クラスタ監視ソフトVGXPの開発・運用を通しての評価と課題

東京大学情報基盤センター
鴨志田 良和　kamo@cc.u-tokyo.ac.jp

Agenda

- 背景
- クラスタ監視ソフトVGXP
- 実環境での評価と課題
- まとめ・今後の方向性

背景

- クラスタシステムの扱いづらさ
  - 計算ノード数が多い
  - しかも状況が時々刻々変化する場合がある
  - →何が起きているか分かりづらい
- システムの利用者・管理者の双方にとって問題
  - 利用者: 並列プログラムのチューニングやデバッグを行う
  - 管理者: システムの異常検知、効率のよいジョブスケジューリングを行う

クラスタシステムの現在の状態が見えると
日常的なクラスタ利用の利便性が向上する

クラスタのリアルタイム監視

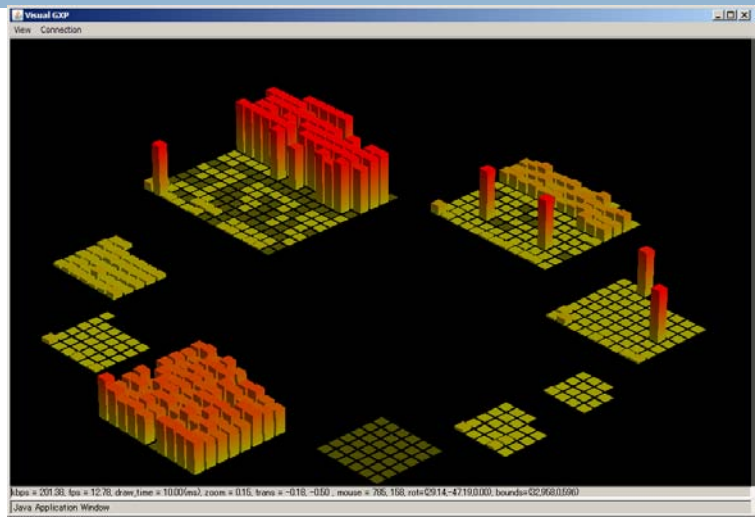
- ノードごとにCPU、メモリの利用率など、さまざまなデータを細かい(100ms～数秒程度の)間隔で取得し、各ノードのデータを集約して可視化する
- 利点
  - システム利用者にとっての利点
    - 投入したジョブの実行中に異常を発見できる
    - 負荷分散の時間変化を見ることができる
  - システム管理者にとっての利点
    - 全体的なリソースの利用状況を一目で把握できる
      - 大量のログファイルとの対峙からの解放
- 欠点
  - システムに与える負荷が大きい

VGXP(Visual Grid Explorer)

- 分散環境向けの軽量な監視システム
  - システムになるべく大きな負荷をかけずに、
  - 高いリアルタイム性(2秒程度の遅れ)で、
  - 複数のクラスタを監視
- ソフトウェア
  - クラスタ用シェルGXPを使用して構築
    http://sourceforge.net/projects/gxp/
  - クライアントはJava(Java Web Start)とOpenGLを使用
  - 公開URL

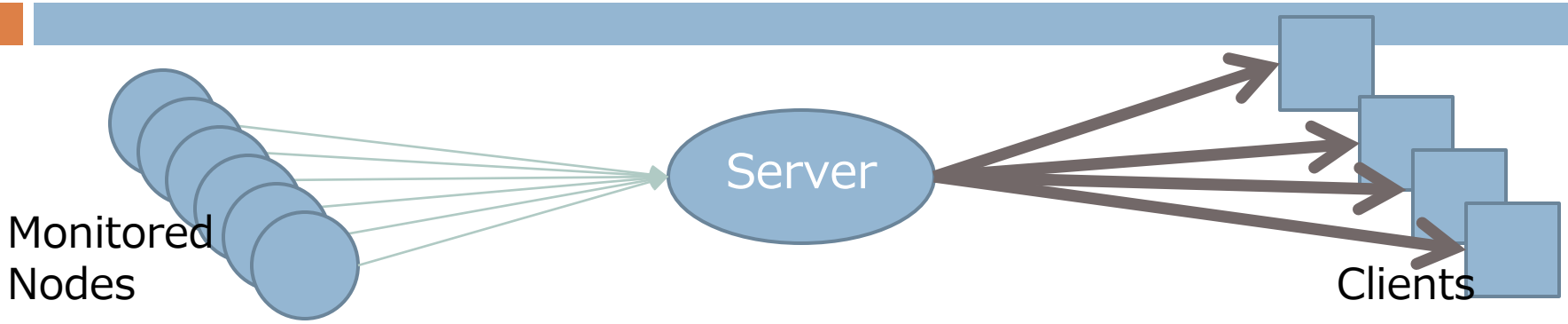
    http://www.logos.ic.i.u-tokyo.ac.jp/~kamo/vgxp/

VGXPの機能



- VGXPが表示する情報
  - CPU time
  - Load average
  - Memory usage(used, swap…)
  - Network(sent, received)
  - I/O (Block r/w, Swap in/out)
  - **プロセスごとの情報(cmdline, user, CPU time)**
- 情報更新の頻度: 2sec.

