

Ninf システムにおける ジョブスケジューラの実装と 予備的評価



竹房あつ子, 中田秀基, 合田憲人,
小川宏高, 松岡聡, 長嶋雲兵

<http://ninf.etl.go.jp/>

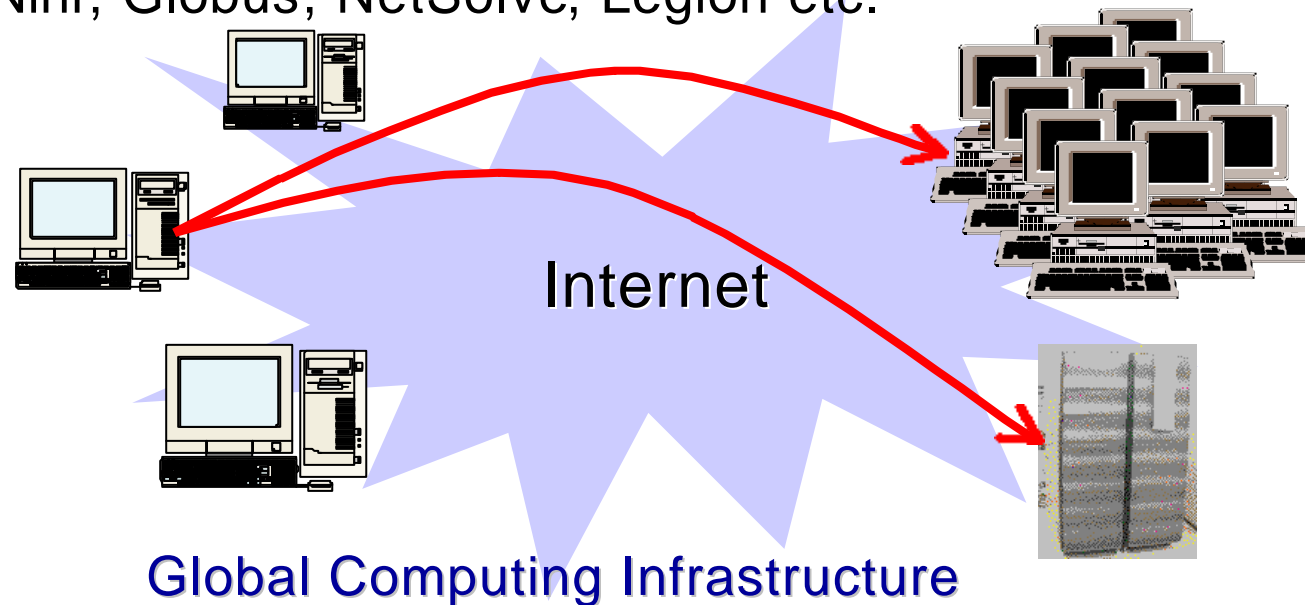
グローバルコンピューティング

■ ネットワーク技術の発展

計算・情報資源を利用した広域並列分散計算

→ **グローバルコンピューティング**

Ninf, Globus, NetSolve, Legion etc.



グローバルスケジューリング

- 不均質かつ変動する環境下で，ユーザの要求性能を満たす→ **グローバルスケジューリング**

- **ジョブスケジューリング**

システム全体の複数ジョブの総実行時間の短縮

- Matchmaker(Condor) , Nimrod/G

- **アプリケーションスケジューリング**

単一プログラムの応答時間の短縮

- AppLeS, Prophet

→ ジョブ/アプリケーションスケジューリングをバランスよく採り入れる

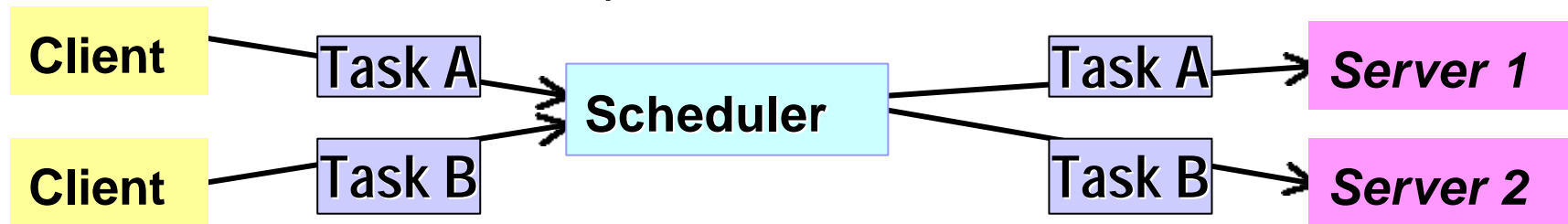
研究の目的と発表内容

- スケジューリングフレームワークの提案
- Ninf のMetaServerアーキテクチャの実装とスケジューリング手法の予備的評価
- 発表内容
 - スケジューリングフレームワークの概要
 - Ninf MetaServer アーキテクチャ
 - スケジューリング手法の評価 (実測・シミュレーション)
 - まとめ

