

IC2004

# グリッド上での並列分枝限定法 アプリケーション

合田 憲人 東京工業大学／科学技術振興機構さきがけ  
大角 知孝 東京工業大学

# グリッド上での高性能計算

## □グリッド計算

- インターネット上の計算資源を安全に, 安定して, 安易に利用することにより高性能計算を実現
  - 高性能計算のコスト軽減, 未知の大規模問題の求解

## □問題点

- 計算に伴うオーバーヘッドの影響大
  - WAN上の通信, セキュリティに関する処理(認証, 暗号化等)
- グリッド上で効率よく実行可能なアプリケーションでは, ある程度のタスク粒度が必要.
  - [Goux 2000]: 190 sec/task
  - [Neary 2002]: 177-430 sec/task
- 大量の細粒度タスクから構成される大規模問題も存在. (例: 最適化問題)

# グリッド上での最適化問題の高速求解

□目的: グリッド上での最適化問題求解  
アプリケーションの高性能実行

□グリッド上での最適化問題求解アプリケーションの  
並列化

□分枝限定法アプリケーション

□階層的マスタ・ワーカ方式による並列化

□アプリケーションスケジューリング技術の開発

□グリッド上でのアプリケーションの実装と実証実験

□成果

□最適化問題分野に対して, グリッド上での高性能計算利用への道を開く.

□グリッドアプリケーションの一分野を開拓する.

# 分枝限定法

## □最適解の探索

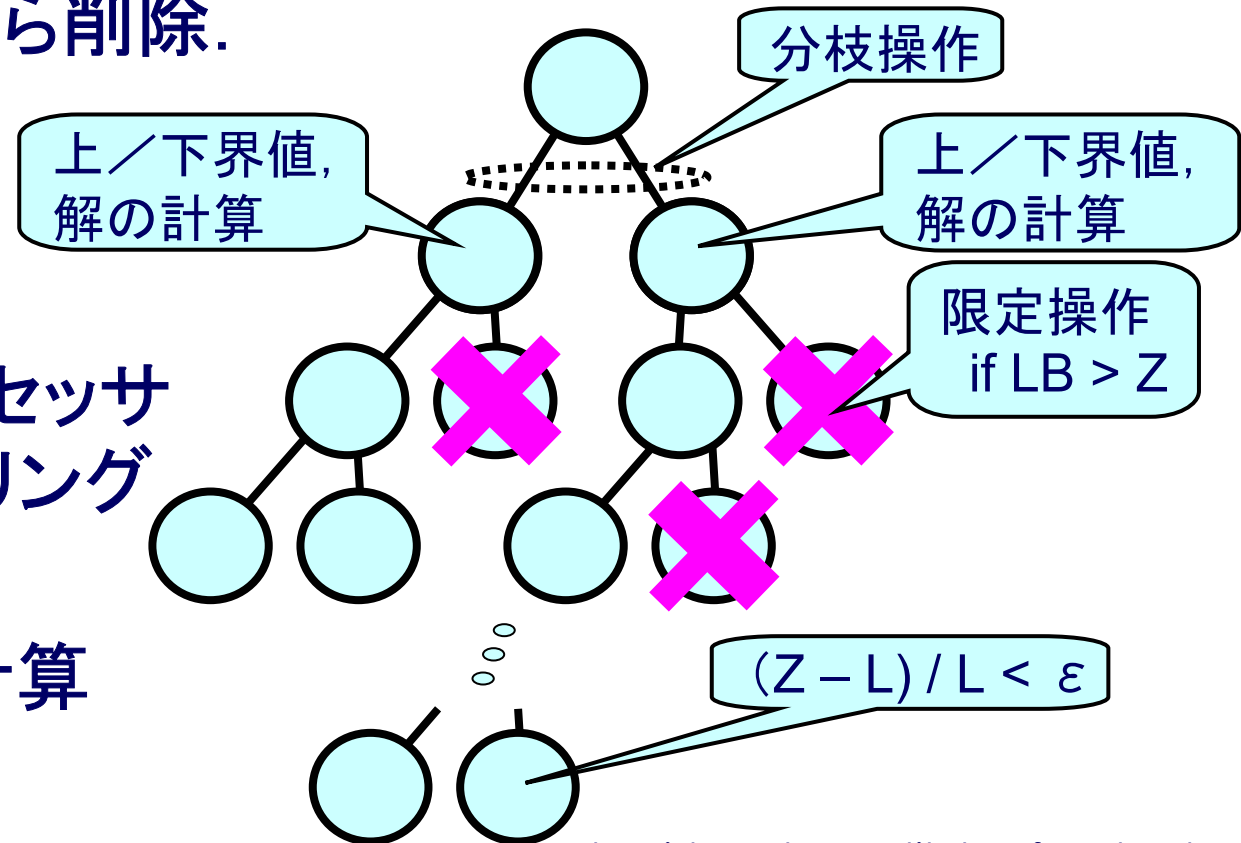
- ❑問題を複数の小規模な問題(子問題)に再帰的に分割して, 各子問題について解の計算を行う.
- ❑最適解の存在しない子問題(例: 下界 > 暫定値)は探索木から削除.

