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[†, †††]

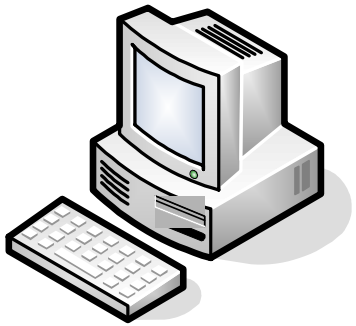
†: 東京工業大学
††: 産業技術総合研究所
†††: 国立情報学研究所

背景

- ジョブスケジューリングシステム
  - 分散した計算資源を集約し、統合された計算機環境をユーザに提供
  - 計算資源の状態を定期的に収集
    - アーキテクチャ・OS
    - CPU使用率・メモリ容量・ネットワーク帯域・ユーザ利用状況
    - 計算で用いるユーザプログラム・データの所有状況
    - 資源所有者が設定した利用ポリシー
  - ユーザから投入されたジョブを計算資源に割り当て
    - 収集された情報を参照してより効率的に実行

ジョブスケジューリングシステム

スケジューリングマシンはグリッドを構成する計算資源の状態を収集する



スケジューリングマシン

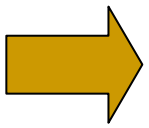
計算ノード	時刻	アーキテクチャ	OS	CPU使用率
X	14:53:02	EM64T	Linux	3%
Y	14:53:34	x86	WinXP	94%
Z	14:53:29	SPARC	SunOS	12%

収集した情報を元に、計算資源にジョブを割り当てる

ユーザからのジョブ投入を受けつけ、キューに蓄える

グリッド環境の大規模化

- 今後予想されるグリッド環境の大規模化
  - 計算資源数の増加
  - 投入されるジョブの増加
- それに伴う問題
  - ジョブスケジューリングに要するコストの増大
    - 計算資源の情報収集
    - 資源とジョブのマッチングに要する計算



大規模化に伴う問題に対応したジョブスケジューリングシステムが求められる

目的と成果

- 目的
  - 大規模環境向け情報共有手法を利用した分散スケジューリングシステムの提案と検証
- 成果
  - Gossip Protocolを情報共有に使用したスケジューリングシステムを提案
  - シミュレーションで提案手法を評価
    - 情報の不完全性によるペナルティが小さいことを確認
    - 計算資源数の増加に対するスケーラビリティを確認
    - 複数台による分散ジョブスケジューリングを行うことによるグリッド全体の利用効率向上

関連研究 Condor [Livny et al. ’88]

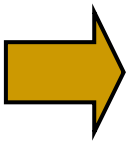
- ウィスコンシン大学で開発されたジョブスケジューリングシステム
  - スケジューリングマシン
    - CentralManager: 計算資源の情報を収集
    - Scheduler: ユーザからのジョブ投入を受け付け
      - ジョブ割り当て機構: Matchmaking
        - 収集した情報を元に、投入されたジョブの実行条件・優先度を最も満たす資源を選択
- 問題点
  - スケジューリングマシンが単一故障点
  - 情報収集およびマッチングのコストの集中

関連研究:XtremWeb[Cappello et al. ’00]

- INRIAで開発されたジョブスケジューリングシステム
- 多数の計算ノードの利用を想定
- 1台ないしは複数の計算機によるジョブスケジューリング
  - スケジューリングマシンを増やすことで資源数の増加、投入ジョブ量の増加に対応することが可能
  - 資源情報収集、ユーザからのジョブサブミッション、ジョブと資源の割り当てなど個々のスケジューリング処理機能を複数マシンに分散
- 問題点
  - 耐故障性に問題
    - スケジューリングマシンのネットワークアドレスが固定されている

既存システムの問題点

- スケジューリングマシンが単一故障点
  - 1台ないしは少数の固定された計算機で行われる
  - ダウンするとシステム全体が利用不可能
- 計算資源・ジョブの増加に伴う負荷の集中
  - スケジューリングのコストが無視できなくなる
    - 情報収集で消費されるネットワーク帯域
    - マッチングを形成するための負荷