ジョブ/アプリケーション スケジューリング

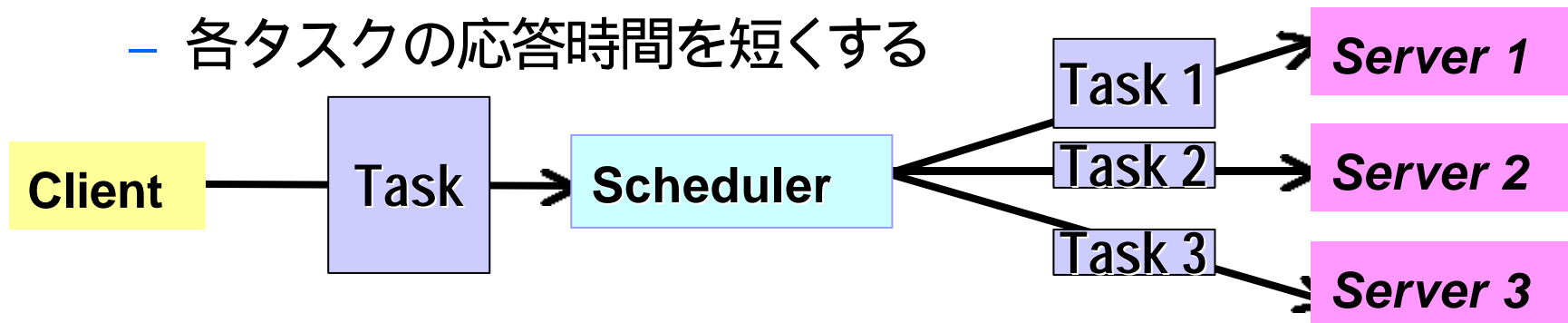
■ ジョブスケジューリング

- ユーザの要求(デッドライン, 利用計算機, ...)を満たす
- 複数ユーザのRequestの総所要時間を最短にする



■ アプリケーションスケジューリング

- アプリケーションを適切に分割
- 各タスクの応答時間を短くする



スケジューリングフレームワーク

- Job / Application Resource Allocator
 - 単一アプリケーションを複数並列タスク (Network Task) に分割
- Low-Level Scheduler
 - Network Taskをサーバへスケジューリング
サーバの負荷, ネットワークのスループット情報
- Resource Monitor / Predictor
 - リソース情報を提供
 - 負荷・スループットのモニタ
 - リソース情報の管理
 - リソース情報の予測

