大規模環境向け情報共有手法を用いた分散ジョブスケジューリングシステム

梅田典宏[†]　中田秀基[††,†]　松岡聡[†,†††]

グリッド環境では、ジョブスケジューリングシステムによって分散した計算機を統合した資源として扱うことを可能にしている。しかし既存のシステムは、スケジューリングに必要な資源情報収集および実行ジョブと資源のマッチングを少数の計算機で行うことによる単一故障点の存在と資源・投入ジョブ数の増加に対するスケーラビリティの欠如という問題を抱えている。我々は、大規模環境向けの通信手法を用いて資源情報を共有し、耐故障性と資源数の増加に対しスケーラブルな分散ジョブスケジューリングシステムを提案する。シミュレータを用いた従来システムとの比較評価によって、本提案手法が大規模環境下でより効率的なスケジューリングを可能にすることを示す。

Decentralized Job Scheduling System based on Information Sharing Framework for Large-Scale Computing Environment

Norihiro Umeda,[†] Hidemoto Nakada[††,†]
and Satoshi Matsuoka [†,†††]

Job scheduling system makes distributed computer into integrated resource. However these systems has a single point of failure that just a few computers makes assignments job to resources, and lack of scalability to increase number of resources and jobs. We claim decentralized job scheduling system to share resources status using communication framework for large-scale computing environment. The evaluation using our simulator shows that our proposal performs more effective scheduling than traditional job scheduling system.

1. はじめに

ネットワークで結ばれた広域に分散する多数の計算機を統合し、仮想的な一つの計算機として利用するグリッドが普及しつつある。グリッド環境では、利用可能な計算資源の状態 (アーキテクチャ、CPU 使用率、空きメモリ、ストレージ容量など) とユーザから投入されるジョブの実行に必要、ないしは実行により適した条件の情報を収集し、適切な資源割り当てを行うジョブスケジューリングシステムを用いることで計算資源の効率的な利用を可能にしている。代表的なジョブスケジューリングシステムとして Condor[1), XtremWeb[2)などが存在し、現実に最大 1000 台程度の計算機を集約して利用する環境が提供されている。

これらのシステムでは、計算資源の情報収集およびジョブの割り当てを固定された少数の計算機によって行っており、それらに障害が発生した際には管理下におかれた資源全体の利用が不可能になってしまう。また、情報収集にかかる通信量や資源とジョブのマッチングを求めるコストが少数の計算機に集中することから、今後予想される計算資源および投入されるジョブの増大に対するスケーラビリティに問題がある。

本研究では、コストの低い不完全な情報共有手法によって各計算資源の状態を複数のノードで共有し、スケジューリングを複数の計算機で分散して行うことにより、負荷集中と単一故障点の排除を目指すシステムを提案する。そして、本提案手法と既存のシステムが基盤としているモデルをシミュレーションで比較し、情報の不完全性と計算資源および投入ジョブ数が性能に与える影響を評価した。

2. 関連研究

2.1 Condor

Condor は、ウィスコンシン大学で開発されたジョブスケジューリングシステムであり、組織が所有する遊休計算機の稼働率の向上を目的としている。

Condor では、利用対象となる各計算資源およびそれらを管理する計算機の集合を Condor プールと呼ぶ。Condor プールには、ユーザからのジョブの投入を受け付け、それを適切な計算資源に割り当てる Central

† 東京工業大学
Tokyo Institute of Technology
†† 産業技術総合研究所
National Institute of Advanced Industrial Science and Technology
††† 国立情報学研究所
National Institute of Informatics

```
MyType = “Machine”
TargetType = “Job”
Machine = “nostos.cs.wisc.edu”
Requirements = (LoadAvg ≤ 0.3) &&
               (KeyboardIdle ≤ 15*60)
Rank = target.Department == my.Department
Arch = “INTEL”
OpSys = “LINUX”
Disk = 3076076
Memory = 128
KeyboardIdle = 173
```

図 1 計算資源の状態を表す ClassAd の例

```
MyType = “Job”
TargetType = “Machine”
Requirements = (target.Arch == “INTEL” &&
target.OpSys == “LINUX”)
&& target.Disk ≤ my.DiskUsage
Rank = (Memory * 10000) + KFlops
Cmd = “/home/tannenba/bin/sim-exe”
Owner = “tannenba”
DiskUsage = 6000
```