ジョブスケジューリングを複数の計算機に

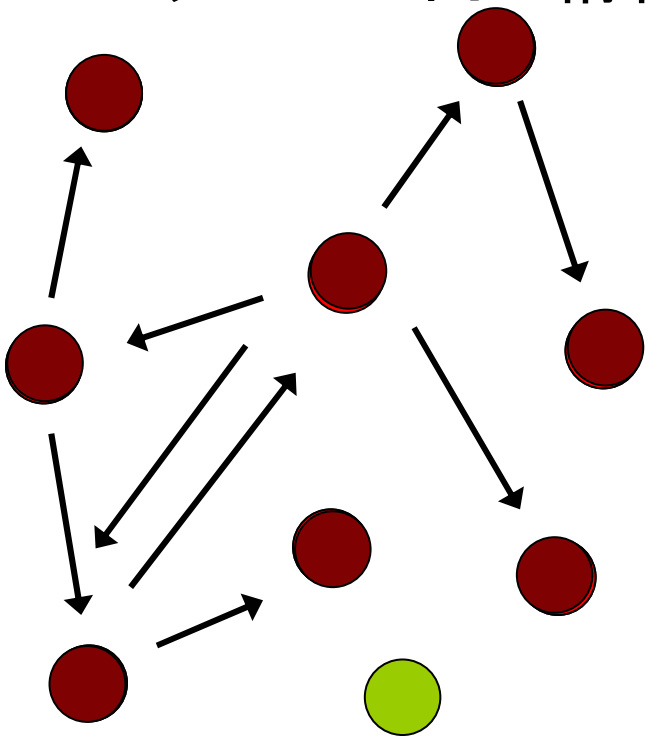
分散させることが必要

提案

- Gossip Protocolによって資源情報を共有するジョブスケジューリングシステム
  - 台数の増加に伴うコストが低い情報収集手法を採用
  - 情報の不完全性とスケーラビリティはトレードオフであり、多数ノード間で完全に情報を共有するためのコストは高く、非現実的
  - 複数の計算機でジョブスケジューリング
  - 負荷分散と単一故障点の排除
    - スケジューリングに必要な計算資源情報を拡散
    - スケジューリングの処理を複数のマシンで分散

Gossip Protocol [Birman et al ’01](1/2)

- グループ内で情報を広報・共有する通信手法



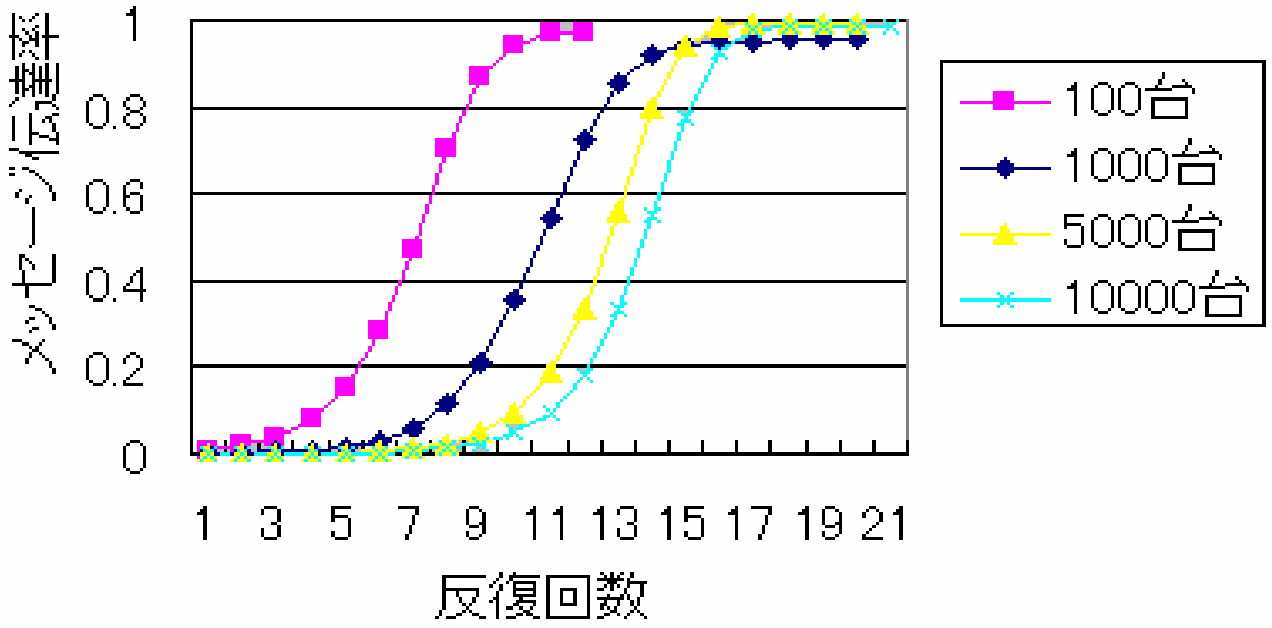
未知ノード
既知ノード
停止ノード

0. 情報を広報したいノードは、自らが発信ノードとなる
1. 発信ノードはノードをランダムに選択し送信
2. 情報を受信したノードは、その情報が
   A) 未知なら自らも発信ノードとなり1から手順を開始
   B) 既知なら発信元ノードに応答を返す
3. 情報の発信ノードは、
   A) 再び1から送信を開始する
   B) 一定回数以上既知応答を受信したら停止