Job / Application Resource Allocator

Low-Level Scheduler

Resource Monitor / Predictor

Network Task スケジューリング

- ネットワークスループットモニタ

- 各クライアント・サーバ間のスループットを測定

- 計算リソースのための負荷モニタ

- 1つのノードで負荷情報を収集

- リソース情報管理

- リソース状況プレディクタ

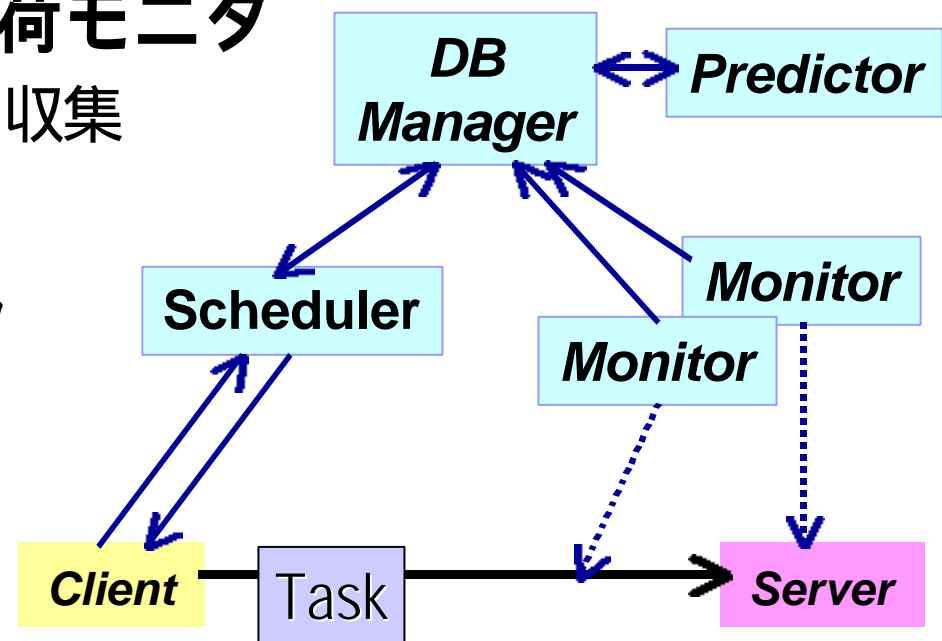
- スケジューラ

- Network Taskの割り当て

Job / Application Resource Allocator

Low-Level Scheduler

Resource Monitor / Predictor



Ninf MetaServer アーキテクチャ

■ Ninfシステム

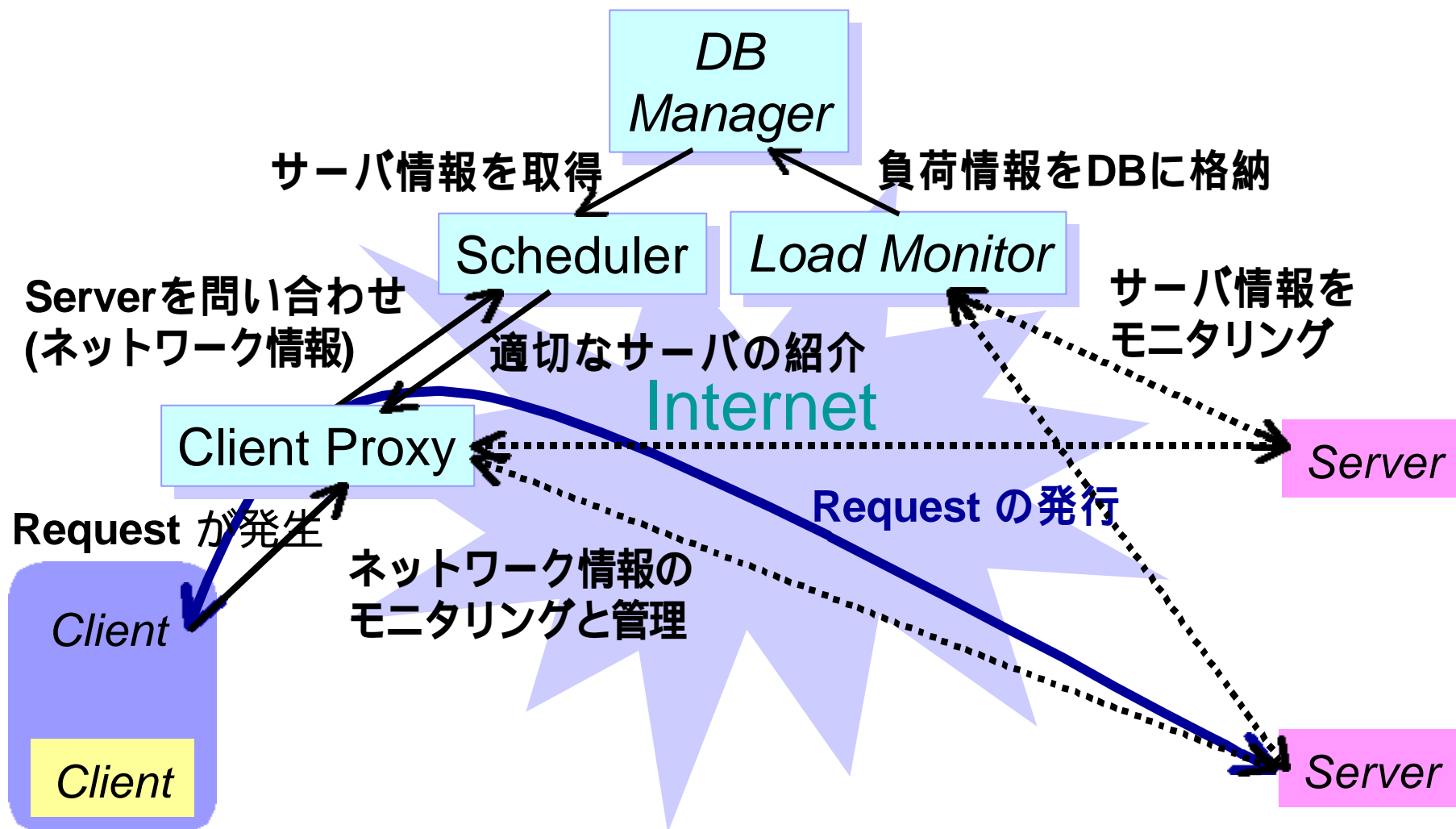
- RPCベースのグローバルコンピューティングシステム
- **MetaServer**により, スケジューリングをサポート

■ MetaServer のコンポーネント

- 計算リソースの Load Monitor
- Client Proxy
 - ネットワークのモニタリング
 - firewall のためのプロキシ
- リソース情報 DB Manager
- リソース Scheduler

**Network Task
スケジューリング**

MetaServerの処理プロセス



Schedulerでのサーバ選択

```
Define lnpack(  
  IN double a[lda:n][n],  
  IN int lda, n,  
  OUT int ipiv[n],  
  OUT int *info  
  INOUT double b[n])  
  CalcOrder (((n^3)*2)/3)+(2*(n^2))
```