図 2 ジョブの実行条件を表す ClassAd の例

Manager が一台のみ存在する。

2.1.1 資源情報の収集とジョブの割り当て

Condor におけるジョブと計算資源の割り当ては Matchmaking[3] と呼ばれる手法を用いている。以下にその詳細を述べる。

計算資源を提供するマシンは、アーキテクチャや OS、CPU 使用率、メモリ・ストレージ容量といった情報を定期的に収集しており、各資源の所有者によって定義された利用ポリシーとともに Central Manager に ClassAd と呼ばれるフォーマットで定期的に送信する。図 1 にその例を示す。

Central Manager はプール内に一台のみ存在し、計算資源から送信された ClassAd をすべて回収する。ユーザは、ジョブの実行条件を図 2 の形でジョブと共に Central Manager に投入する。Central Manager はジョブと資源の ClassAd を参照し、より理想的な計算資源にジョブの実行を割り当てる。

2.2 XtremWeb

XtremWeb は、ネットワーク上に分散した計算機の遊休時間を利用してマスタワーカ型分散ソフトウェアを大規模に実行させるためのミドルウェアである。XtremWeb は計算資源を提供する Worker とそれらに計算ジョブを割り当て管理する Server から構成される。

Worker はアーキテクチャ、OS といった自らの状態およびキーボードやマウスの使用状況などから得られる資源所有者の利用状況と計算に利用するプログラムやライブラリ、データなどの所有情報を Server 側に定期的に送信する。

Server は、ユーザからのジョブ投入を受け付け、Worker から送られてきた情報を参照して適切にジョブを割り当てる役割を担っている。

マスタワーカ型計算モデルでは、一台ないしはごく少数のマスタに対し多数のワーカがぶら下がる形になる。XtremWeb では、Linux Virtual Server[4] や DNS ラウンドロビンといったネットワークレベルでの冗長化手法によって、ワーカ数の増大に対応している。

2.3 大規模環境向け情報共有手法

この章では、多数の計算機間で情報を共有する手法の一つである Gossip Protocol について述べる。

2.3.1 Gossip Protocol

Gossip Protocol[5] は、多数のノード間において高速かつ効率的な情報共有を実現するための通信手法である。共有する情報の生成元は、次に示す過程を反復して実行され、最終的にすべてのノードが (3) で停止する収束状態をもって伝達が終了する。

(1) メッセージの発信元は接続可能なノードをランダムに選択し、メッセージを転送する。
(2) メッセージを受信したノードは、メッセージが既知か否かで次の動作に分かれる:
  (a) 既知であったときは、それを送信元に伝える。
  (b) 新しいメッセージを受信したときには、自身もそのメッセージの発信元となり、過程 (1) を開始する。
(3) 送信元は、一定回数以上既知の応答を受けるまで繰り返す。回数を超えたら停止する。

Gossip では (1) において毎回経路が動的に選択されるため、通信路に異常が発生したときでもそれを自動的に迂回することができる。また、過程を繰り返すごとに発信ノードが指数的に増加するため、ノード数に対するスケーラビリティが高い。

一方、通信経路がランダムに生成されることから全ノードに確実に情報を広報することは不可能である。

3. グリッド環境の大規模化に伴う問題点

今後グリッドを構成する計算資源の量およびそこに投入されるジョブの数が共に飛躍的に増大することが予想されるが、既存のシステムではそれに伴う負荷や障害の増加に十分に配慮されていないという問題がある。

Condor では、資源情報の収集およびジョブと資源のマッチングを行う Central Manager を一台の固定された計算機で実行している。よって、資源数とジョブ数の増加に伴う負荷が無視できないものとなる。また、Central Manager に障害が発生すると計算資源全体の利用が不可能になることから、Central Manager

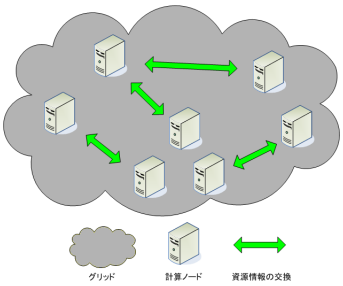

図 3 分散ジョブスケジューリングシステムの概要

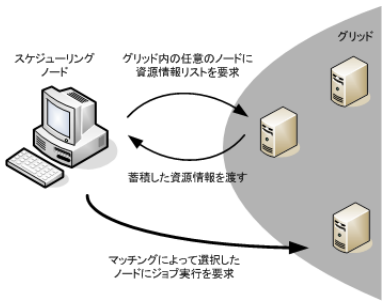

図 4 スケジューリングに必要な資源情報の取得

には高い信頼性が必要となる。

XtremWeb では、Server を複数台で構成することが出来るため、資源とジョブの増加による負荷への対応は可能である。しかし、Server のアドレスはあらかじめ固定されたものになるため、それらが存在するネットワークに障害が起きた際にはシステム全体に影響が及んでしまう。