## □应用

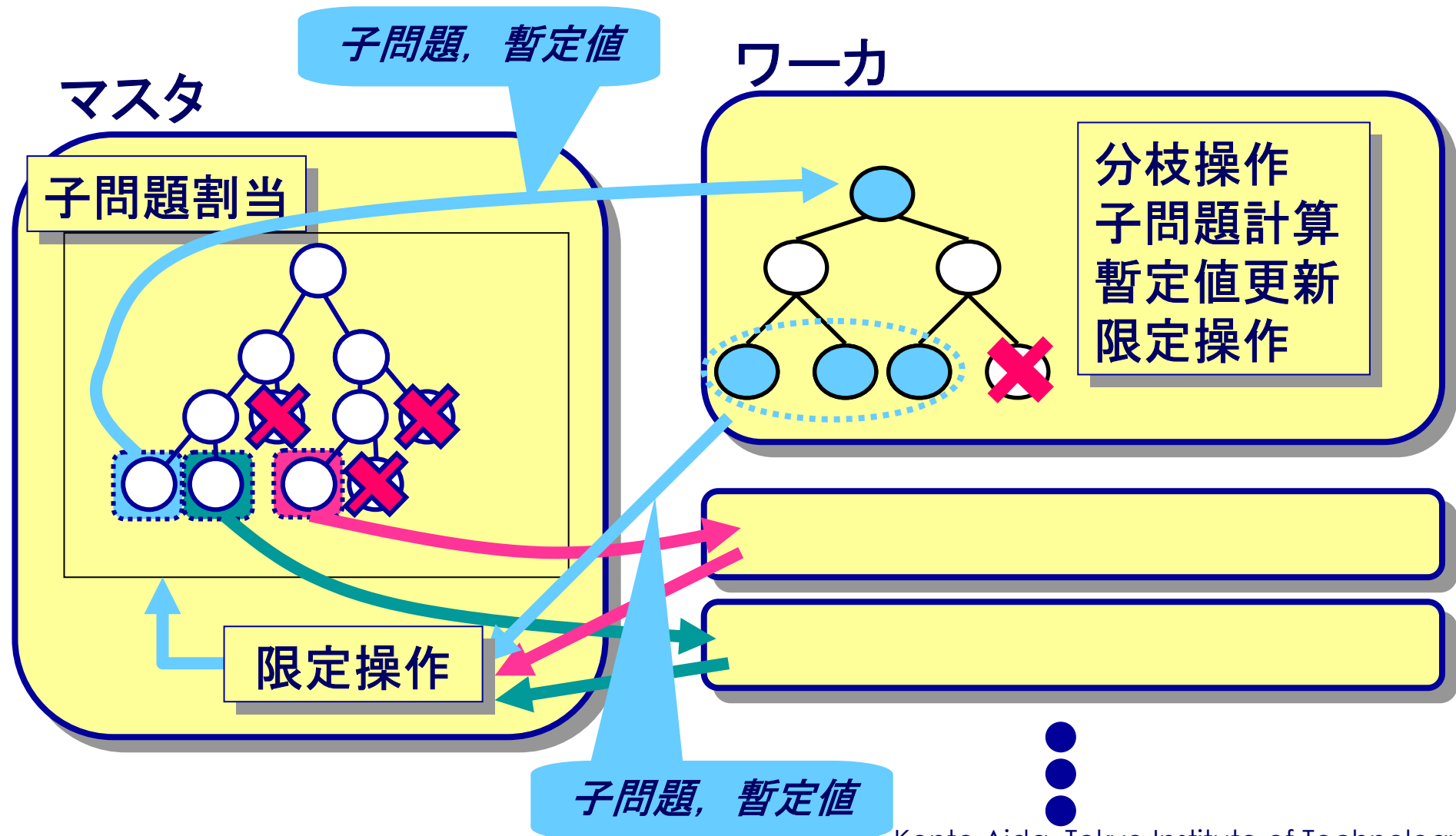
- OR
- 制御工学
- マルチプロセッサスケジューリング

## □ 並列化

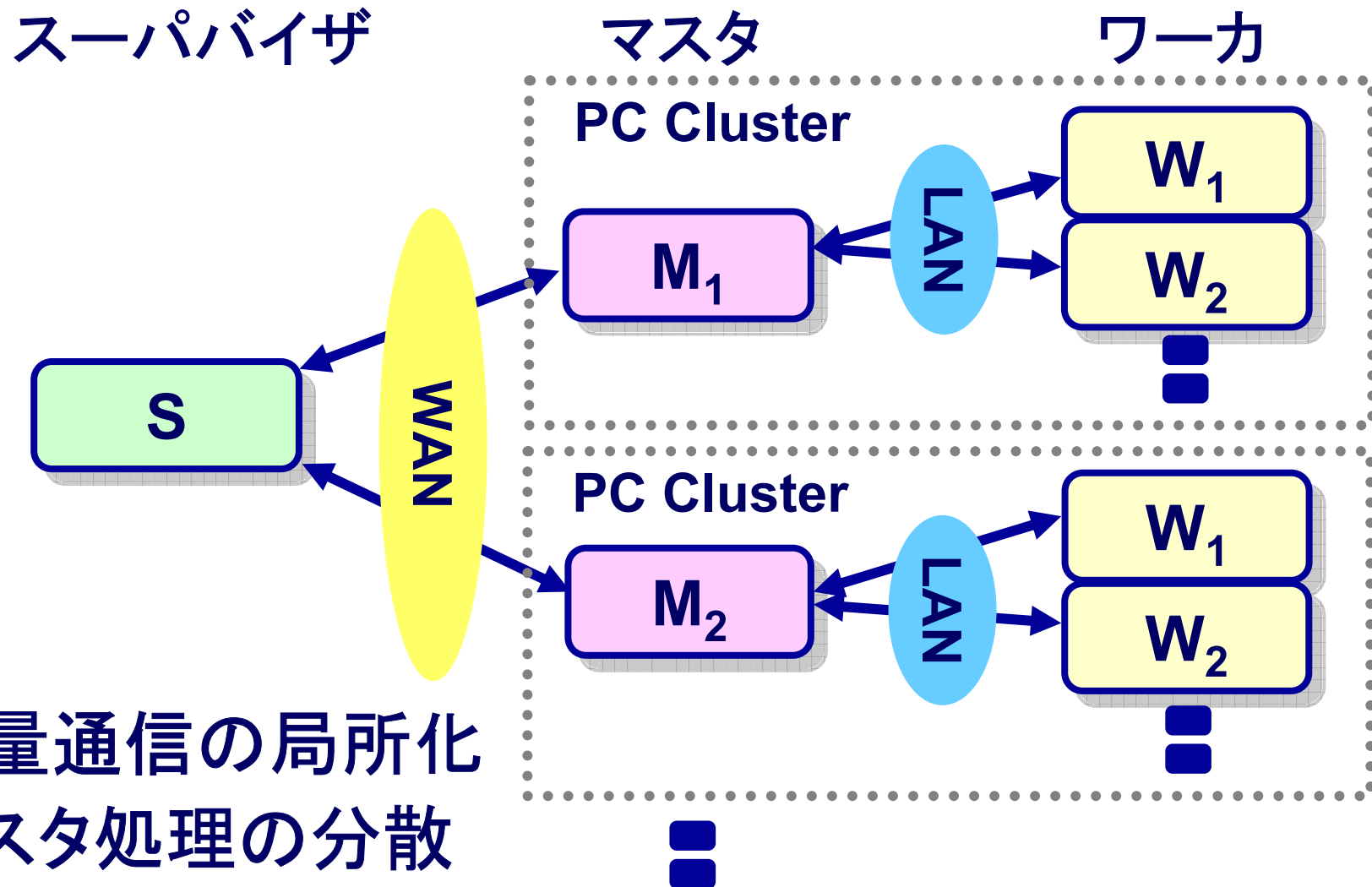
## 各子問題の計算は独立.