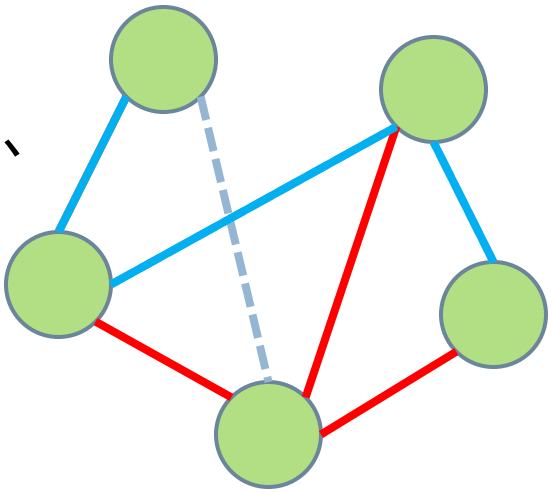
VGXPの動作の仕組み



- 監視対象ノード間で木構造のTCP接続を構築してデータを集約する
- 集約したデータをサーバに転送し、サーバが各クライアントにデータを配信する
- ノードの追加や脱退の際は、木を再構築する

ネットワークの再構成

- 監視プロセスの状態を監視するメタ監視用ネットワークを構成
- 故障が発生した場合
  - メタ監視プロセスが故障を検知
  - 他のノードの監視は続行
  - 監視用ネットワークを再構成したのち、監視プロセスを再起動



他のシステムとの比較

- 監視にかかる負荷に着目し、他のシステム監視ソフトと比較を行う
  - Ganglia
    - **http://ganglia.sourceforge.net/**
  - Munin
    - **http://munin.projects.linpro.no/**
  - Real Time Monitor (GridPP内)
    - **http://www.gridpp.ac.uk/**

どちらも、そのままでは高頻度の情報更新には向いていない

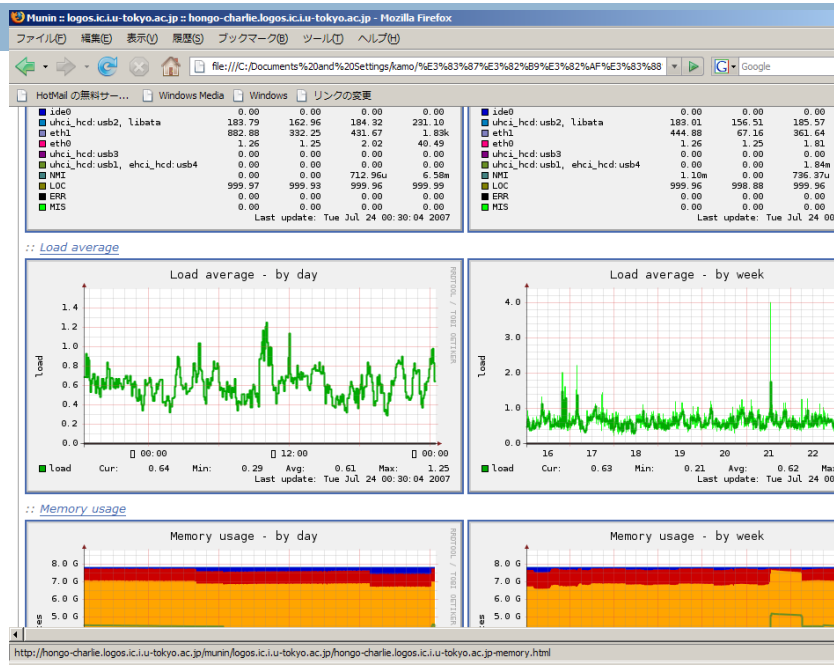
クライアントの表示負荷に改善の余地あり

監視にかかる負荷 - Ganglia

- Ganglia
  - 定期的にノード情報をクラスタ内でマルチキャスト
  - 1画面あたりのデータ転送量: 0.9MBytes (120 nodes)
  - グラフ生成の負荷はサーバ上に集中

