの機能試験を可能にする．

他システムのスケジューリングに関するモジュールの Bricks への組み込み例として，グローバルコンピューティングのリソース予測システムである NWS[2),3)] を用いた．本システムと NWS による評価実験では，通信スループットの実測値を用いた通信モデルにより Bricks が実環境とほぼ同等の挙動を示す再現性のある評価環境が提供できることを確認した．また，NWS が実環境上での運用と同様に Bricks 上で正常に動作したことから，Bricks が既存システムのスケジューリングに関するモジュールの機能試験を可能にすることを示した．

以後 **2**では Bricks の概要を説明し，**3**では Bricks のアーキテクチャについて述べる．また，**4**では NWS の Bricks への組み込みを例にした他のグローバルコンピューティングシステムの機能試験のための枠組について説明し，**5**でその評価結果を示す．

2. Bricks の概要

Bricks は Java で実装されたグローバルコンピューティングシステムのスケジューリング手法の性能評価ツールであり，スケジューリングアルゴリズムの評価とそのフレームワークの運用テストを行うための大規模かつ再現性のある評価実験環境を提供することを目的としている．Bricks の特徴は以下のようにまとめられる．

- Bricks シミュレータはグローバルコンピューティング環境とスケジューリングユニットから構成されている（図**1**）．シミュレーションにおける
  - スケジューリングアルゴリズム
  - スケジューリングに関する各モジュール
  - クライアント，サーバ，ネットワークの構成
  - ネットワーク／サーバでの処理方法（待ち行列）
  - シミュレーションで用いられる乱数分布

  等の設定を Bricks 環境設定スクリプトにより自由に組み立てられることから Bricks と名付けた．Bricks のユーザは Bricks 環境設定スクリプトで記述した環境設定を実行時の入力として与えることにより，様々な環境下でのスケジューリングアルゴリズムの評価実験ができる．
- グローバルコンピューティングのスケジューリングにはリソースのモニタ・予測情報が必要不可欠である．Bricks ではグローバルコンピューティングシステムのスケジューリングに関する各モジュールへのインターフェイスを提供し，既存システムモジュールの Bricks 上での機能試験を可能にする．

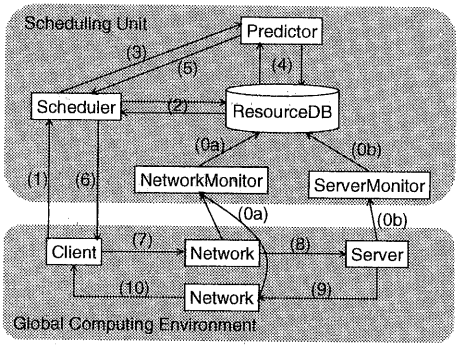

図1 Bricks アーキテクチャ

3. Bricks のアーキテクチャ

Bricks はグローバルコンピューティング環境とスケジューリングユニットにより構成される（図**1**）．

グローバルコンピューティング環境 グローバルコンピューティングをシミュレートする環境を提供する．

スケジューリングユニット グローバルコンピューティングにおけるスケジューリングフレームワークを提供する．また，他のグローバルコンピューティングシステムのスケジューリングフレームワークのためのインターフェイスも提供している．

以下にその詳細な説明を行う．

3.1 グローバルコンピューティング環境

グローバルコンピューティング環境は次のモジュールにより構成される．

Client グローバルコンピューティングにおけるユーザを表し，グローバルコンピューティングのタスクを発行する．

Network ユーザの計算機とグローバルコンピューティングの計算リソースをつなぐネットワークを表す．Bricks ではこのネットワークを待ち行列で表現する．実際のネットワークの挙動を表すために，バンド幅，混雑度とその分散を指定するパラメータを用意している．

Server グローバルコンピューティングの計算リソースである計算サーバを表す．ネットワーク同様にサーバも待ち行列で表す．サーバの挙動を表現するパラメータとして，サーバの性能と負荷，およびその分散がある．