Ninf IDL

- ClientProxy
 - 通信スループット
- DB Manager
 - 計算サーバ性能
- Ninf IDL
 - 通信量
 - 演算数

スケジューリングポリシーに従いサーバを選択

スケジューリング手法の予備的評価

- Ninf MetaServer を用いた評価 (実環境)
 - Client, ClientProxy - 東工大 (東京)
 - Scheduler , DB Manager , Load Monitor - 東工大
 - Server1 - 電総研 (つくば)
 - Server2 - 東工大
- シミュレーションによる評価
 - Client, MetaServer, Server が分散した環境

スケジューリング手法

■ スケジューリング

– Round Robin : RR

サーバを Round Robin で割り当てる

– 負荷 + 計算性能 : LOAD

$(L + 1) / P$ が最小となるサーバを割り当てる

(L : 平均負荷, P : サーバの性能)

– 負荷 + 計算性能 + 通信スループット : LOTH

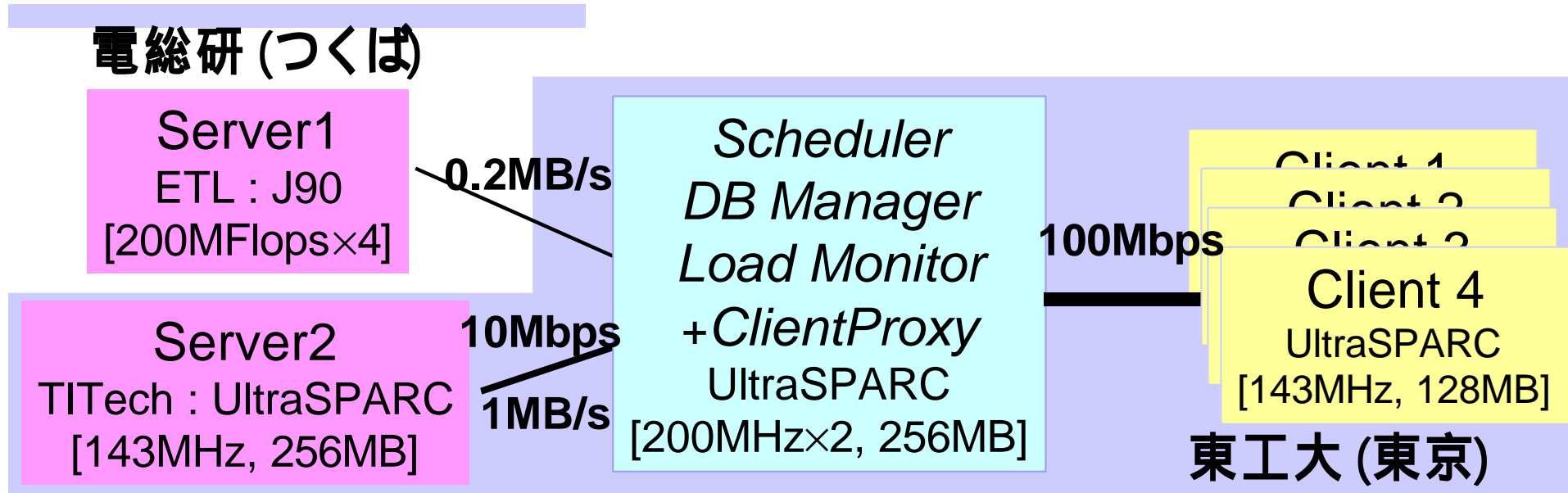
$\text{演算数} / (P / (L + 1)) + \text{通信量} / T$ が最小となるサーバを割り当てる (T : 通信スループット)

■ リソースの予測

– 通信スループット : $T / \text{ネットワークを共有するジョブ数}$

– サーバの負荷 : ジョブの割り当て \rightarrow 負荷 = 負荷 + 1

実測による評価環境



- 計算ルーチン : Linpack ($n = 600$, 固定)
演算数 $2/3n^3 + 2n^2$, 通信量 $8n^2 + 20n + O(1)$ [byte]