監視にかかる負荷 - Munin

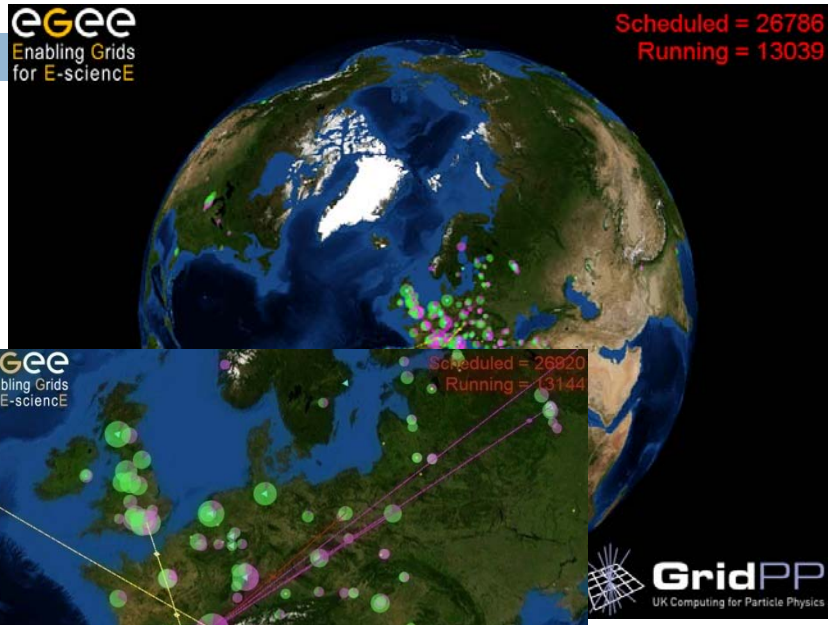
- Munin
  - 普段は各ノードでは何も負荷は発生しない
  - 5分ごとに、サーバが全ノードのデータを収集し、グラフ画像を生成
    - 1.5～3分(120 nodes)
  - グラフ画像のサイズ 1.99MBytes / node

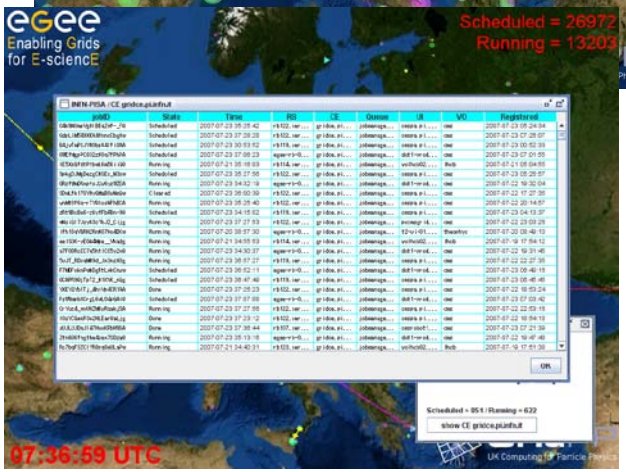


→**300ノード程度で限界**

クライアント側の可視化

- Real Time Monitor (2006)
  - サーバは生データだけを送ってクライアントが可視化処理を行う
  - 3分遅れで情報を表示
  - 拠点ごとにジョブ情報を表示





比較のまとめ

	Ganglia	Munin	RTM	VGXP
情報の粒度	Per node/Per site	Per node	Per site	Per node
収集されるデータの種類	Various info.(CPU, mem, net, disk…)	various	Jobs only	various
プロセスごとのデータ	×	×	×	○
時系列データの表示	○	○	×	×
表示更新の間隔	5min.	5min.	<1sec.	<1sec.
ノードごとの通信頻度	20sec.	5min.	1min.	~2sec.
可視化の遅延	<5min.	<5min.	3min.	~2sec.
可視化が行われる場所	Server	Server	Client	Client

- リアルタイム監視システムは、より取得負荷が高い情報を必要とする

性能比較(CPU)

	Ganglia	Munin	RTM	VGXP
CPU(when monitoring 500nodes)				
更新頻度	20sec.	-	-	2sec.
Server CPU	9.1%	-	-	1.1%
Nodes CPU(各ノードの平均値)				
obtaining system-wide data	0.0324%	-	-	0.0460%
obtaining per process data	-	-	-	0.3286%

- VGXPは更新頻度が高い割にはCPU使用率はGangliaと同程度で済んでいる
- プロセス情報の取得負荷は高い

性能比較(ネットワーク)

	Ganglia	Munin	RTM	VGXP
Network(client <-> server)				
更新頻度	5min.	5min.	1min.	2sec.
更新あたりの転送量	0.9MB(per rendering)	1.9MB(per rendering)	90KB(raw) 10KB(compressed)	50KB~ 300KB

- 値の予測を行い、その予測が当たっている間はデータを送らないことで、転送量をさらに減らすことができる (Kamoshida et al. (CCGrid2008))

リアルタイム監視システムに求められること

- 大量の情報の効率的な収集
  - 運用中に明らかになった課題に対する取り組みについて紹介
- 分かりやすい情報可視化
- 多数のクライアントへの効率的なデータ配信

VGXPが運用されているシステム

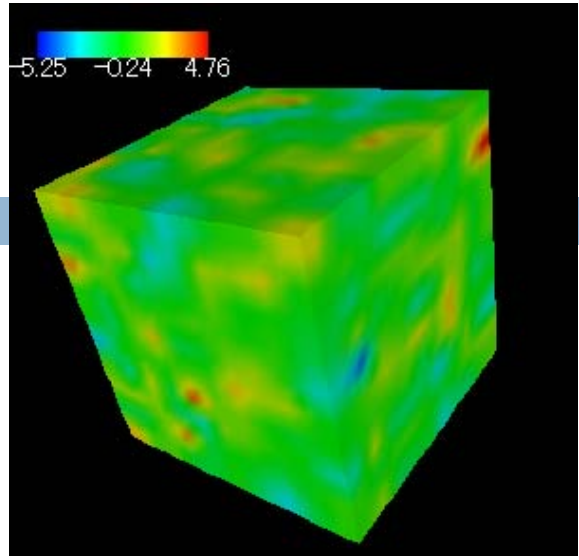
- InTriggerプラットフォーム
  - http://www.intrigger.jp
  - 国内15拠点、約300ノード