Bricksにおけるクライアントからのタスクのモデル，通信モデル，サーバのモデルについて以下で説明する．

3.1.1 タスクモデル

シミュレーションにおけるタスクにとって重要なことは，通信時間，計算時間がどの程度かかるかということである．現在のBricksの実装では，これらを調べる上で必要な情報

- タスク実行に要する通信量（送信／受信）
- タスクの実行時の演算数

をパラメータとしてタスクを表現する．

3.1.2 通信モデル

Bricksではシミュレーションの実行時のユーザの設定により，様々な通信モデルが実現できる．現在Bricksで表現できる代表的な通信モデルは以下の2通りある．

1つめは，ネットワークにはグローバルコンピューティングシステム以外のシステムから流されるデータ（外乱）があることを想定し，通信スループットを外乱のデータの到着率を変化させて表現するモデル[4]である．図2にBricks環境設定スクリプトでの設定例を示す．`Network`はネットワークの設定を行う宣言であり，名前，ネットワークの待ち行列の種類，外乱のデータの設定を記述する．待ち行列はFCFSで処理されるスループット1500[KB]，バッファ長2の有限バッファ待ち行列を指定している．また，外乱のデータの平均データサイズは100.0[KB]で乱数系列の種が88である指数分布，到着間隔は平均0.008のポアソン到着で決定することを示している．

2つめのモデルは実環境で計測された通信スループットをもとに，ネットワークの待ち行列のスループットを決定するモデルである．図3にその設定例を示す．このモデルではネットワークにFCFSで処理する無限バッファ長の待ち行列を指定し，通信スループットは1000個のデータが含まれている`nws_throughput`ファイルを参照するように指定している．また，外乱のデータを流さないように到着間隔を∞に設定する．

図3の設定では，時刻と通信スループットの離散データから全ての時刻に対する通信スループットを算出するために，3点の時刻と通信スループットのみで補間値を計算できる3次スプライン補間を用いている．補間法もまた，他のアルゴリズムを採用することができる．

3.1.3 計算サーバモデル

計算サーバは現在FCFSで処理することを前提としている．サーバもネットワーク同様に待ち行列で表され，外乱のジョブの平均演算数，到着間隔を指定することによりその稼働率を決定する．サーバも負荷の変動を実環境で測定された数値により表現することができる．

3.2 スケジューリングユニット

グローバルコンピューティングシステムのスケジューリングをサポートする枠組として，Bricksではスケジューリングユニットを提供する．スケジューリングユニットではグローバルコンピューティングのスケジューリングで必要とされる各モジュールを実装している．

NetworkMonitor グローバルコンピューティング環境におけるネットワークの通信スループット，レイテンシ等のネットワークの状況をモニタする．得られた情報はResourceDBに格納する．

ServerMonitor グローバルコンピューティング環境における計算サーバの性能，負荷，稼働率をモニタし，得られた情報はResourceDBに格納する．

ResourceDB グローバルコンピューティングシステムにおける総合データベースである．各モニタからリソース情報が格納され，Predictor，Schedulerに対してその情報を提供する．

Predictor ResourceDBからリソース情報を入手し，そのリソースの可用性を予測する．この予測は，新たに投入されるタスクのスケジューリングに利用される．

Scheduler ResourceDBで管理されている情報とPredictorで予測された情報をもとに，ユーザのタスクを利用可能な計算サーバから最適なサーバに割り当てる．

スケジューリングユニットの各モジュールは容易に組み変え可能である．

3.3 Bricksの実行の流れ

Bricksによるシミュレーションの手順を以下に示す．この手順は図1に対応している．

(0a) NetworkMonitorは定期的にNetworkにプローブパケットを流し，Networkの通信スループット，レイテンシを測定する．測定結果はResourceDBに格納する．

(0b) ServerMonitorは定期的にServerに問い合わせて，Serverの負荷情報を調べる．結果はNetworkMonitor同様，ResourceDBに格納する．