1～3の過程を反復することで一台の情報生成元ノードから他ノードに情報が伝わり、最終的にすべてのノードが3.Bの状態に収束することで情報共有が行われる

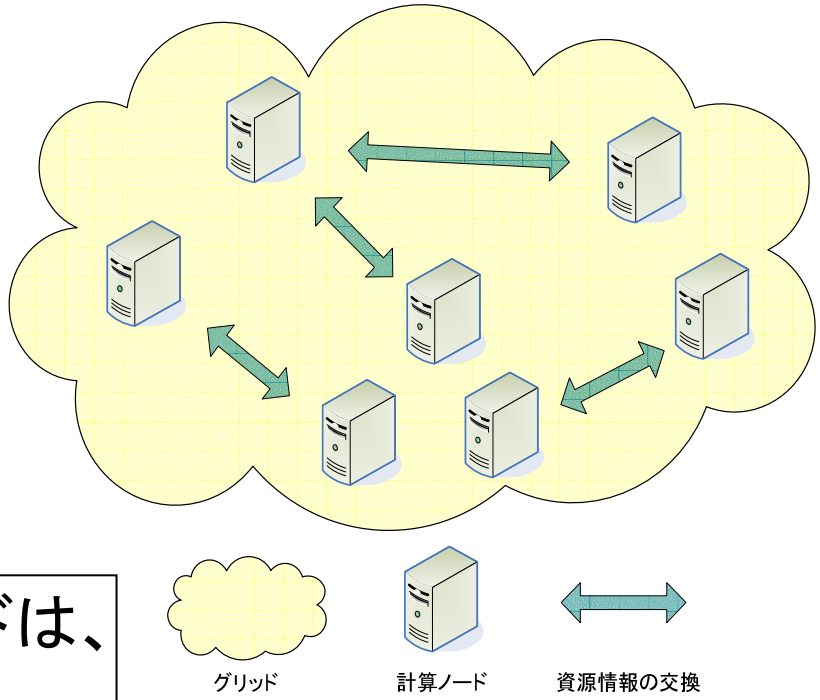
Gossip Protocol [Birman et al ’01](2/2)

- Gossip Protocolの特徴
  - 通信路・ノードに対する耐故障性
    - 通信経路は各過程ごとに毎回動的に選択・生成
  - ノード台数の増加に対しスケーラブル
    - 過程を繰り返すごとに指数的に伝達ノードが増加する
  - 全ノードに情報を確実に送信することは不可能



ジョブスケジューリングシステムの設計(1/2)

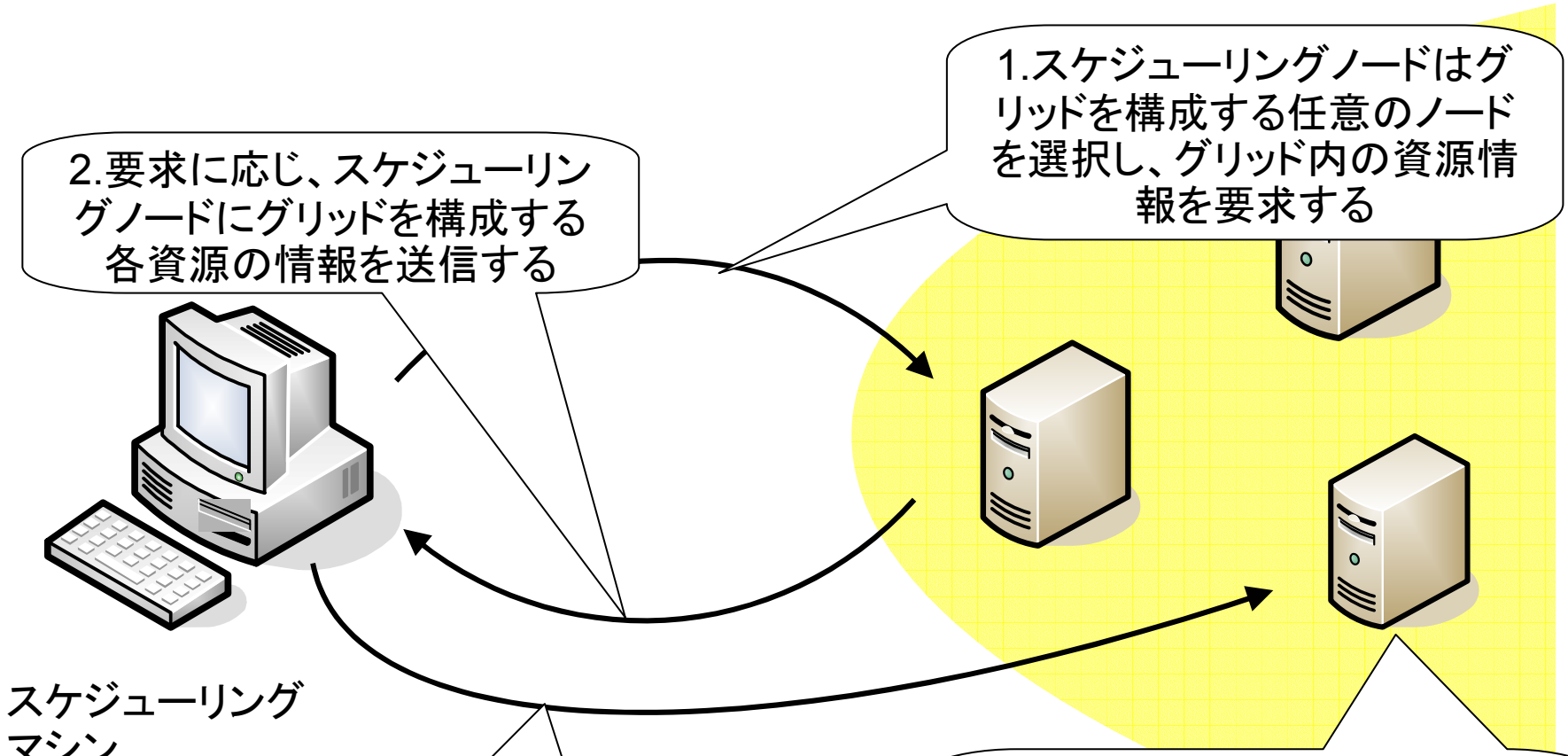
- グリッドを構成するノードが互いに資源情報をGossip Protocolによって交換・共有
  - スケジューリングに必要な資源情報を多数のノードで保持
- グリッドに計算資源を追加、削除するにはその情報をグリッド内の任意のノードに広報