- HA8000クラスタシステム
  - 2009年2月まで運用
    - 無料の試行期間+α

見えてきた課題

- ノードで情報収集デーモンを実行していると一部の並列アプリケーションが異常に遅くなることがある
  - アプリケーション以外のOSプロセス(ノイズと呼ばれることがある)の影響として、以前から知られていた問題(Jones et al. (SC03)など)だが、同期を頻繁に行うアプリケーションの場合、比較的少ないノードのジョブでも大きな問題になるため、対策が必要
  - 極端なケースでは64ノードで3倍の時間がかかることも

実験



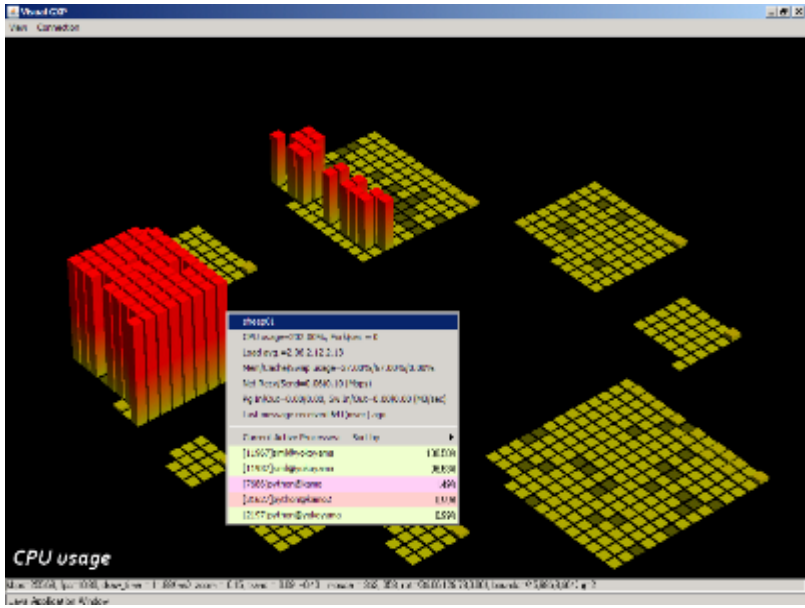
- 並列アプリケーションを、
  監視プロセスの有無を変えて
  実行し、処理時間の違いを調べる
- アプリケーションの概要
  - 不均質な物性値分布を有する立方体形状における三次元静的弾性問題を並列有限要素法(Finite-Element Method，FEM) で解く
  - 連立一次方程式はSGS(Symmetric Gauss-Seidel) を前処理手法として、共役勾配(Conjugate Gradient，CG)法を使用して解く

通信の概要

- アプリケーション内で行われる通信
  - 階層型領域間境界分割(HID)によって担当領域を分担
  - (1)隣接領域を担当するプロセスと境界部分のデータを交換。このために1対1通信による送受信を平均7 ～8回実行
  - (2)全プロセスでMPI_Allreduceを3回実行(全体の同期を伴う処理)
  - (1),(2)を収束するまで繰り返す
    - 実験で与えた入力データでは1250回程度

監視プロセス(pinfo)

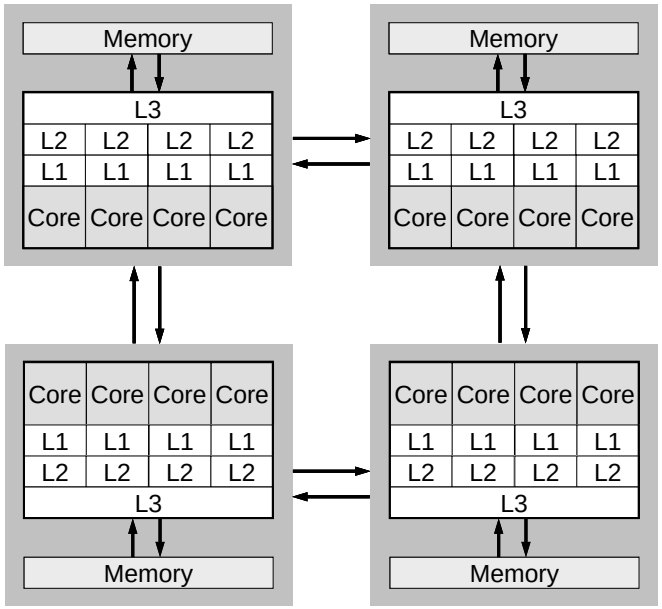
- 監視プロセスの動作概要
  - 定期的(500msに1回)に動作
    - VGXPのデフォルト頻度の4倍
  - システム情報(メモリ使用率やLoad averageなど)と各プロセスの情報(プロセスごとのCPU使用率など)を取得
  - 1回の情報取得時間は15～20ms