- Request 発行間隔 : 指数分布

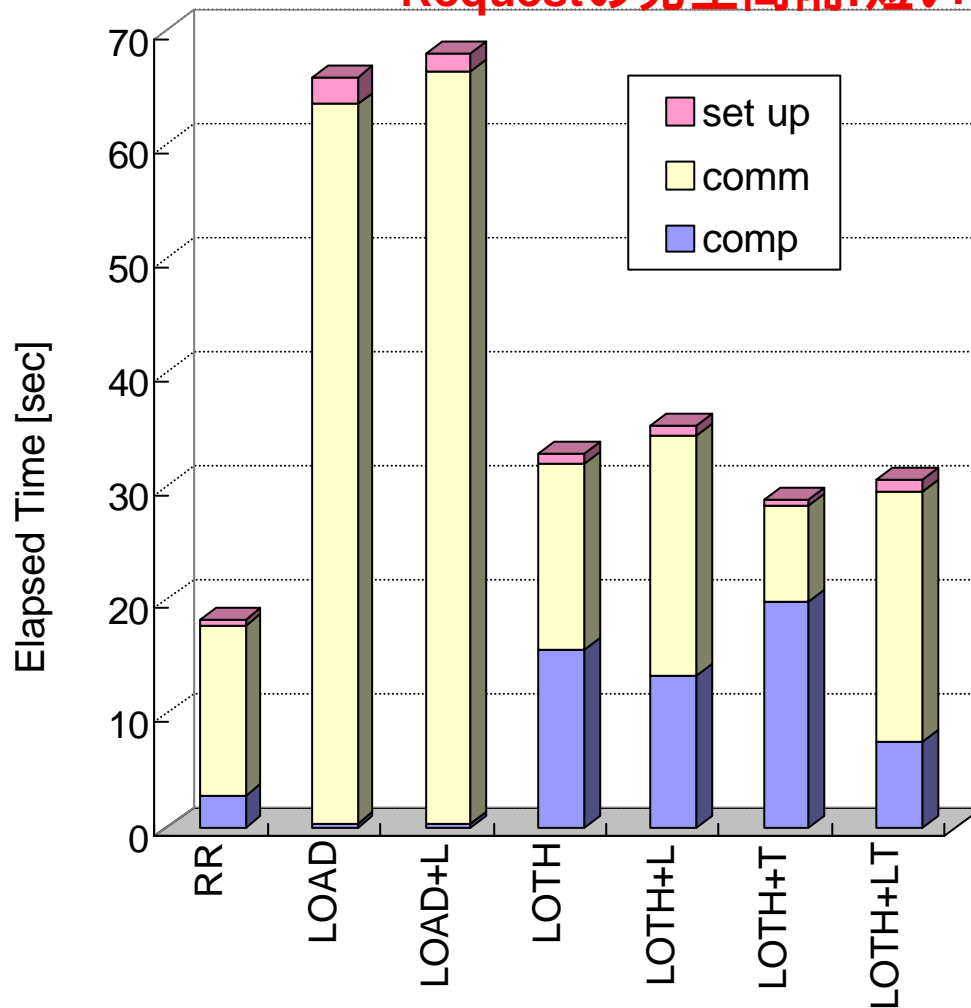
- **CaseA** : [Requestの所用時間] + *interval*
- **CaseB** : [Requestの所用時間] × 2 + *interval*

Requestが頻繁に発生



実測による評価結果 (CaseA)