グリッドを構成する全てのノードは、
- 計算ノード
- スケジューリングノード

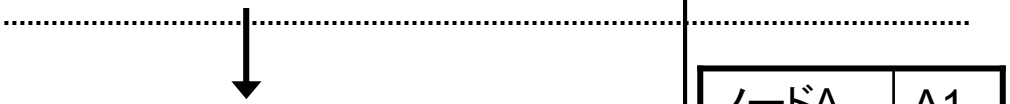
両方の機能を兼ねることが可能

ジョブスケジューリングシステムの設計(2/2)

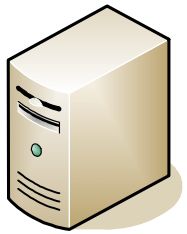


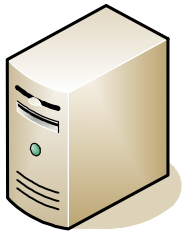
Gossipを用いた資源情報の共有

ノード名	生成時刻
ノードA	A1
ノードB	B1
ノードC	C1



ノード名	生成時刻
ノードA	A1
~~ノードB~~	~~B0(<B1)~~
ノードB	**B1**
ノードC	**C1**





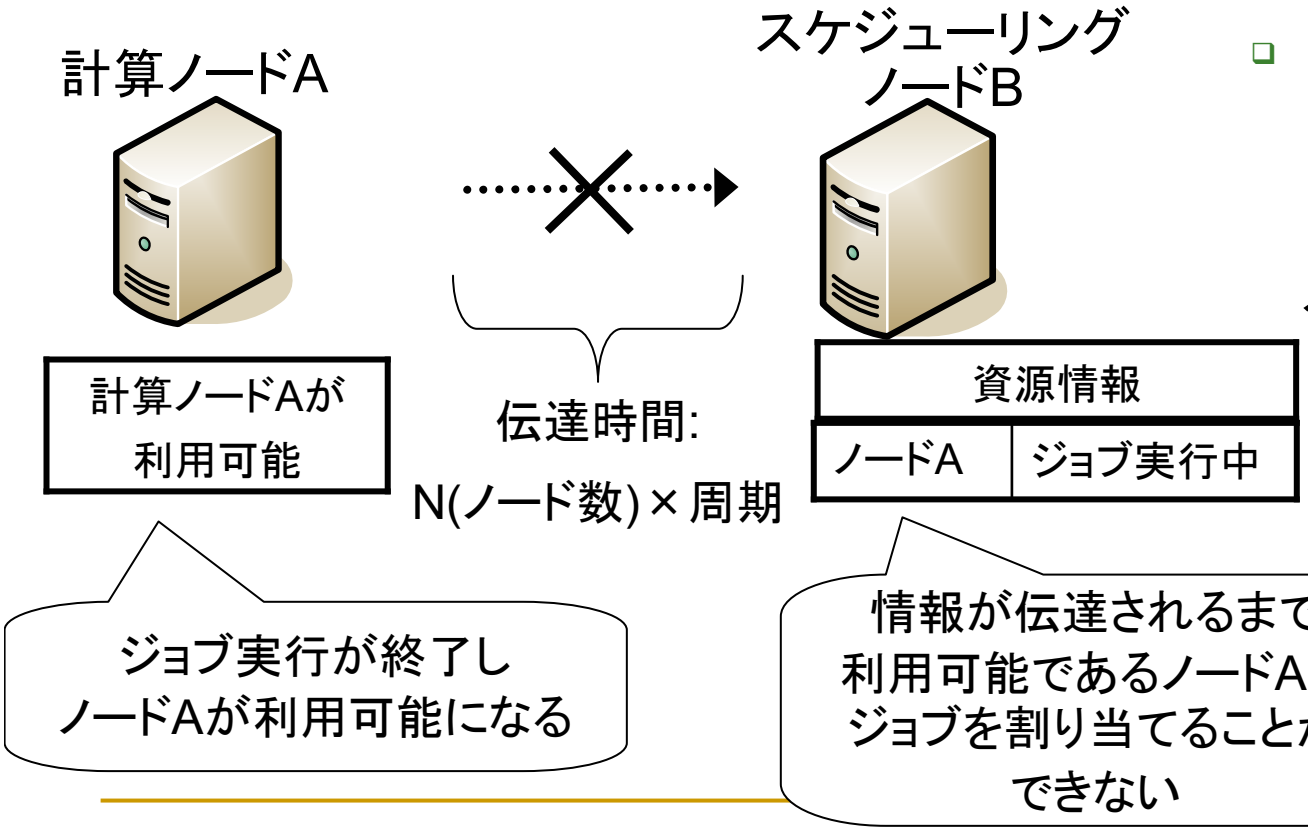
ノードA	A1

各ノードは、他のノードも含めた資源情報を保存するテーブルを保持し、自分の情報は適宜更新する。

1. Gossipにより、定期的にノードをランダムに選択し、相手に自分のテーブルを送信する