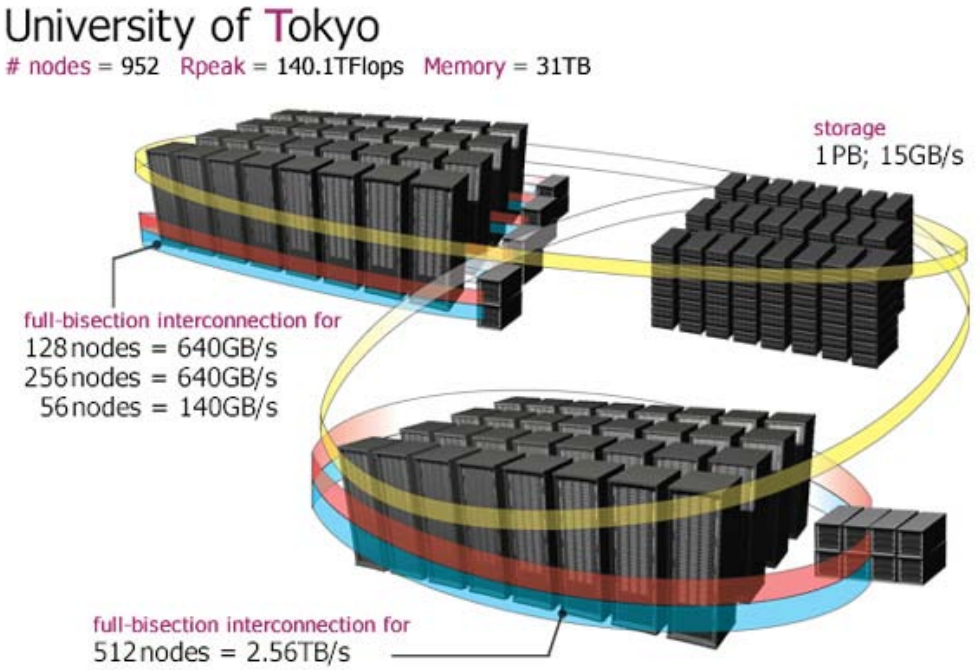


実験環境

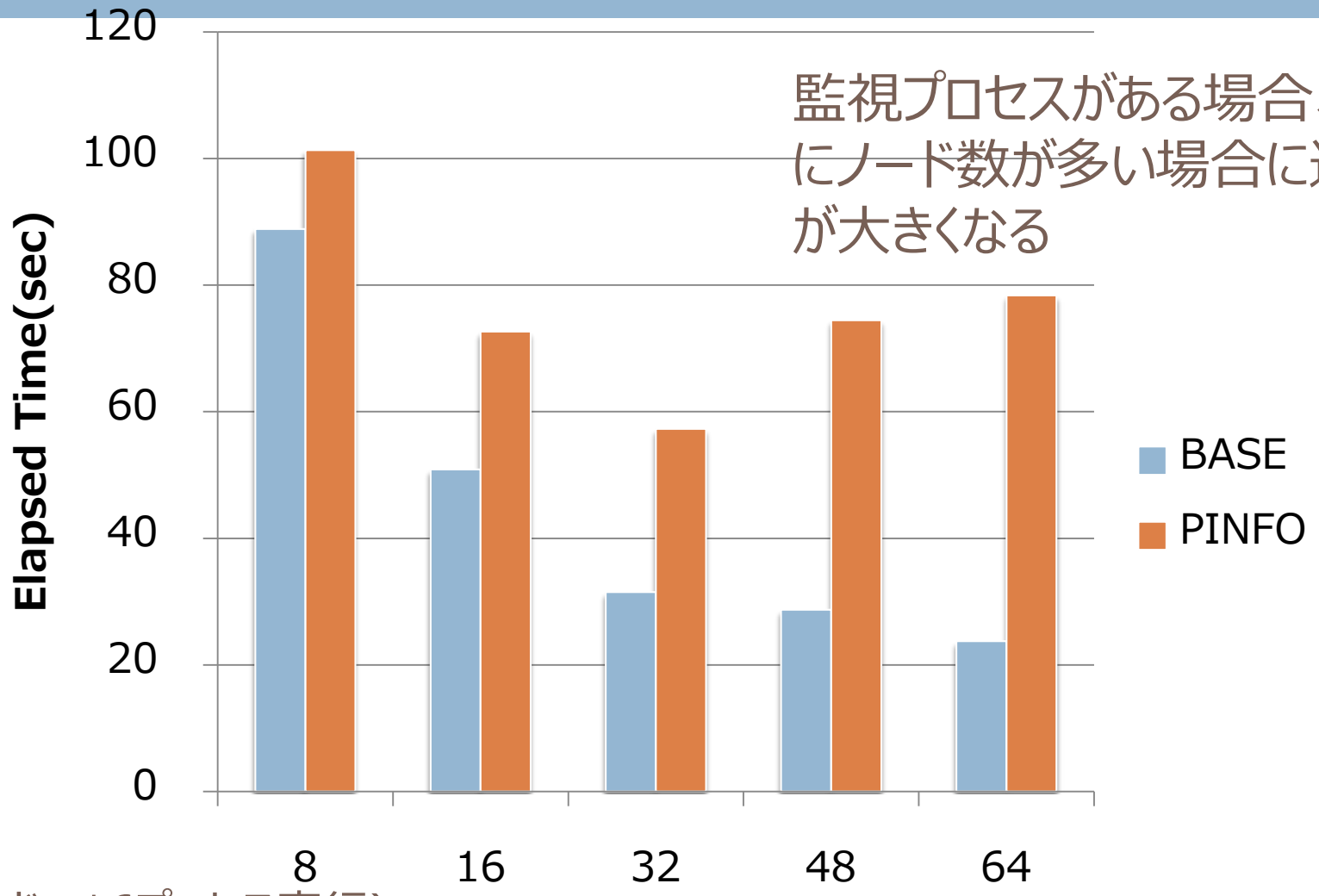
- 東大 HA8000クラスタシステム(T2Kスパコン東大版)
  - 32GB memory, Myri-10G x4
    Linux kernel 2.6.18