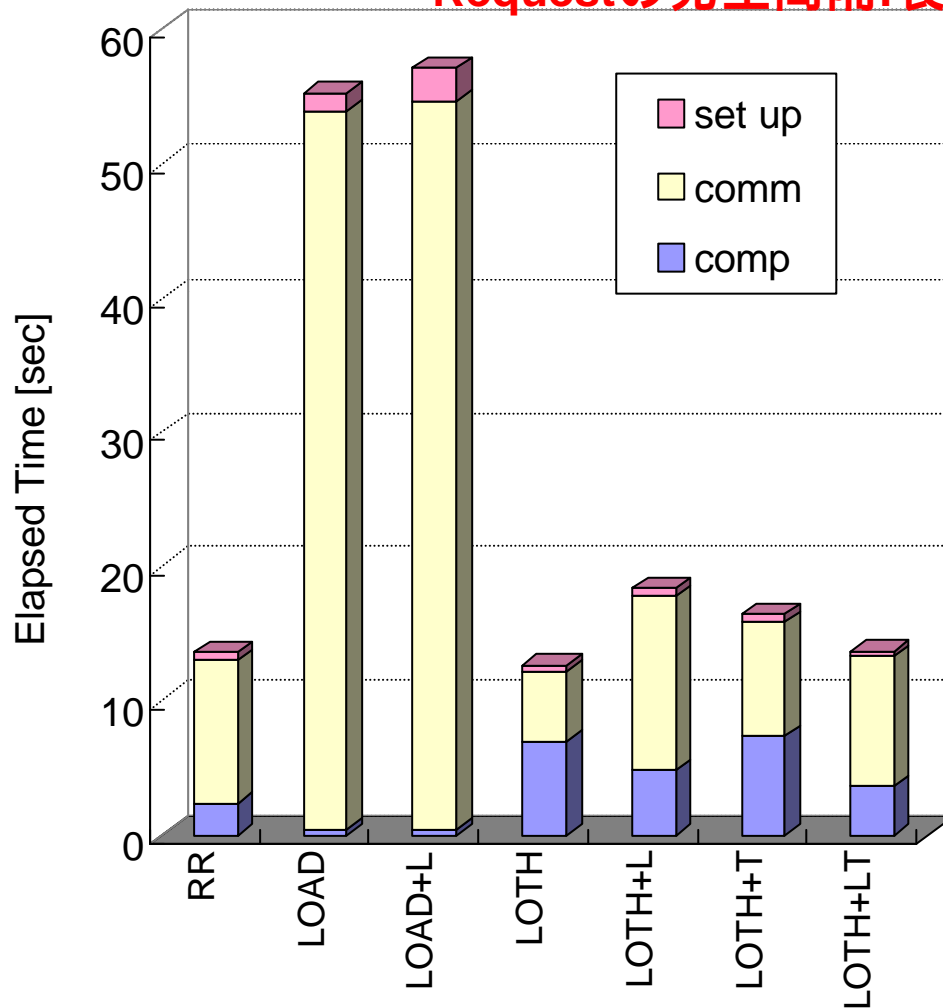
Requestの発生間隔:短い



- RRが1番よい性能
→ LOTHではリソースの干渉
- Loadの性能は悪い
→ すべてのRequestがJ90に投げられた

実測による評価結果 (CaseB)

Requestの発生間隔:長い



- LOADが悪い
- LOTHがよい性能
 - サーバ/ネットワークの負荷をバランスよく考慮
- LOTH+L, LOTH+Tの性能がよくない
 - 予測が単純

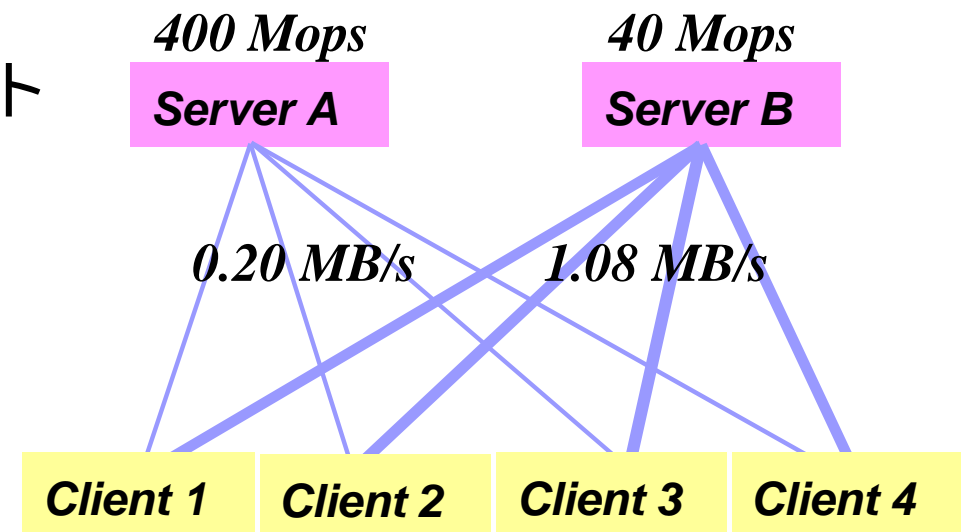
シミュレーションによる評価

■ 性能評価モデル (JSPP '98)

- サーバ, ネットワークを待ち行列で表す
- 様々なシステム構成, 振舞いが表現可能

■ 評価環境

- 実測とほぼ同様の設定
- クライアントは複数サイトに分散
- 計算ルーチン:
Linpack
(n = 600, 固定)



シミュレーションの設定パラメータ

■ クライアント

- Request の発行間隔 : 指数分布
 - **CaseA** : [Requestの所用時間] + *interval*
 - **CaseB** : [Requestの所用時間] × 2 + *interval*
- 論理パケットサイズ = 0.1 [MB] (固定)

Requestが頻繁に発生



■ ネットワーク (FCFS)