(1) Clientでグローバルコンピューティングのタスクが発生すると，ClientはSchedulerにタスクを投入すべきServerを問い合わせる．その際，ClientはSchedulerにタスクの情報を提供する．

(2) SchedulerはClientのタスクに対してResourceDBにグローバルコンピューティングシステム上で利用可能なServerを問い合わせる．

(3) Schedulerは(2)で問い合わせたServerとそのServerへのNetworkに関するリソース状況の予測値をPredictorに問い合わせる．

(4) PredictorはResourceDBにServerおよびそのServerへのNetworkのモニタ情報を問い合わせ，その情報をもとに各リソース状況を予測する．

(5) Predictorはリソース状況の予測が終了すると，Schedulerに予測値を返す．

(6) Schedulerはこの予測値をもとにタスクを発行するServerを決定し，Clientに通知する．

(7) Clientは決定したServerへのNetworkにタスクを投入する．この際，タスクの送信データは論理パ

```
Network network1 QueueFCFSFiniteBuffer(FixedNumberGenerator(1500.0), 2) \
  OthersData(ExponentRandomGenerator(100.0, 88), ExponentRandomGenerator(0.008, 221))
```

図2 外乱のデータを用いたBricks環境設定スクリプトによるネットワークの設定例

```
Network network1 QueueFCFS(SplineThroughputGenerator(1000, ./nws_throughput)) \
  OthersData(ExponentRandomGenerator(10.0, 9988), InfNumberGenerator())
```

図3 実データを用いたBricks環境設定スクリプトによるネットワークの設定例

ケットサイズに分割する．

(8) Clientから投入されたパケットがNetworkの待ち行列のサーバ上で

[送信データ量]／[Networkのスループット]

時間処理されると，パケットはServerに送られる．

(9) 分割されたタスクのデータがすべてServerに到着すると，Serverでタスクが実行される．Serverの待ち行列のサーバ上で

[タスクの演算数]／[Serverの性能]

時間処理されてタスクの実行が終了すると，Serverはその結果をタスクを発行したClientへのNetworkに投入する．ただし，Clientからの送信時と同様にタスクのデータは論理パケットサイズに分割される．

(10) Serverから投入されたパケットが

[受信データ量]／[Networkのスループット]

時間Networkの待ち行列のサーバ上で処理されると，Clientにパケットが送られる．すべてのパケットがClientに送られると，そのタスクの実行が終了する．

4. 外部モジュール組み込みインターフェイス

Bricksではスケジューリングユニットの各モジュールに対して様々なアルゴリズムを実現したプログラムモジュールのためのインターフェイスを提供する．これらのモジュールを既存のグローバルコンピューティングシステムに置き換えることで，既存システムのBricks上での機能試験が可能になる．

本稿ではその第一歩として，グローバルコンピューティングシステムのリソース状況のモニタと予測を行うシステムとしてUCSDで開発されたNWS（Network Weather Service）の組み込みを行った．NWSはAppLeS，Legion，Globus，Condor等のシステムで予測機構として利用する試みがあり，他システムへのAPIを提供している．Bricksへの組み込みでは，我々が開発したNWSのJava API[5]を用いた．