2. 受信側は、送られてきた情報のうち自分の保持していないものおよびより新しい情報を自分のテーブルに追加する
3. 受信側は、すでに既知であった情報を相手に通知する
4. 一定回数以上既知応答を受けた情報は以後外部に広報しない

本提案手法における懸念

- 性能低下となる原因
  - 情報伝達に要する時間
  - 情報の不完全性

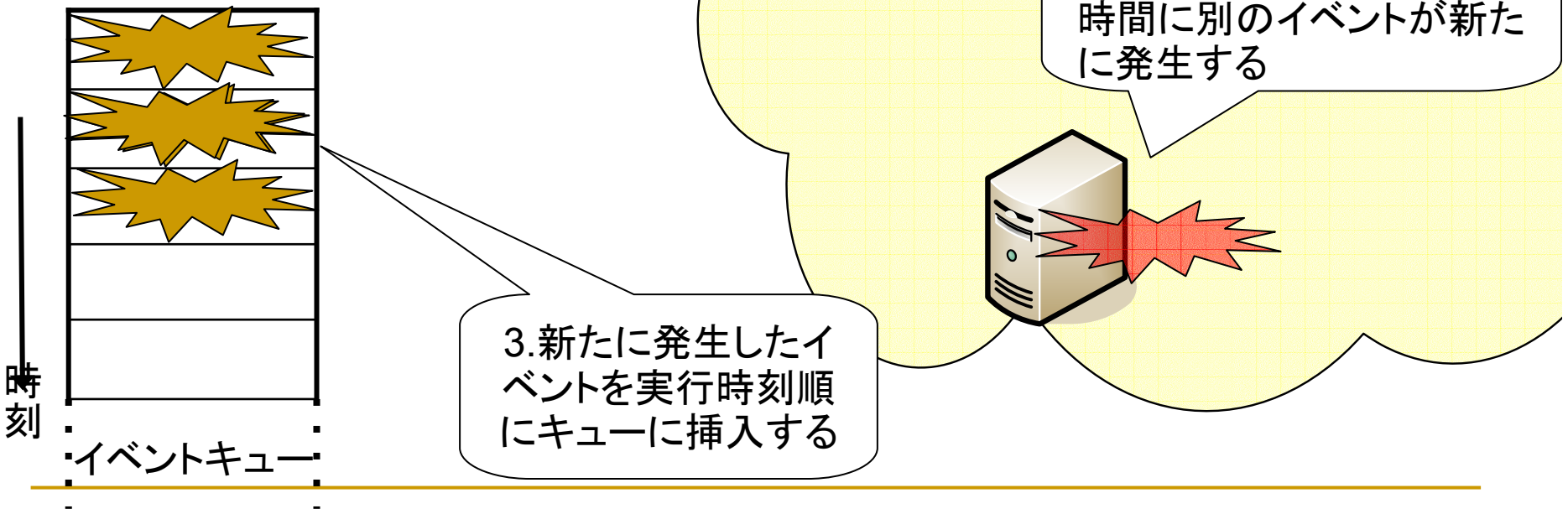


- 保持する情報が現実を反映していない
  - 複数ノードを経由することによる情報伝達の遅れ
  - 全ノードへの広報が保障できない

これらGossip の欠点がスケジューリングに与える影響をシミュレーションで評価する

シミュレーション環境(1/2)

- 離散事象シミュレータ
  - シミュレーション対象イベント
    - ノード間での送受信
    - ユーザからのジョブ投入
    - ジョブと資源のマッチング
    - ジョブ実行・終了
    - etc…