4. 分散ジョブスケジューリングシステムの提案

グリッド環境の大規模化に伴う負荷と障害数の増加への対応は、既存のシステムでは不十分なものとなっている。そこで我々は、スケーラブルでかつ単一故障点の排除を目標とした分散ジョブスケジューリングシステムを提案する。

4.1 概　　要

スケジューリングに必要な計算資源の情報を、大規模環境に適したスケーラブルな情報拡散手法を用いて複数のノードで共有する。システムにジョブの投入を行うマシンは、グリッドを構成する任意のノードから共有する資源情報を取得し、その情報を基にジョブと資源のマッチングを形成し実行を行う。

本提案手法は次の二種類の要素から構成される。

計算ノード

計算資源の提供主体であり、また計算ノード同士で自らの資源状態を表す情報を互いに交換・共有する。

スケジューリングノード

計算ノードからグリッドを構成する各計算資源の状態情報を取得し、ユーザから投入されたジョブを適切に割り当てる。

なお、単一のマシンが両方の機能を担うこともありうる。

4.2 資源情報の共有

各計算ノードは、定期的に自らの CPU 使用率・空きメモリ・ディスク容量などに代表される情報を収集し、資源の所有者が定義された資源利用ポリシーと共に他のマシンに伝達する。伝達された情報は、Gossip などの大規模環境に適した手法を用いてグリッドを構成する多数のノードに広報され、互いの資源状態・利用ポリシーを共有する形になる。

4.3 ジョブと計算資源のマッチメイキング

スケジューリングノードはグリッドに参加しているノードを選択し、スケジューリングに必要な資源情報を取得する。ユーザはそれらスケジューリングノードに対し、ジョブと計算資源のマッチメイキングを依頼する。利用する資源が決定したら、まずその計算ノードに対し割り当て要求を出し、実行可能という返答があったら正式にジョブの実行を依頼する。すでに別のジョブが投入されている、あるいはグリッドから離脱するなどジョブを実行できないときはその旨を元のスケジューリングノードに返す。その場合はマッチングをやり直し別のノードに再度要求を出して実行されるまで繰り返す。

4.4 大規模環境に適した情報共有手法

多数のノードで情報を共有することで単一故障点および負荷の集中が回避されるものの、ブロードキャストなどが利用できない環境下において、多数のノード間で同じ情報を完全に共有することは非常にコストが高い。

また、計算ノードの動的な追加・離脱や通信経路障害への対応も必要であり、あらかじめ静的に決定された経路による情報伝達を行うことは本提案手法において現実的ではない。それらの変化に対応した通信手法を用いる必要がある。

以上を踏まえると、大規模なグリッド環境で完全な情報をすべてのノードで矛盾なく共有するのはコストが高くまた困難が多い。よって、妥当な精度でより負荷の軽い手法として本研究では Gossip Protocol を採用した。

4.5 資源情報広報への Gossip Protocol の適用

Gossip Protocol を以下のように用いることで大規模なグリッド環境下での高速で安定した資源情報の広報を行う。図 5 に概要を示す。

0 定期的に自らの資源情報を更新

計算ノードは、定期的に自らの状態を収集し、資源情報を更新する。資源情報には、それを生成した時刻と生存期限が付加されている。

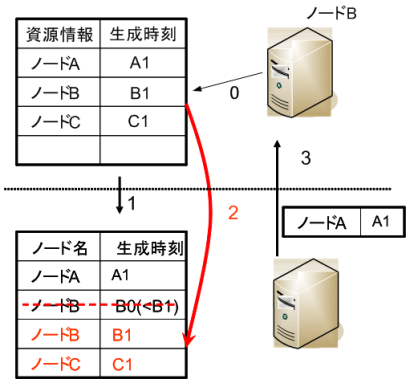

図 5 Gossip Protocol による資源情報の交換

1 ランダムに選択したノードと情報を交換
各計算資源は、グリッドを構成する計算ノードをランダムに選択して接続し、互いに自らの持つ計算ノード群の各資源情報を交換する。