# マスタ・ワーカ方式による 分枝限定法の並列化



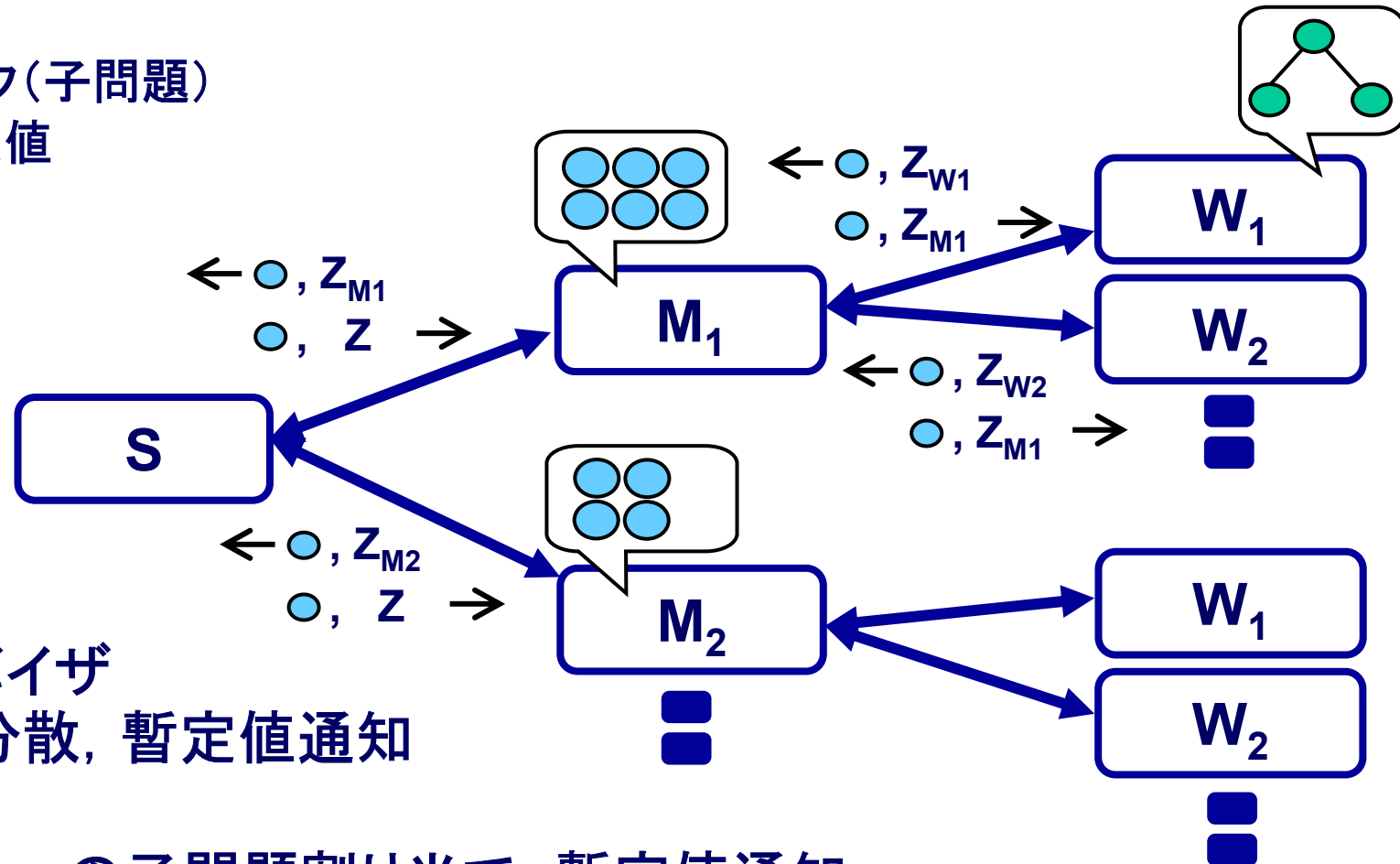
# 階層的マスタ・ワーカ方式



- 大量通信の局所化
- マスタ処理の分散

# 階層的マスタ・ワーカ方式を用いた 並列分枝限定法

● : タスク(子問題)  
Z : 暫定値



□ スーパーバイザ

負荷分散, 暫定値通知

□ マスタ

ワーカへの子問題割り当て, 暫定値通知

□ ワーカ

子問題計算

# GridRPC

## □GridRPC

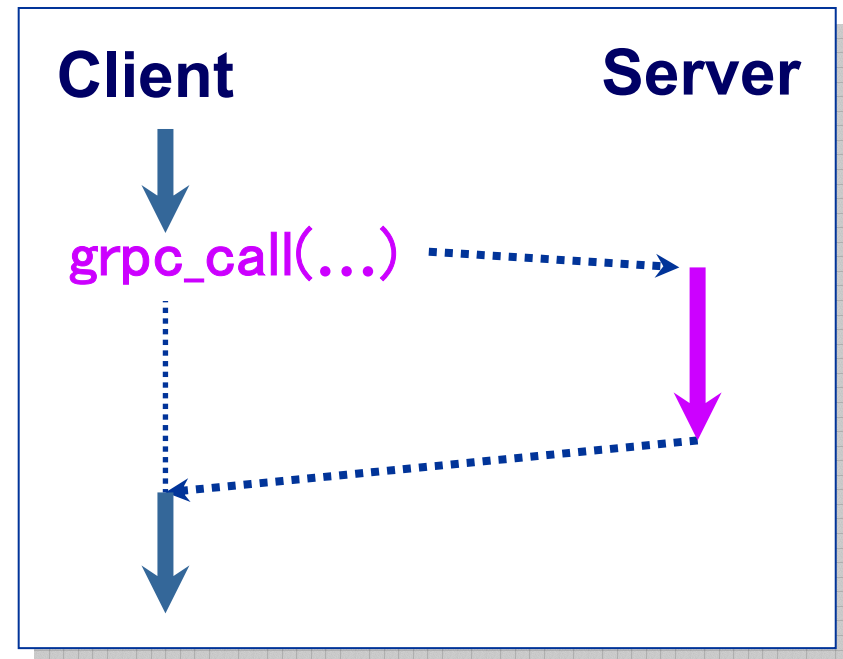
- グリッド上でクライアント・サーバ型のRPCを実現するプログラミングモデル
- GGFにおける標準化