Quad-core Opteron(2.3GHz) x4





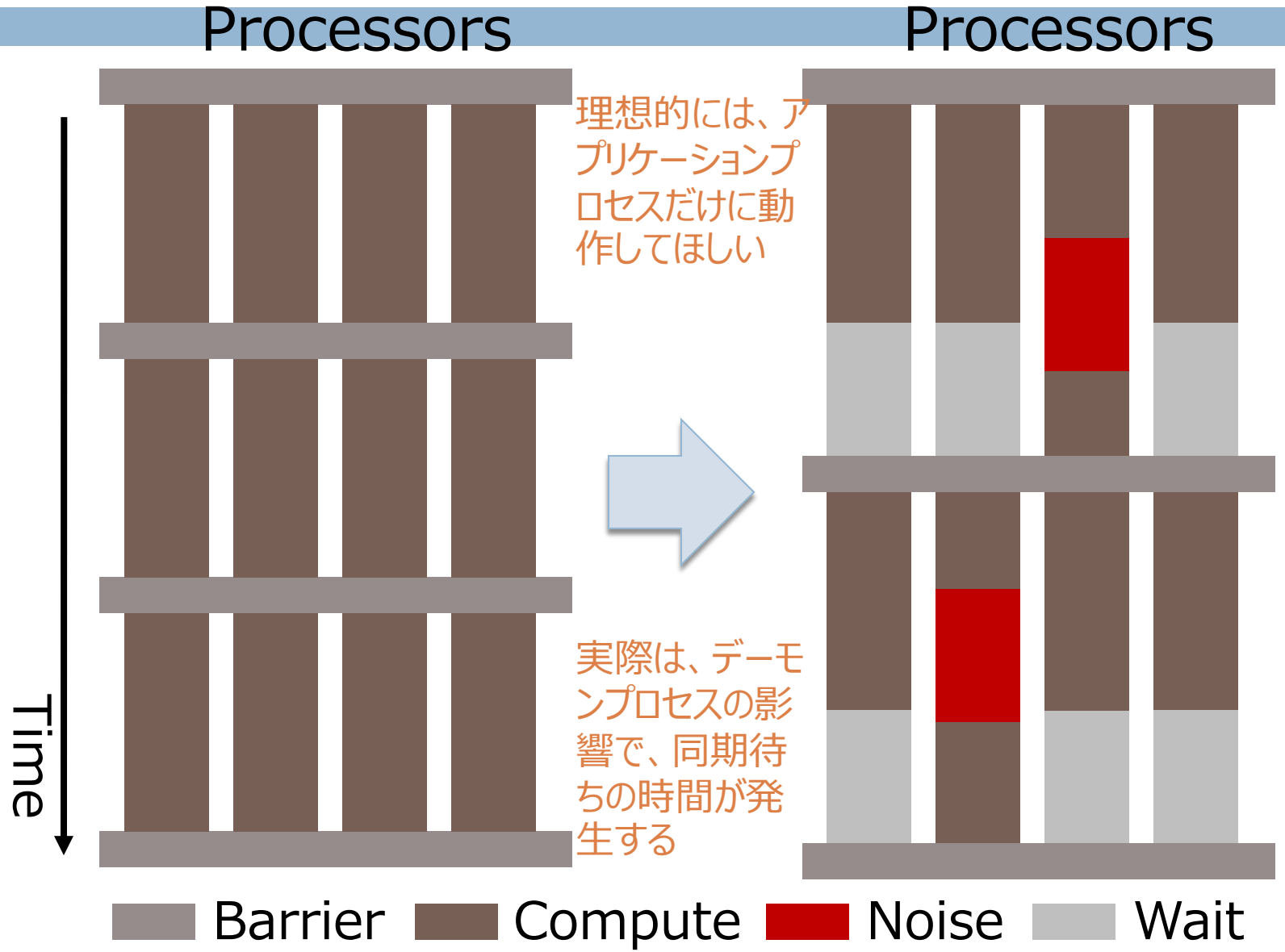
実行時間



監視プロセスがある場合、特にノード数が多い場合に遅延が大きくなる

(各ノードで16プロセス実行)