2 受信ノードは自らの保持する情報を更新
受信ノードは、送られてきた情報のうち保持していないものを自らのテーブルに追加する。同じ情報源から生成された情報を持っている場合は、タイムスタンプを参照してより鮮度の高い情報を採用する。

3 既知であった情報を送信元に通知相手から送られてきた情報のうち、既知であったものを送信元に通知する。

計算資源が新たにグリッドに参加するときには、スケジューリングノードと同様にグリッドに参加している近隣のノードに対し資源情報を要求することで Gossip に必要な他ノードの位置を取得する。また、計算資源がグリッドを離脱する際には、離脱メッセージを Gossip で広報することによって、そのメッセージを受信した各ノードはその資源が利用不可能であるとしてその資源情報を破棄する。また、各自ノードは保持する情報の生存期限をチェックし、期限を過ぎた情報は破棄を行うことによって、計算機やネットワークの障害によってアクセスできなくなった資源は自動的に利用対象から除外される。

5. 性 能 評 価

5.1 Gossip Protocol の性能

5.1.1 Gossip Protocol の基礎評価

一周期に選択する伝達ノード数を 1、既知応答を受ける回数を 1 としたとき、収束時における過程の反復回数による伝達率の変化を図 6 に示す。

5.1.2 通信の非対称性が存在する環境

現在 NAT やファイアウォールなどによって、外部から内部への接続が制限される環境に属する計算機同士が多数存在する。Gossip Protocol では各ノードが

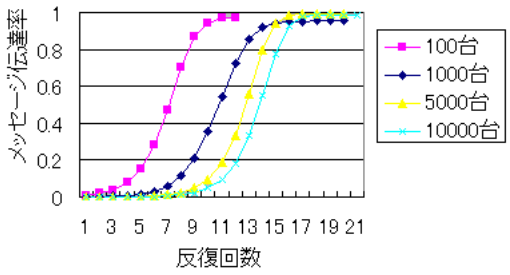

図 6 反復回数によるメッセージ伝達率の変化

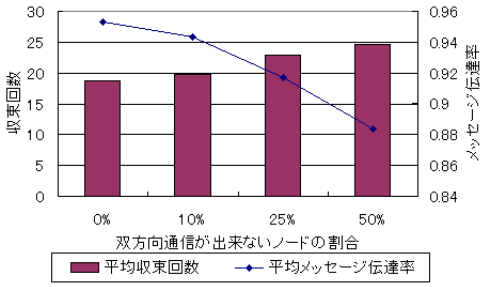

図 7 メッセージ伝達率

直に接続することで情報交換を行うため、そのようなサイト間をまたいで直接資源情報を交換することはできない。

しかし、TCP のように内部から接続を張ることで双方向通信が可能な通信手法であれば情報の交換を行うことができるため、双方向通信を行えるホストを経由して資源情報の流通を行うことが可能である。ノード 1000 台のうち、外部から通信を張ることができないノードの割合を変えたときの資源情報の流通速度を測定した。結果を図 7 に示す。

双方向通信が不可能なノード数が増えるにつれ収束に要する回数は増加し最終的な情報の伝達率も低下するが、ノードの割合が大きくてもその低下は比較的小さく、情報の完全性が極端に悪化することがないことを確認した。

5.2 分散ジョブスケジューリングシステムの性能評価

分散ジョブスケジューリングシステムと、Condor をモデルとした既存の集中管理型システムをそれぞれシミュレータ上に実装し、同じ性能を持つ計算ノード、ネットワークに対し同じジョブを投入した際のスループットを測定した。

両システムで共通とした資源とジョブのマッチングに必要な処理時間は、Condor の MatchMaking システムをモデルにしたものを Java で実装し、実際に処理にかかった時間をコストとして用いた。