## □Ninf-G

- GridRPCのreference implementation
- Globus Toolkit上に実装
- GSIによる安全なRPC

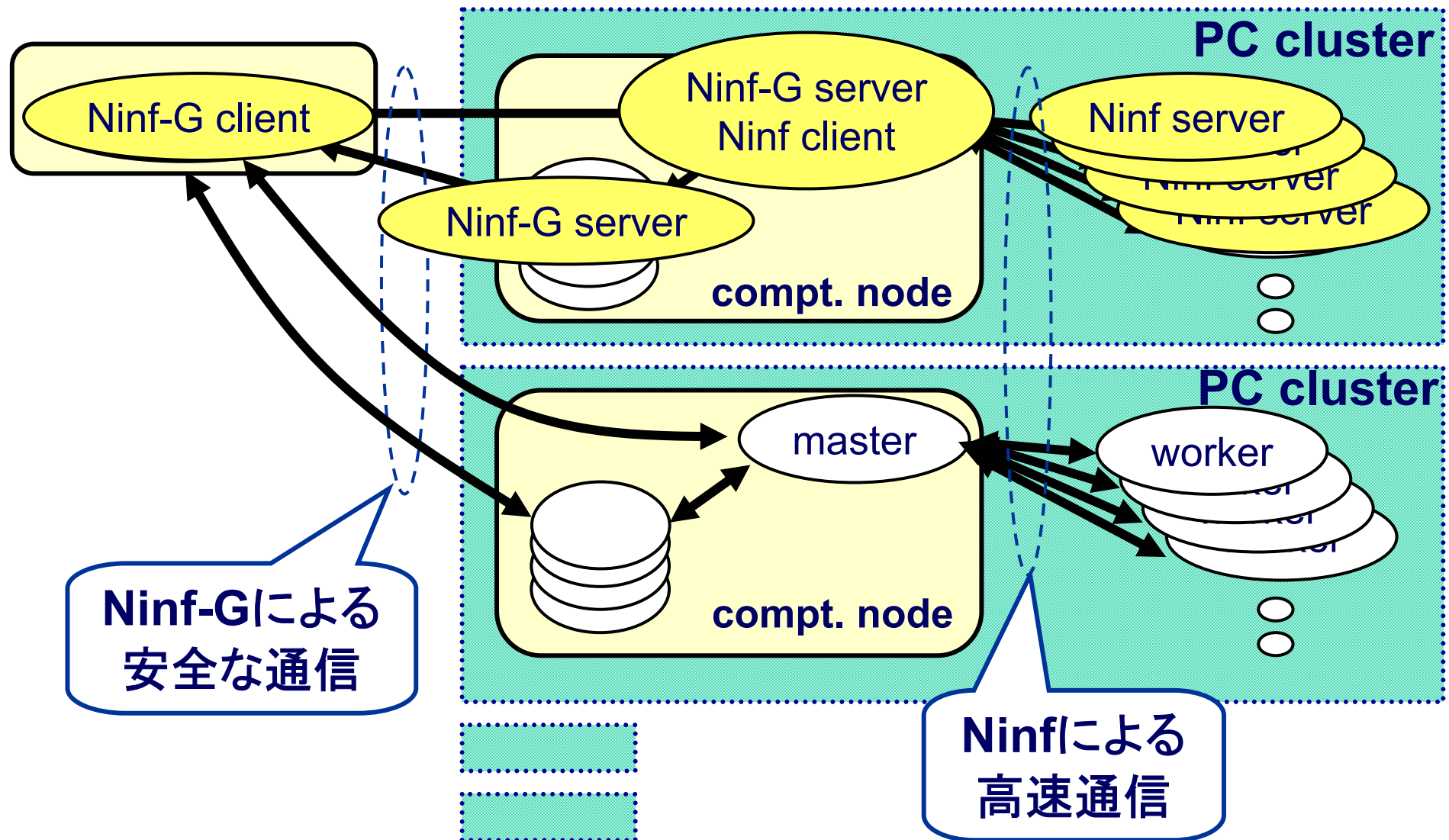
## □Ninf

- Ninf-Gの前身として開発
- 高速なRPC





# GridRPCによる実装(続)



# グリッド実験環境

Globus Toolkit 2.\*  
Ninf-G 1.1.1

## Sdpa

dual Athlon 2GHz  
東京電機大（埼玉）



RTT=14ms

## Mp

dual Athlon 1.6GHz  
徳島大（徳島）



RTT=20ms

client/  
supervisor



**LAN**

RTT=0.04ms

**Super  
Titanet**

**SINET**

RTT=1ms



## Blade

dual PIII 1.4GHz  
東工大（横浜）

## Prestoll

dual Athlon 1.6GHz  
東工大（東京）

Kento Aida, Tokyo Institute of Technology

# グリッド実験環境(続)

	specification of a single node	Grid software	RTT/ distance
Client PC	PIII 1.0GHz, 256MB mem. 100BASE-T NIC	GTK 2.2 Ninf-G 1.1.1	
Blade	PIII 1.4GHz x2 512MB mem. 100BASE-T NIC	GTK 2.2 Ninf-G 1.1.1	0.04ms
Prestolll	Athlon 1.6GHz x2, 768MB mem. 100BASE-T NIC	GTK 2.4 Ninf-G 1.1.1	1ms 30km
Mp	Athlon 1.6GHz x2 512MB mem. 100BASE-T NIC	GTK 2.4 Ninf-G 1.1.1	20ms 500km
Sdpa	Athlon 2GHz x2, 1024MB mem. 1000BASE-T NIC	GTK 2.4 Ninf-G 1.1.1	14ms 50km