NWSは次の4つのモジュールにより構成される．

Persistent State 測定された情報を格納するストレージであり，Bricksにおける ResourceDBに対応する．

Name Server 各モジュールのTCP/IPポート番号やIP／ドメインアドレス等の参照に用いる．

Sensor ネットワーク，計算サーバの情報をモニタするもので，Bricksにおける NetworkMonitor，ServerMonitorに対応する．

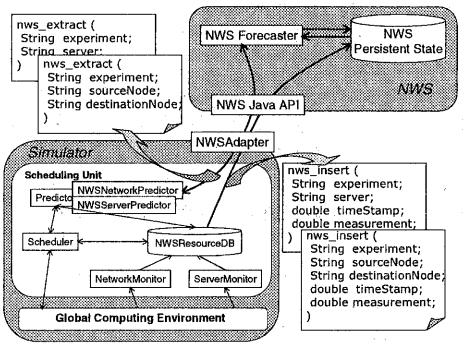

図4 NWSのBricksへの組み込み

Forecaster リソース情報を予測するもので，Bricksの Predictorに対応する．

Bricksに組み込む際には，NWSのPersistent State，Forecasterを用い，Bricks内で測定された情報をPersistent Stateに格納し，Forecasterで予測されたデータをスケジューリングの際に利用する．これらの処理をサポートするモジュールとしてNWSResourceDB，NWSNetworkPredictor，NWSServerPredictorと，NWS Java APIとBricksでのデータタイプの変換を行うNWSAdapterを用意した．

図4にBricksをシミュレータに組み込んだ例を示す．Bricks内でNetworkMonitor／ServerMonitorがモニタした情報をNWSResourceDBに格納すると，NWSResourceDBはNWSAdapter経由でPersistent Stateにその情報を格納する．また，Bricks内で予測情報が必要とされると，NWSNetworkPredictor／NWSServerPredictorがNWSAdapter経由でNWS Forecasterから予測値を取り出すことができる．

5. Bricksの評価実験

評価では，Bricksがスケジューリングアルゴリズムを評価するための再現性のある試験環境を提供し，実際のネットワークの挙動を示す性能評価ツールであることを示す．また，スケジューリングに関する外部モジュールとしてNWSを用い，NWSのBricks上での正常な動作を確認することでBricksが既存システムモジュールの機能試験環境を提供できることを示す．

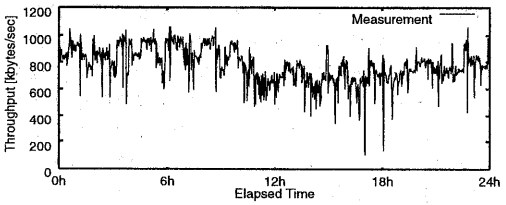

図5　実測における通信スループット（24時間）

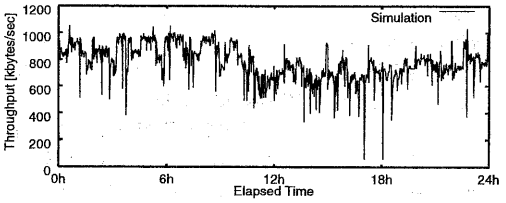

図6　Bricksにおける通信スループット（24時間）

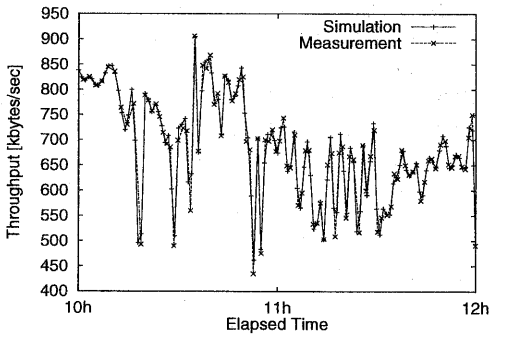

図7　実測とBricksでの通信スループットの比較（2時間）

5.1　評価実験方法

評価では実環境においてグローバルコンピューティングシステムの資源予測フレームワークを提供するNWSを実行して実際のネットワークの変動の測定と予測を行い，それをシミュレータ上で再現する．

まず，東工大と電総研の2つの計算機にNWS Sensorを設定し，2つのサイト間の通信スループット，レイテンシ，サーバの稼働率を測定する．NWS Forecasterには，測定されたネットワークの情報をもとに各タイムステップにおける予測を行わせる．ただし，サーバをモニタする間隔は10[sec]，ネットワーク状況をモニタする間隔は60[sec]とし，通信スループットを測定する際のプローブデータサイズを300[KB]とした．