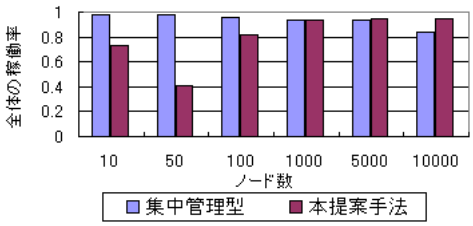

図 8 本提案手法と集中管理型との性能比較

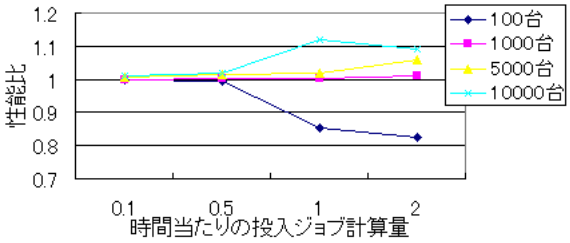

図 9 計算ノード数の変化による性能比

すべての実験において、Gossip による資源情報の交換を行う頻度は 1 分に 1 回とし、グリッドを構成する計算ノードの条件は以下のように設定した。

- 計算ノードの性能
  ノードのプラットフォームおよび計算性能はすべて同じものとした。
- ネットワーク
  ノード間を結ぶネットワークのトポロジおよび遅延は GridG[6] によって生成されたものを利用し、ノード間遅延が最大 200ms 程度となる環境とする。

今回投入されるジョブは、実行条件に特別なものはつけず任意のノードで実行可能であるとした。

グリッド全体の稼働率

スケジューリングシステムがどれだけ計算資源を効率よく利用できるかを評価する基準として、ここではグリッド全体の稼働率を用いる。グリッドを構成する全計算ノードを n 台とし、その内ジョブ実行中であるノードを m 台とするとき、グリッド全体の稼働率は $\frac{m}{n}$ と定義する。

同じジョブの集合を投入したとき、より多くの計算ノードに実行させることができ稼働率を高めることが出来るものほど優れたスケジューリングシステムであると言える。

5.2.1 集中管理型システムとの性能比較

本提案手法と既存の集中管理型システムの比較を行うため、スケジューリングノード数を集中管理型と同じ 1 台に固定し、グリッドを構成する計算ノード数の規模による性能変化の評価を行った。

ジョブ一つの実行に必要な時間は、基準マシンにおいて 30 分から 1 時間の間になるように設定した。また、投入されるジョブの時間当たりの計算量は各計算ノードの総処理能力の和に等しくなるように設定し、理想的なスケジューリングが行われた際のシステム全体の稼働率は 1 となる。

このジョブをそれぞれのジョブスケジューリングシステム下で実行した際のスループットを図 8 に示す。

ノード台数が 10 台から 100 台までの中小規模な環境においては、本提案手法が集中型システムに比べ大きく性能が劣っている。これは、集中管理型システムが各計算ノードから 1 ホップで直接資源情報を回収しているのに対し、Gossip Protocol の場合他ノードを複数回経由して伝達される遅れに起因する情報の不完全性がスケジューリングに大きく影響しているためである。

表 1 スケジューリングノード数によるシステム全体の稼働率の変化

スケジューリングノード数	システム全体の稼働率
1	0.939
2	0.941
4	0.967
8	0.970

ノード数が増えるにつれ、集中管理型システムでは情報収集の集中によるオーバーヘッドが無視できなくなり、相対的に Gossip による通信時間よりも大きくなるため、現在大規模とされる 1000 台以上の環境では、既存のシステムとほぼ同程度ないしは最大 10%程度上回る稼働率を示している。

5.2.2 投入ジョブの計算量による性能の変化

計算ノードが持つ処理能力の総和に対する計算量に対し、時間当たりの計算量が 10 %,50 %,100 %,200 %となるようなジョブの集合を、一台のスケジューリングノードから投入するシミュレーションを行った。集中管理型システムのシステム全体の稼働率を基準とした際の本提案手法の性能比を図 9 に示す。

ノード数が少ないときは、ジョブ数が増加しシステムの利用率が高くなるにつれて性能が既存システムに比べ低下する。これは、利用率が高くなると利用可能な計算ノードが少なくなり、5.2.1 で述べた Gossip の情報の不完全性がよりスケジューリングに悪影響を与えるためである。ノード数が増加すると同様に集中型のオーバーヘッドが Gossip のそれを上回るため本提案手法がより高い性能を示している

5.2.3 複数マシンによるスケジューリングの性能

本提案手法の上で計算資源とジョブのマッチングを行うノードの台数による性能の変化を測定した。計算資源 1000 台のシステムに対し、処理能力の総和に対する計算量が 100 %となるような投入間隔に従うジョブ列を投入し、それぞれ 1 台、2 台、4 台、8 台でマッチングした際のシステム全体の稼働率を表 1 に示す。台数が増えることでより効率的な割り当てが行われている結果が得られた。