# BMI固有値問題

## □定義

双線形行列関数  $F(x,y)$  の最大固有値を最小化するベクトル変数  $x, y$  を求める問題.

$$F(x,y) := F_{00} + \sum_{i=1}^{n_x} x_i F_{i0} + \sum_{j=1}^{n_y} y_j F_{0j} + \sum_{i=1}^{n_x} \sum_{j=1}^{n_y} x_i y_j F_{ij}$$

$$F: R^{n_x} \times R^{n_y} \rightarrow R^{m \times m}$$

$$F_{ij} = F_{ij}^T \in R^{m \times m} \quad (i=0, \dots, n_x, j=0, \dots, n_y)$$

## □応用

### □制御工学

□ヘリコプター制御, ロボットアーム制御

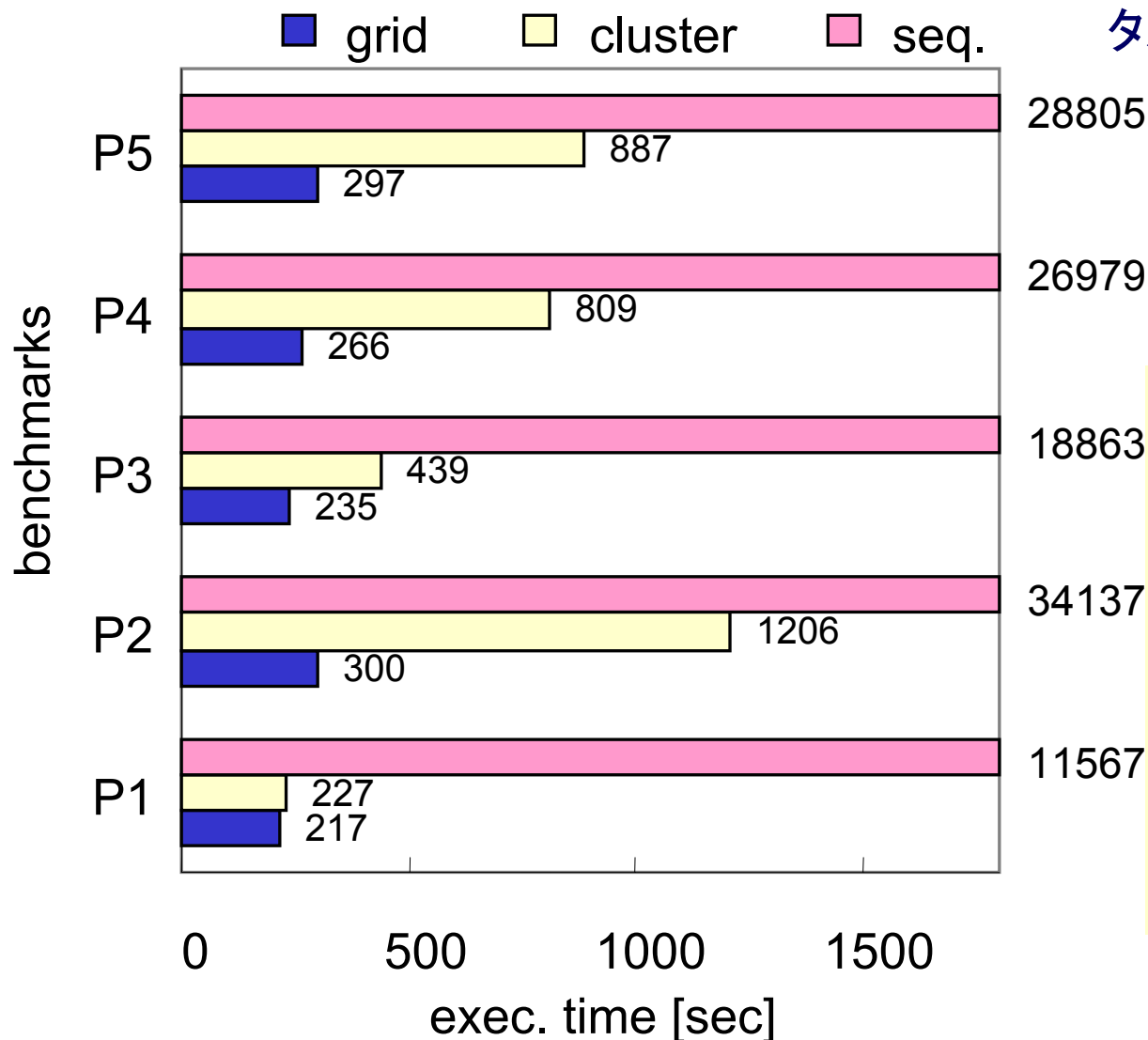


### □OR

□大規模問題求解への挑戦

# グリッド上でのアプリケーション実行時間

P1-P5:  $n_x=6$ ,  $n_y=6$ ,  $m=24$ ,  
タスク実行時間 = 0.5sec程度



seq: Blade (1CPU)  
cluster: Blade (73CPUs)  
grid: 4sites (348CPUs)