次に，実環境でNWSにより測定された通信スループット値と3次スプライン補間法を用いた通信モデルを設定し，Bricksによるシミュレーションを行う．シミュレーションでは実際にNWSのPersistent StateとForecasterを実行し，同じモニタ間隔を設定して実環境で測定された通信スループットがBricksで得られるか，Bricks上のNWS Forecasterが実環境同様に予測を行うかどうか調べる．

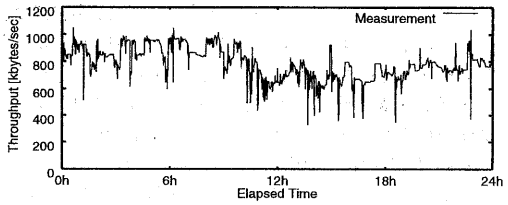

図8　実環境における通信スループットの予測値（24時間）

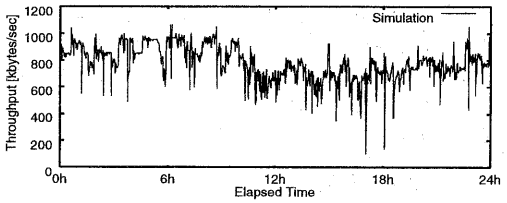

図9　Bricksにおける通信スループットの予測値（24時間）

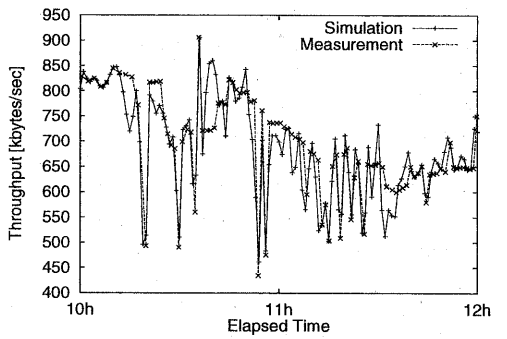

図10　実環境とBricksでの通信スループットの予測値の比較（2時間）

5.1.1　評価実験結果

図5, 6に実環境でNWSのSensorが測定した通信スループットとBricks内でNetworkMonitorが算出した通信スループットを示す．横軸は実時間およびシミュレーション時間を示しており，実測は1999年2月1日月曜日深夜0時から24時間分の測定結果となっている．縦軸には通信スループットを[KB/sec]で示した．

図5, 6より，全体的に実測とBricksのシミュレーション結果はほぼ同様の通信スループットを示していることが分かる．図7は2時間分の実測値とBricksで算出された通信スループットの比較をしたものであり，この図からも実測値とBricksでの通信スループットの算出値の一致が確認できる．実際のネットワークにおけるTCP/IP通信のモデル化の試みは複数行われているが，その挙動は複雑で，特定のパケットに対するモデル化を行っているのが現状である．Bricksではそれらのモデルを採用可能であることは勿論，実環境における通信スループットの実測値を利用することで，実環境に則したグローバルコンピューティングシステム上での通信を再現できることが分かる．

図8, 9に実環境とBricksにおいてNWS Forecaster

が予測した通信スループットを示す．また，**図10**は実測とBricksでの通信スループットの予測値の2時間分のデータを比較をしたものである．**図8, 9**から，通信スループット値と同様に実測とBricksでの予測値の傾向がほぼ一致していることが分かる．NWS ForecasterがBricks上で正常に動作したことから，他のスケジューリングに関する外部モジュールの機能試験がBricks上で実施可能であることが示された．ただし，**図10**では実環境とBricksで出した予測結果が微妙に異なっている．これは実測とBricksでのモニタするタイミングが異なっていたことが影響したと考えられ，実通信スループットを用いたモデルにおける補間値の算出法を検討する必要があることが示唆された．

6. 関連研究

分散システムにおけるスケジューリング手法の研究は以前から行われているが，未実装であったり，実装されている場合でも他の研究者により実装されたスケジューリングアルゴリズムとの比較が非常に難しい．これらの問題に着目したシミュレーションによる評価の試みが，以下のプロジェクトでなされている．