6. 考　　察

本提案手法において性能が低下すると思われる主な要因は次の二つである。

- Gossip の伝達時間のオーバヘッド
- 情報の不完全性

これらについての考察を述べる。

6.1 Gossip の伝達時間のオーバヘッド

本提案手法では、資源情報が複数のノードを経由して伝達されるため、既存の集中管理型システムに比べ資源情報の収集時間が長くなる欠点がある。

5.2.2 の結果では、計算ノード数によって投入ジョブ量の増加による性能変化の形が大きく異なる。これは、グリッド全体の利用率が高く、実行待ちジョブが多く存在する状態では、計算資源情報の収集速度がスケジューリングの効率に大きい影響を与えるためである。Gossip Protocol においては、ノード数が少ないときでもある程度の反復回数が必要なため、そのようなときには単純な集中管理型のほうがより高速な情報収集が行える。

しかし、集中管理型では情報収集のコストが計算資源数に対し線形に増加するため、資源数が増えると Gossip の伝達速度より通信の集中の影響が相対的に大きくなる。流通する情報量は計算ノード数と実行ジョブ数に比例するため、提案手法のほうがより大規模環境に適したスケーラブルなスケジューリングが可能であるといえる。

6.2 情報の不完全性

Gossip Protocol では全ノードに完全に情報を広報することはできないが、5.1.1 および 5.1.2 によれば、ノード台数の増加に対し広報に必要な時間・伝達度ともに高い結果が得られている。

ジョブ実行開始・終了といった計算ノードの状態変化が頻繁に発生する状態では、Gossip によって流通する情報がすでに現状を反映していない状況が起こりうる。しかし、一つのジョブ実行に必要な時間が Gossip での広報に要する時間に対し十分長いものであれば、上のような情報の不完全性は十分小さいといえる。

上の二つの理由により、計算ノード数および投入されるジョブが大規模なグリッド環境においては、Gossip による情報共有が十分実用的であると言える。

5.2.3 では、マッチングを行うノードが多いほど稼働率が向上し、効率の良いスケジューリングが行われている結果が得られた。これは、マッチングの負荷が分散されたことと、情報の取得を複数地点から行ったため、Gossip の欠点である情報の不完全性が小さくなったためと考えられる。スケジューリングノードが複数存在することでそれらの故障に対してよりロバストでかつ効率的なスケジューリングが可能になることが示された。

7. おわりに

7.1 ま と め

不完全な情報共有手法で資源情報の収集を行う分散ジョブスケジューリングシステムを提案し、シミュレータ上で集中型との性能比較を行った。現在小中規模とされる台数下では既存システムに対する通信コストが大きいことにより性能が大きく低下するが、それを超える大規模環境においては同等ないしは若干上回る利用効率を確認し、本提案手法の計算ノード数増加に対するスケーラビリティが示された。また、スケジューリングマシンの複数化によってグリッド全体の利用効率が向上することが確認された。

7.2 今後の課題

今後の課題としては以下が挙げられる。

- Gossip 以外の情報共有手法の検討
- 広報される情報の検証機構
- 実際のジョブスケジューリングシステムへの適用および実環境での検証

参 考 文 献

1) Litzkow, M.J., Livny, M. and Mutka, M.W.: Condor - A Hunter of Idle Workstations, *Proceedings of the 8th International Conference on Distributed Computing Systems (ICDCS)*, Washington, DC, IEEE Computer Society, pp. 104–111 (1988).
2) GillesFedak, Ce'cileGermain, V.N. and Cappello, F.: XtremWeb : A Generic Global Computing System, *CCGRID2001, workshop on Global Computing on Personal Devices* (2001).
3) Raman, R., Livny, M. and Solomon, M.: Matchmaking: Distributed Resource Management for High Throughput Computing, *Proceedings of the Seventh IEEE International Symposium on High Performance Distributed Computing (HPDC7)*, Chicago, IL (1998).
4) : The Linux Virtual Server Project - Linux Server Cluster for Load Balancing, http://www.linuxvirtualserver.org/.
5) Jenkins, K., Hopkinson, K. and Birman, K.: A Gossip Protocol for Subgroup Multicast, *International Workshop on Applied Reliable Group Communication (WARGC 2001)* (2001).
6) Lu, D. and Dinda, P.: Synthesizing Realistic Computational Grids, *"Proceedings of ACM/IEEE Supercomputing 2003 (SC 2003)"* (2003).