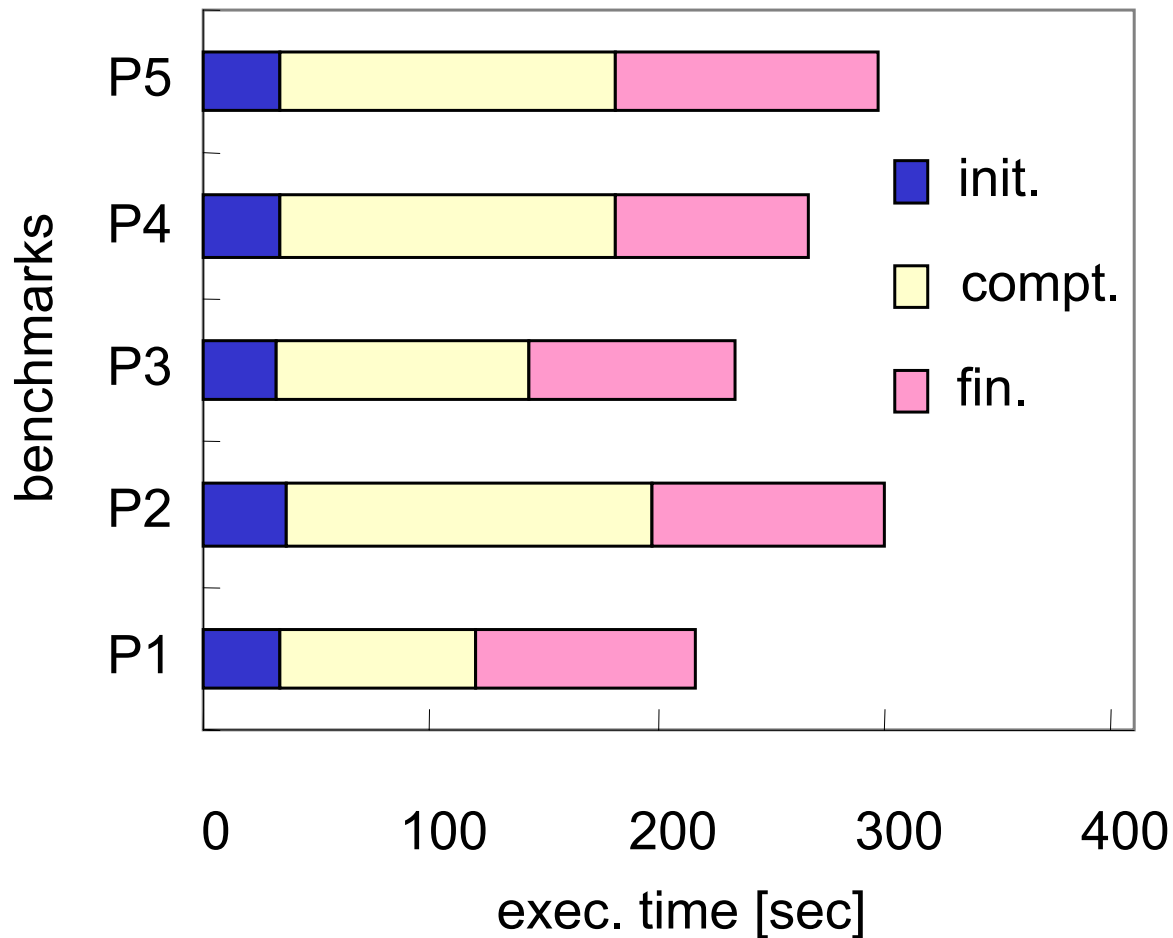
## 例: P2

逐次計算: 9.5h

グリッド計算: 5min

細粒度アプリに対する階層的マスター・ワーカ方式の有効性を確認.

# グリッド上での実行時間内訳



- 終了処理オーバーヘッドによる性能低下大
- 解は終了処理以前にユーザに提示される.

# PCクラスタ間負荷分散

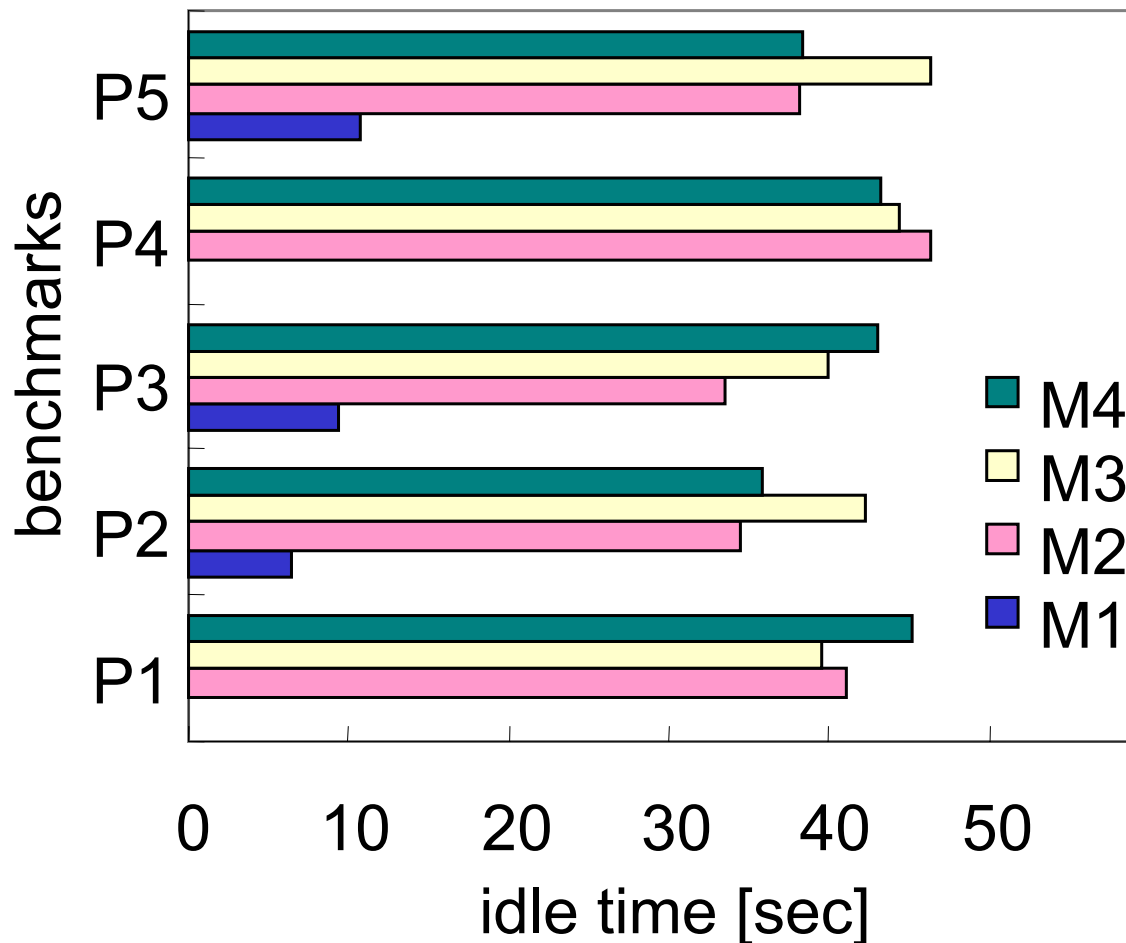
## □supervisor

- master(=PCクラスタ)上の負荷(=未処理子問題数)を問い合わせ
- アイドル状態のマスタを発見する度に, タスクを再分散.
  - タスク実行履歴の保存
  - PCクラスタの性能に応じて分配タスク数を決定.