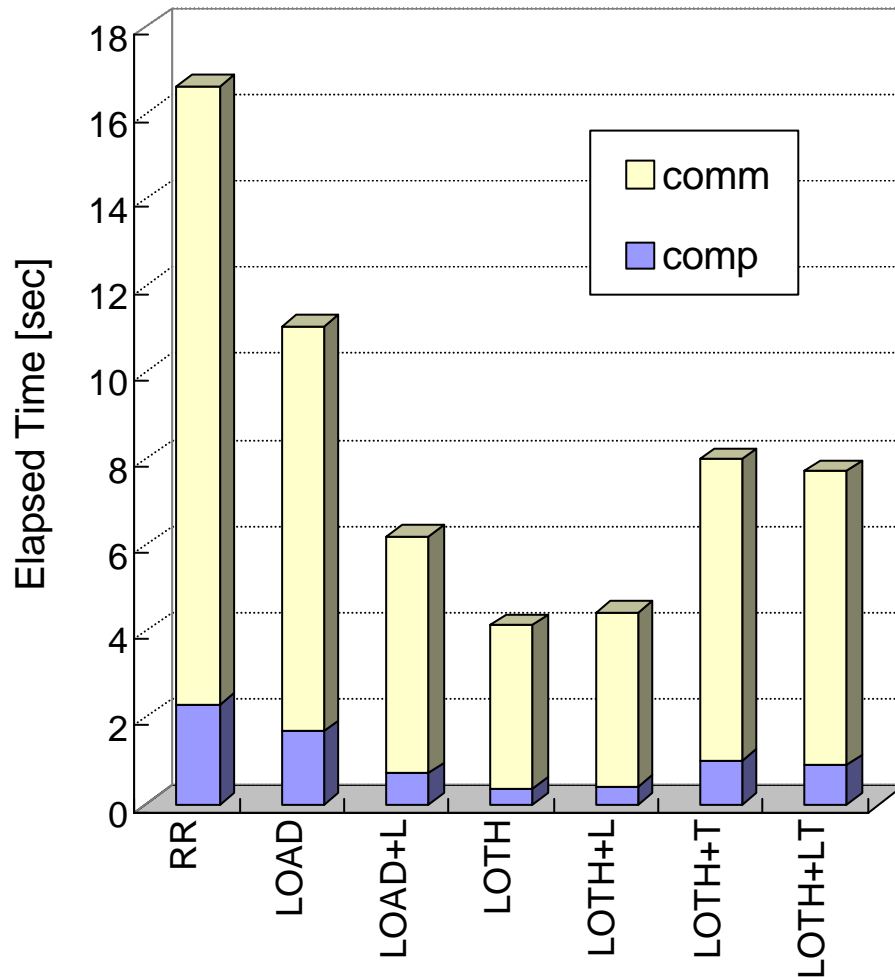
- バンド幅 $T_{net} = 1.5$ [MB/s]
- 外乱のデータ : 平均サイズ = 0.1 [MB] , ポアソン到着

■ サーバ (FCFS)

- 外乱のジョブ : 平均演算数 = 10 [Mflops]
稼働率 10 [%] , ポアソン到着

シミュレーションによる評価結果 (CaseA)