ノイズのアプリケーションへの影響



並列プログラム遅延への対策

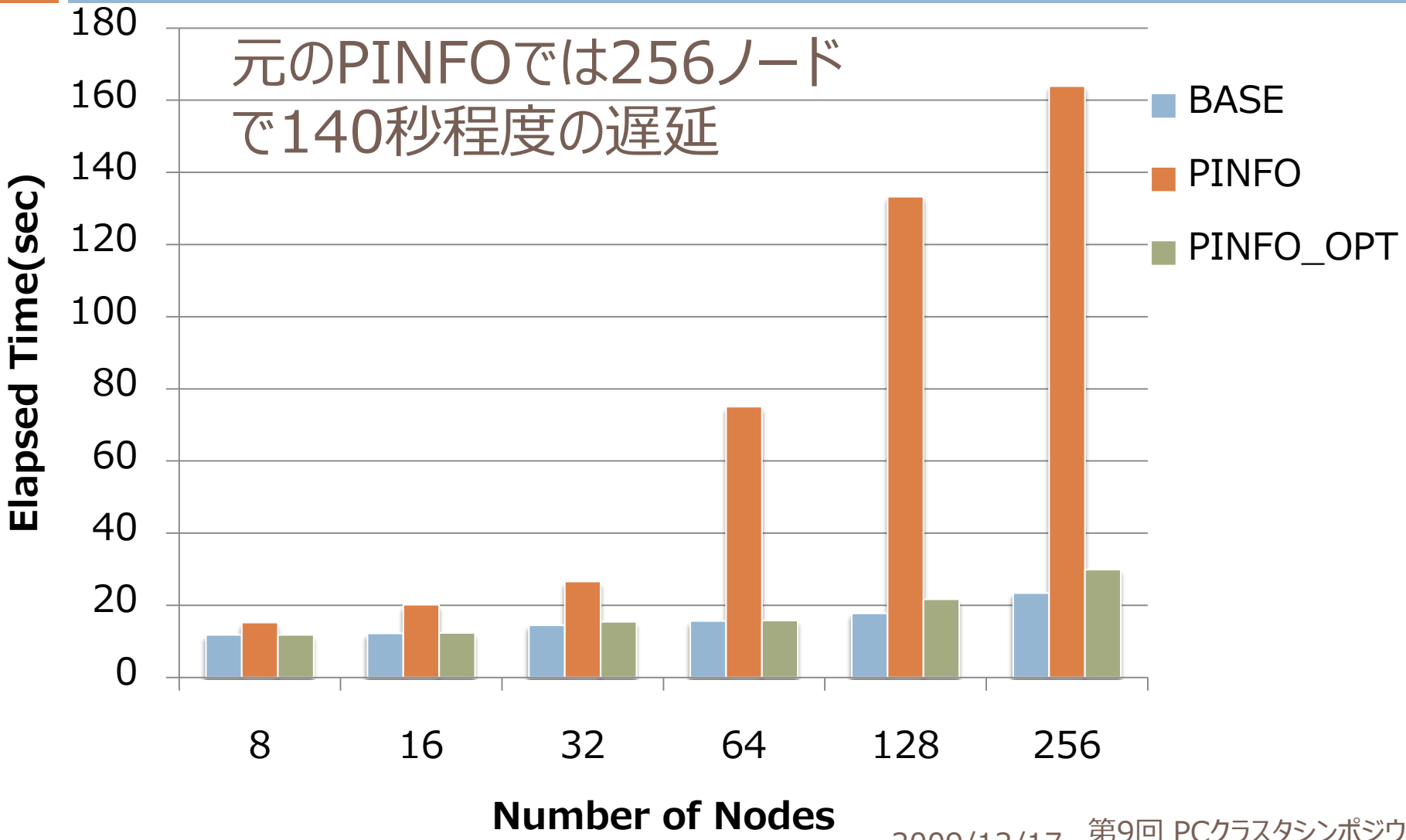
(Kamoshida et al. (SWoPP2009))

- 監視プロセスについて以下の改良を行う
- ノード間の同期
  - 他のノードとなるべく同時にスケジュールされるようにする
- 個々の処理の効率化
  - 呼び出すシステムコールの回数を減らす
  - プロセスごとの情報収集負荷を下げる