$$N_{\text{task}(i)} = N_{\text{task}(\text{total})} \frac{T_{\text{task}(i)} N_{\text{workers}(i)}}{\sum_j (T_{\text{task}(j)} N_{\text{workers}(j)})}$$

# PCクラスタ上でのアイドル時間

P1-P5:  $n_x=6$ ,  $n_y=6$ ,  $m=24$

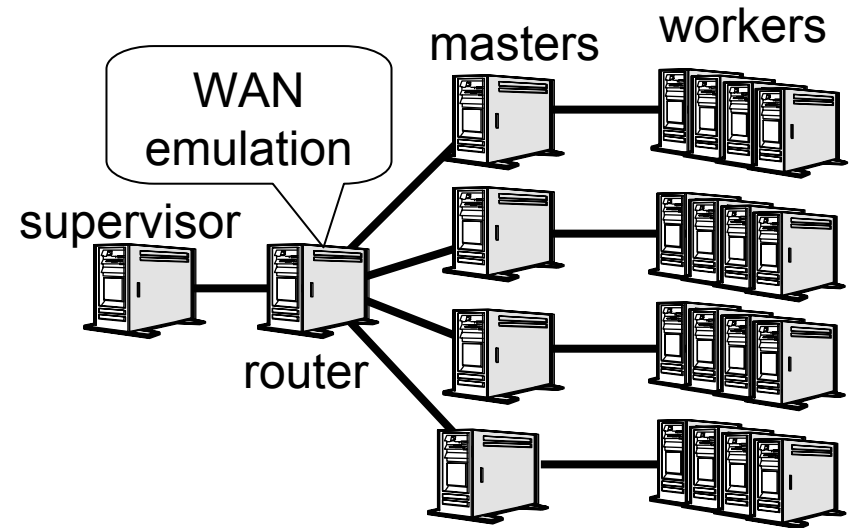
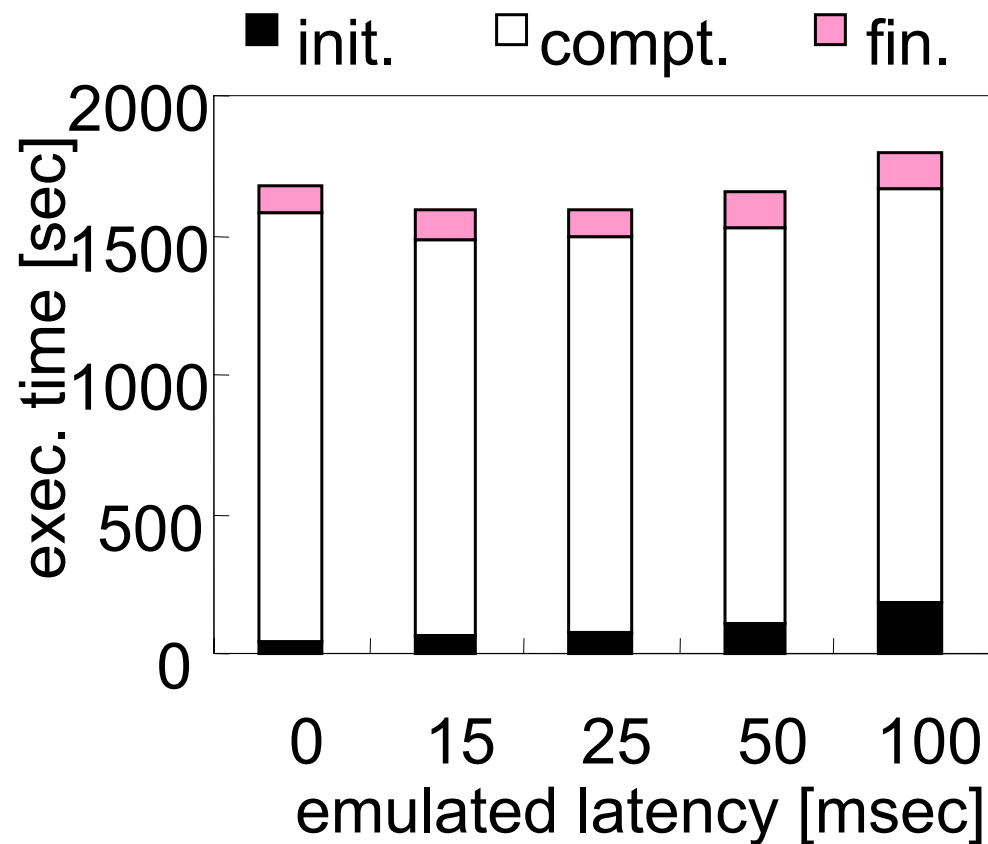


- M1以外は, 40 [sec]程度のアイドル時間
- アイドル時間のほとんどは, アプリケーション実行開始時のタスク数が少ないことが要因.
- 負荷分散は, 適切に実現されている.