シミュレーション環境(2/2)

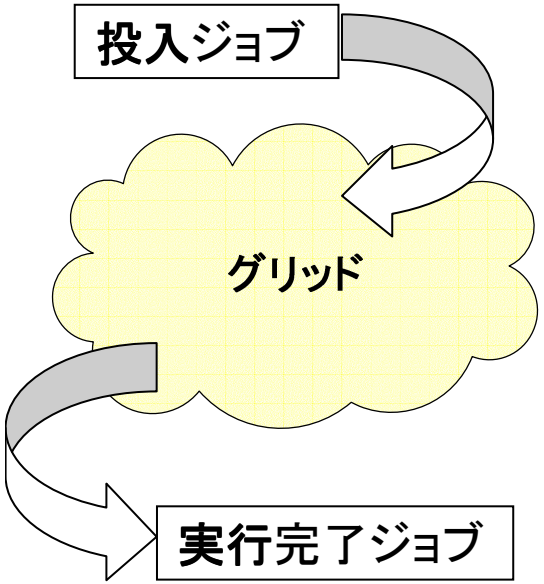
- シミュレーション対象
  - スケジューリングノード
    - ジョブと計算資源のマッチング
    - 資源情報の収集
  - 計算ノード
    - ジョブ実行を開始したら、実行終了時間まで計算機のCPU使用率が変化
  - ノード間通信
    - ノード間の通信遅延・帯域
    - メッセージサイズ
  - 投入ジョブ列
    - 投入間隔
    - ジョブの計算量

ジョブスケジューリングシステムの評価(1/2)

- 評価指標
  - スケジューリング効率

$$\text{理想的なスループット} = \frac{\Sigma\,(\text{時間T内に投入されたジョブの計算量})}{(\text{時間T})}$$

$$(\leqq \Sigma\,(\text{計算ノードの処理性能}))$$



•投入したジョブがオーバーヘッド無しに実行される

$$\text{実際のスループット} = \frac{\Sigma\,(\text{時間T'内に完了したジョブの計算量})}{(\text{時間T'})}$$

“実際のスループット”が理想に近いほど性能の高いジョブスケジューリングシステムである

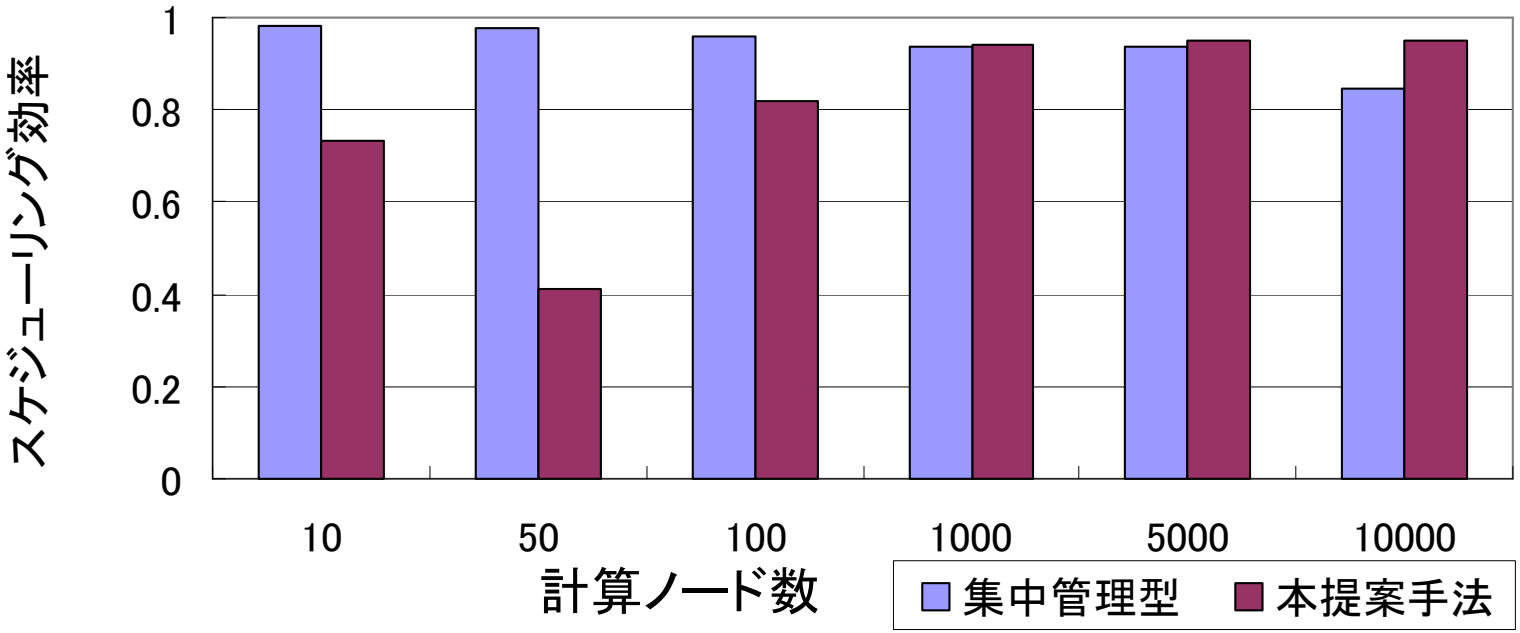
$$\text{スケジューリング効率} = \frac{\text{実際のスループット}}{\text{理想的なスループット}}$$