分散システムにおけるスケジューリングアルゴリズムの評価技術としては，フロリダ大で行われているOsculant[6]のOsculant Simulatorがあげられる．Osculantはbiddingポリシーにしたがって計算サーバの性能をあらわすbidsを計算し，bidsが一番高いものを選択するボトムアップリソーススケジューラである．Osculant SimulatorではBricks同様にネットワークのトポロジや計算ノードの設定が柔軟に行える．グローバルコンピューティングシステムのスケジューリングアルゴリズムの評価基盤を提供するものではない．

WARMstonesは未実装であるが，Bricks同様スケジューリングアルゴリズムの評価基盤としてバージニア大で提案されているシステムである．シミュレーションにおけるタスクやシステムの表現形式，および複数スケジューリングアルゴリズムを柔軟に表現するためのMIL（MESSIAHS Interface Language）やライブラリなど，その基礎技術はMESSIAHS[7]に基づいている．BricksでもMESSIAHSの技術をもとにグローバルコンピューティングでのスケジューリングに適したスケジューリングアルゴリズムの表現言語を提供する予定である．ただし，WARMstonesでは既存のグローバルコンピューティングのスケジューリングフレームワーク技術に対する評価基盤の提供は考えられていない．

7. まとめ

本稿では，グローバルコンピューティングシステムのスケジューリング手法およびそのフレームワークの評価基盤を提供するグローバルコンピューティングシミュレータBricksを提案した．Bricksはネットワークトポロジ，計算サーバアーキテクチャ，通信モデルおよびスケジューリングフレームワークの各モジュールを設定するためのBricks環境設定スクリプトを提供し，様々なスケジューリングアルゴリズムの性能評価を可能にする．また，グローバルコンピューティングのスケジューリングを行う上で不可欠なリソース状況のモニタ・予測情報の提供をサポートする既存システムモジュールの機能試験環境を提供する．

BricksとグローバルコンピューティングのリソースNWSを用いた評価実験では，実測通信スループットを用いた通信モデルを採用することでBricksが実際の環境に則したグローバルコンピューティング上での通信を再現できることを確認した．また，Bricks上でNWSのPersistent State, Forecasterが実環境上と同様に動作したことから，Bricksが既存のスケジューリングに関する各モジュールの機能試験環境を提供できることを実証した．

今後はタスク／通信／サーバモデルをより洗練していく．Bricksではグローバルコンピューティングシステムで実行される様々なアプリケーションに対応する必要があるため，並列タスクの表現等のタスクモデリングが必要である．サーバのモデリングに対しては，タイムシェアリングなどの異なるタスクの処理方式や，マルチプロセッサ等のアーキテクチャへの対応を行う．また，Bricksを用いてグローバルコンピューティングシステムにおける適切なスケジューリングアルゴリズムの調査，考案を行う．

参考文献

1) Foster, I. and Kesselman, C.(eds.): *The Grid: Blueprint for a New Computing Infrastructure*, Morgan Kaufmann (1998).
2) NWS. `http://nws.npaci.edu/NWS/`.
3) Wolski, R., Spring, N. T. and Hayes, J.: The Network Weather Service: A Distributed Resource Performance Forecasting Service for Metacomputing, Technical Report TR-CS98-599, UCSD (1998).
4) 竹房あつ子, 合田憲人, 小川宏高, 中田秀基, 松岡聡, 佐藤三久, 関口智嗣, 長嶋雲兵: 広域計算システムのシミュレーションによる評価 – Ninf システムの広域分散環境でのジョブスケジューリング実現に向けて –, 並列処理シンポジウム JSPP'98 論文集, pp. 127–134 (1998).
5) NWS Java API. `http://ninf.etl.go.jp/~nakada/nwsjava/`.
6) Osculant. `http://beta.ee.ufl.edu/Projects/Osculant/`.
7) Chapin, S. J. and Spafford, E. H.: Support for Implementing Scheduling Algorithms Using MESSIAHS, *Scientific Programming*, Vol. 3, pp. 325–340 (1994).