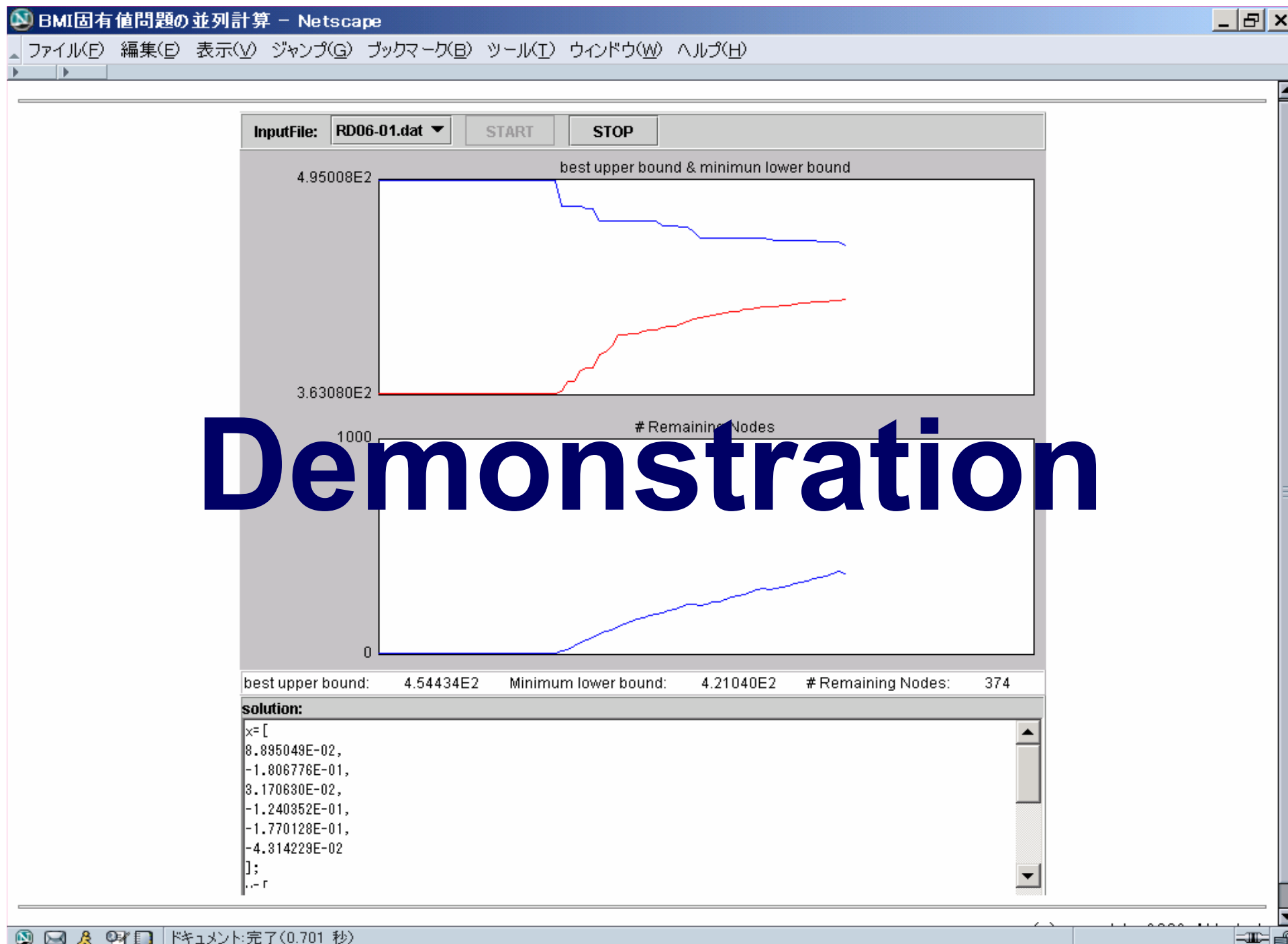


# 擬似グリッド実験環境上での実行時間

P2:  $n_x=6$ ,  $n_y=6$ ,  $m=24$



□ 高レイテンシの条件下でも、アプリケーションの性能低下は小さい。  
→ 階層化の効果



# グリッド実験環境

Globus Toolkit 2.4.\*  
Ninf-G 2.2.0

## Sdpa

dual Athlon 2GHz  
東京電機大（埼玉）



## Mp

dual Athlon 2.0GHz  
徳島大（徳島）



## Blade

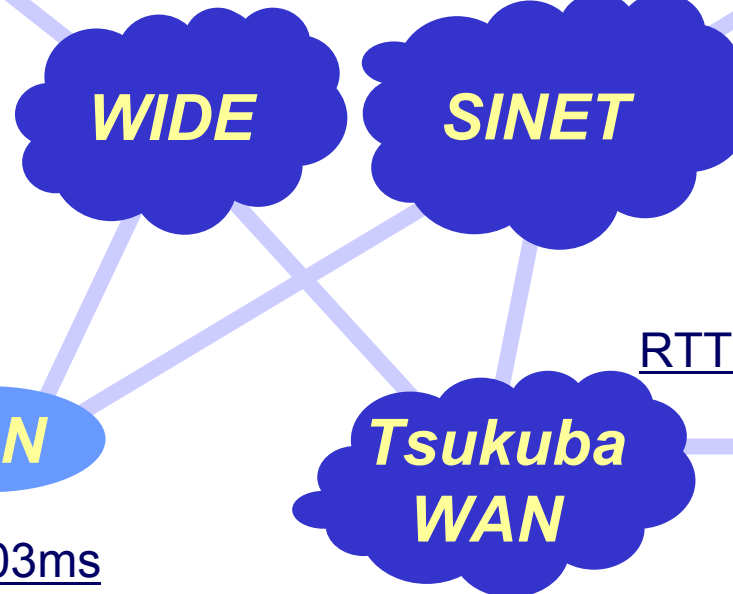
dual PIII 1.4GHz  
東工大（横浜）

RTT=12ms

RTT=28ms

RTT=6ms

RTT=0.03ms



## Prestolll

dual Athlon 1.6GHz  
東工大（東京）

Kento Aida, Tokyo Institute of Technology

## まとめ

# □グリッド上での並列分枝限定法アプリケーションの実装と実証実験

□階層的マスタ・ワーカ方式による並列化および  
GridRPC (Ninf-G, Ninf)による実装の有効性を  
確認

## □今後の課題

## □PCクラスタ間負荷分散アルゴリズム, 擬似グリッド テストベッド

## □耐故障性

## Acknowledgments:

- JST さきがけ「情報基盤と利用環境」
- JST ACT-JST「コモディティグリッド技術によるテラスケール大規模数理最適化」
- Ninfプロジェクト