ジョブスケジューリングシステムの評価(2/2)

- 想定環境
  - 計算ノード
    - プラットフォームおよび計算性能は全て同一
    - 投入されたもの以外のジョブ実行は発生しない
  - ネットワーク:グリッド環境生成ソフトウェアGridG [Lu et al. ’03]によるトポロジと帯域
    - ノード間遅延は最大200ms程度
  - スケジューリングに要するコスト
    - CondorのMatchmakingに相当する機能を実装し、そこで実際に要する時間を採用
  - 投入ジョブ
    - 一つのジョブの実行に要する時間は計算ノードで30～1時間程度
    - 投入間隔: ポアソン分布 λ =(平均ジョブ投入間隔)
  - Gossip パラメータ
    - 各ノードは1分に1回の間隔で情報交換を行う
    - 一過程で選択するノードは1台

ノード数がスケジューリング効率に与える影響

グリッド全体の処理能力と同等の計算量を持つジョブ列を
スケジューリングシステムに投入したときのスケジューリング効率を測定



- 小規模なグリッドでは集中管理型のほうが効率的
- 1000台を越える大規模環境では同等ないし若干高い性能を示している
  - 情報の不完全性によるペナルティは十分小さい

情報伝達時間の増加が与える影響

- グリッドの利用率について

$$(\text{グリッドの利用率}) = \frac{\text{スループット}}{\Sigma\,(\text{計算ノードの性能})}$$

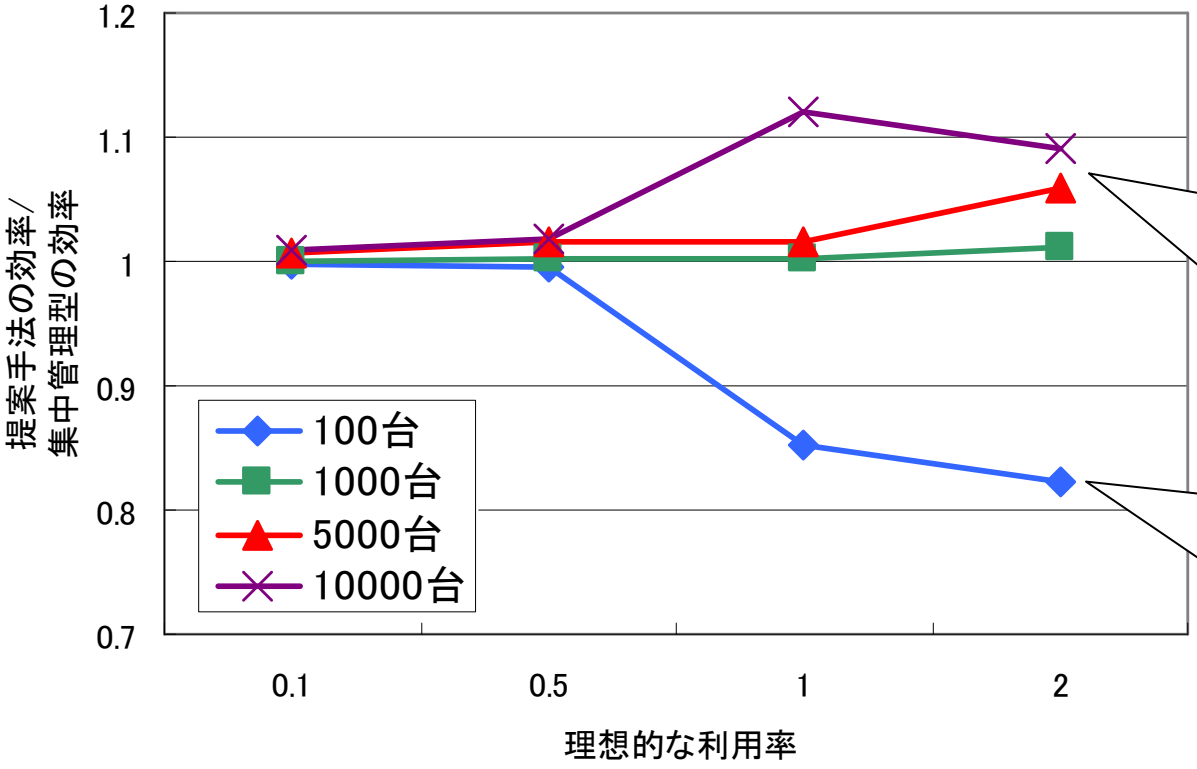
  - 利用率が小さい場合
    - 利用可能なノードが多数残っているため、情報伝達が遅くても実質的な影響は小さい
  - 利用率が大きい場合(利用可能な計算ノードが少ない)
    - ジョブ実行開始、終了時に発生する計算ノード状態変化の広報速度がジョブスケジューリングに大きく影響する
- 情報伝達速度による影響の評価
  - グリッドの利用率を変化させたときの性能を測定
    - グリッドに投入するジョブの投入間隔を変える
    - ノード台数N, 平均ジョブ実行時間がEのとき、理想的なグリッド利用率がXとなるようなジョブ投入間隔 λ はE/(N･X)となる