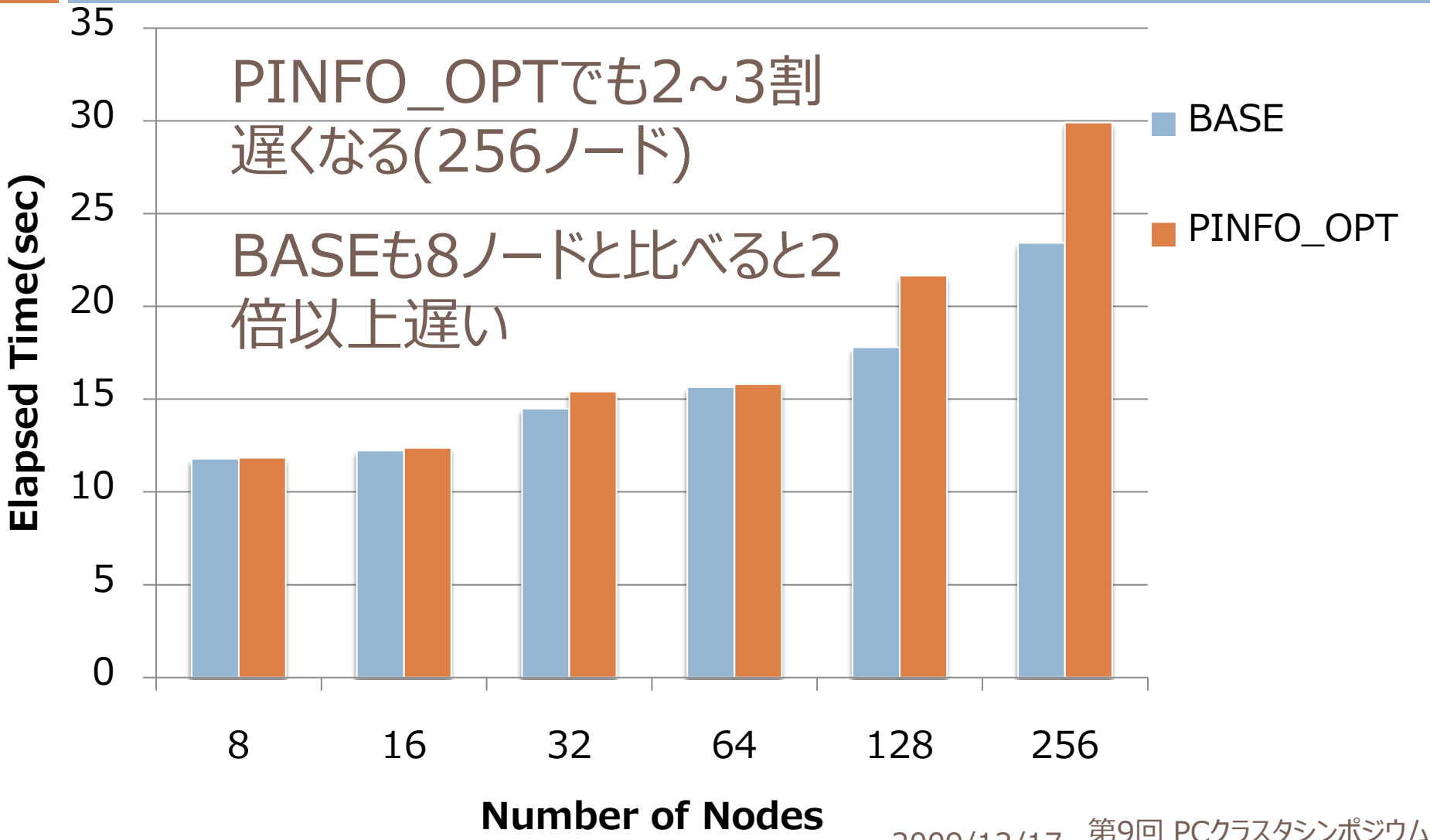
実験

- ベンチマーク
  - 各ノードで16プロセス実行
  - 約1.25msのローカル計算
  - 全プロセスでAllreduce
  - 以上を8192回繰り返し
- 実行環境
  - 東京大学HA8000クラスタシステム
    - 8～256ノードを使用
  - 監視なし(BASE)・監視あり(PINFO)・改良版監視プロセスあり(PINFO_OPT)を比較

実行時間


元のPINFOでは256ノードで140秒程度の遅延
Elapsed Time(sec)
180
160
140
120
100
80
60
40
20
0
8
16
32
64
128
256
Number of Nodes
BASE
PINFO
PINFO_OPT

実行時間(除くPINFO)



実験結果(まとめ)

- 256ノードでの実験結果
  - 元のPINFOでは256ノードで140秒程度の遅延
  - 改良版PINFOでも2~3割遅くなる(256ノード)
  - 監視プロセスがない場合でも8ノードと比べると2倍以上遅い
- BASEのデータの揺れも大きい
  - 標準偏差が2割程度
  - 改良版PINFOでも大差ない

その他の課題

- 効率的なデータ配信
  - サーバ負荷の集中を避ける転送アルゴリズム
  - いろいろなクエリを受け付けられる枠組みの構築(cf. Content-based pub-sub system)
- 必要な情報を可視化
  - 大量の情報から意味のある変化を自動的に選択
  - 対象のクラスタリング

まとめ

- クラスタ監視ソフトVGXPの仕組みと設計の背景を説明
  - 数百～1,000ノード規模のクラスタシステムで稼働している
- 運用中に見えてきた課題とそれに対する対策を説明
  - 並列プログラムの遅延の問題は大きく改善されたが、それでも、256ノードで20%程度遅くなる場合もまだ存在する
  - 遅延の原因を、監視システムによるものとそれ以外に分けて分析する必要がある