Requestの発生間隔:短い



■ LOTHで適切にスケジューリング

→リソース情報を適切に考慮

■ 予測したものの性能はあまり良くない

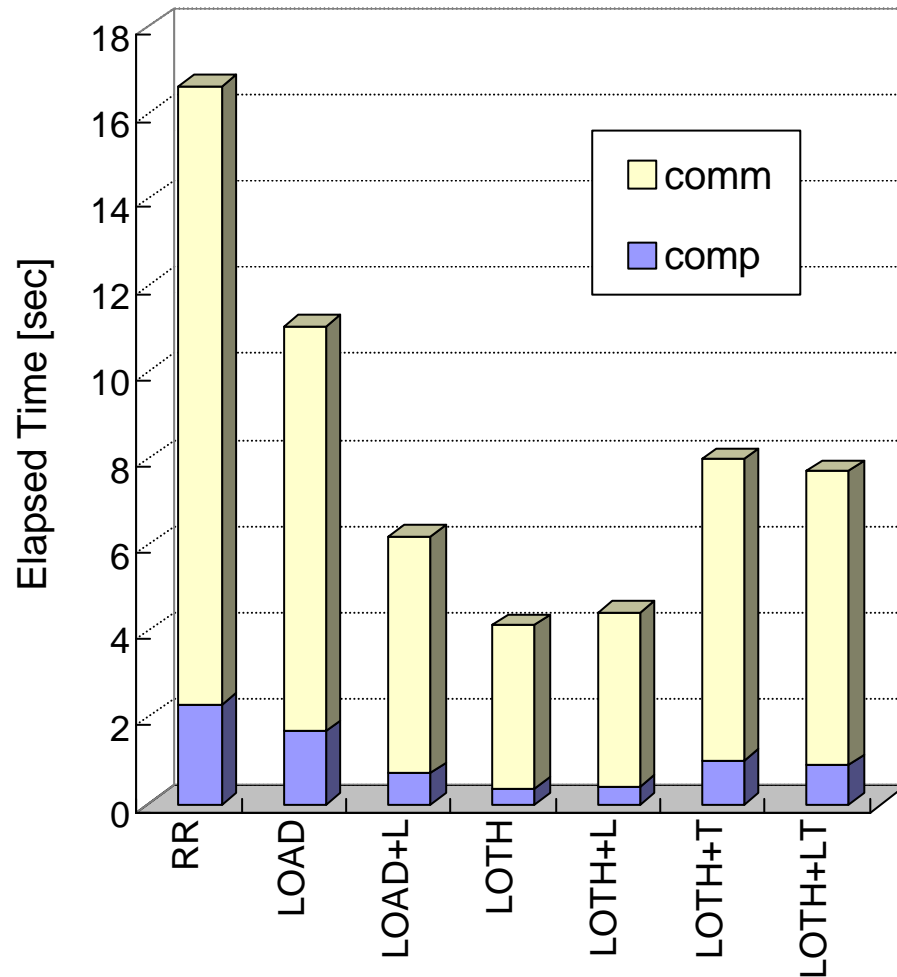
→性能予測が不十分

■ RRの性能が悪い

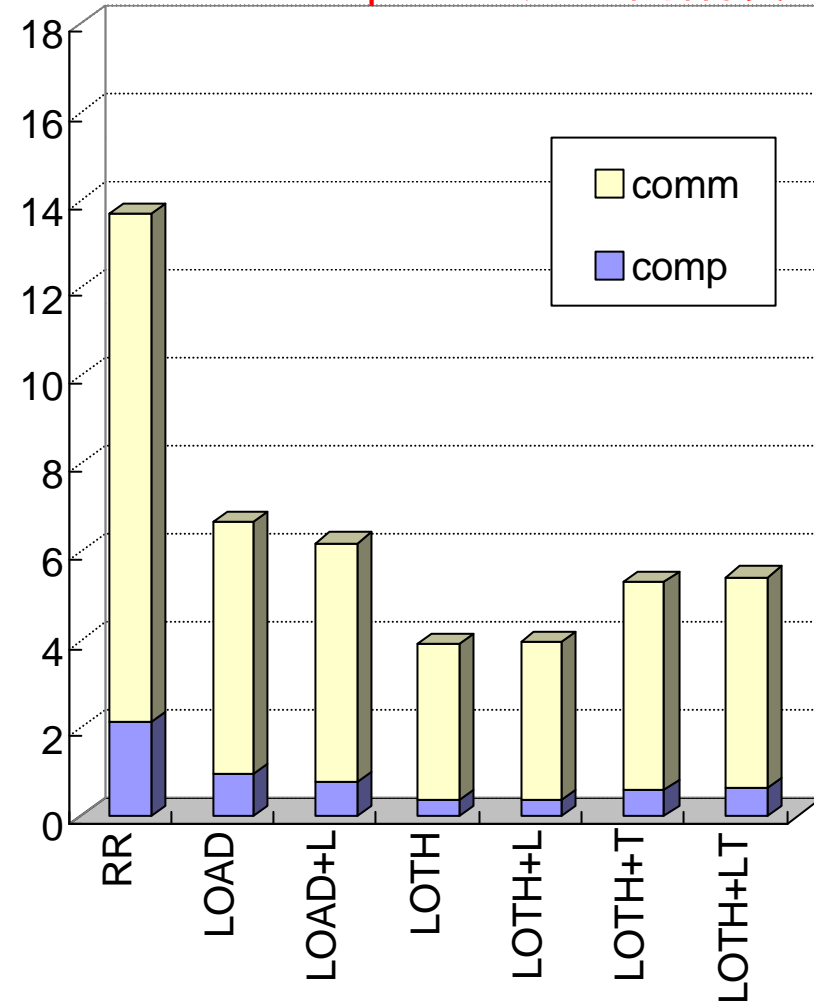
→ネットワークの干渉がないため、相対的に悪い

シミュレーションによる評価結果 (CaseAとCaseBの比較)

CaseA : Requestの発生間隔短い



CaseB : Requestの発生間隔長い



実測・シミュレーションによる スケジューリング手法の評価結果

■ 実測

- Request 発生間隔が短い場合, RRが1番よい性能
- Request 発生間隔が長い場合, LOTHがよい性能
- 予測を考慮したスケジューリングの性能はあまり良くない
- Loadの性能は悪い

■ シミュレーション

- LOTHで適切にスケジューリング
- 予測したものの性能はあまり良くない → 予測が不十分
- RRの性能が悪い

Ⓜ リソースの状況を適切に予測・考慮することが重要

まとめ

- ジョブ/アプリケーションスケジューリングを統合するフレームワークを提案
- Ninf MetaServer アーキテクチャの実装
- スケジューリング手法の評価
 - サーバの性能・負荷, 通信スループットを適切に考慮したスケジューリングが有効
 - リソースの干渉が性能低下に大きく影響
 - 適切なリソース予測が重要

今後の課題1

- Network Taskスケジューリングの評価
 - シミュレータによる大規模評価
 - シミュレーションの信頼性の向上 - ネットワーク
 - 実際のアプリケーションのWork Load を考慮
- Job / Application Resource Allocatorの実現
 - 既存のグローバルスケジューリングシステムとの比較
 - 最適なスケジューリング手法の提案

今後の課題2

■ 計算サーバの性能指標

- アプリケーションや問題サイズにより、マシンの性能が異なる

Ⓜ 問題の性質, サイズ, 通信パターンなどを考慮

■ Requestの混雑度とスケジューリング

- リソースを共有するRequestが頻繁に発生すると、LOTHで適切なスケジューリングが行えない

Ⓜ 適切なリソースの予測