情報伝達時間の増加が与える影響

- 理想的なスケジューリングが行われた際の利用効率が

0.1/ 0.5 / 1 / 2(処理性能の2倍のジョブを投入) となるように間隔のみを変更してジョブを投入する

- 集中管理型での効率を基準としたときの本提案手法の性能比を測定



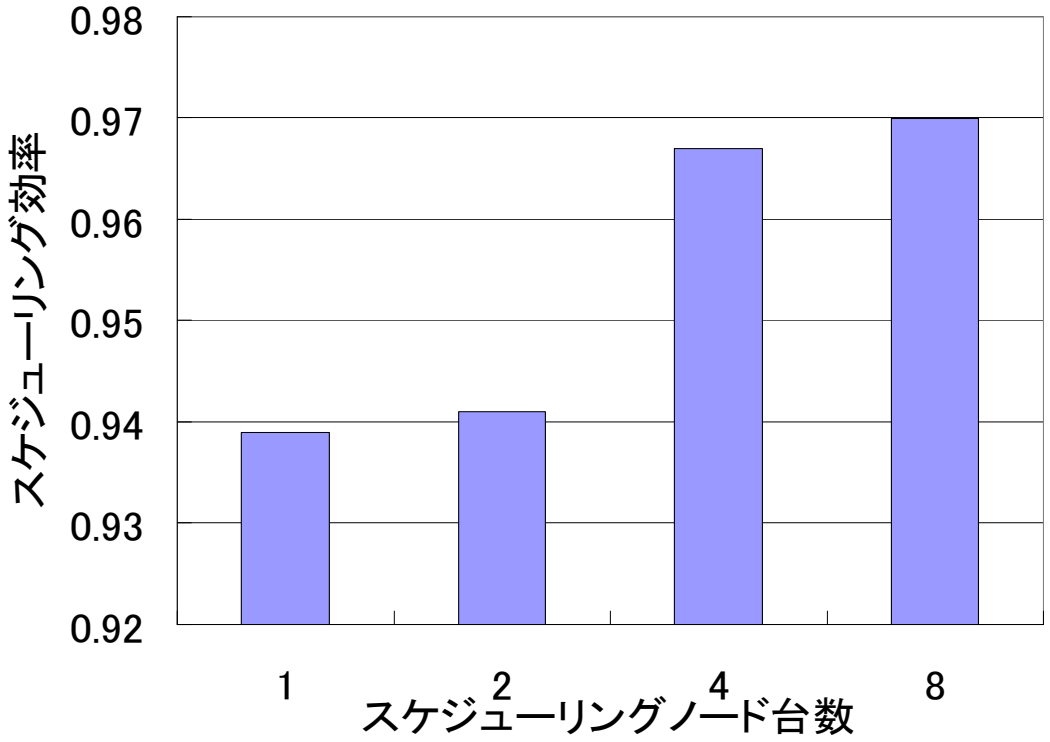
ノードが多い環境では、利用可能な計算ノードの絶対数が多いため、情報の不完全性および情報伝達速度の低下によるデメリットが隠蔽される

小規模なグリッドでは、ジョブ実行中のノードが増えるにつれ情報伝達速度の低下によるペナルティが大きい

スケジューリングノード数によるスループットの変化

- スケジューリングノード数を増やし、同じジョブを投入した際のグリッド全体のスループットを測定

計算ノード	1000台
理想的な利用率	1.0



- スケジューリングノード台数を増やすことでグリッド全体のスループットが向上
  - 異なるノードから計算資源情報を取得することにより、より完全に近い情報を取得できるため

まとめと今後の課題

- まとめ
  - グリッド環境の大規模化に対応した分散ジョブスケジューリングシステムを提案
    - Gossip Protocolを用いて資源情報をグリッド内に拡散
  - シミュレータ上で既存の集中管理型システムとの比較
    - 計算ノード数の増加に対するスケーラビリティを確認
    - 大規模環境下ではGossip Protocolに伴う情報の不完全性による性能低下が十分小さいことを確認
    - スケジューリングノードの複数化による利用効率の向上
- 今後の課題
  - より多様な環境における性能評価
  - 実システムでの本手法の適用・検証