

広域計算システムの シミュレーションによる評価

— Ninf システムの広域分散環境での
ジョブスケジューリング実現に向けて —

竹房あつ子，合田憲人，小川宏高，中田秀基，
松岡聰，佐藤三久，関口智嗣，長嶋雲兵

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





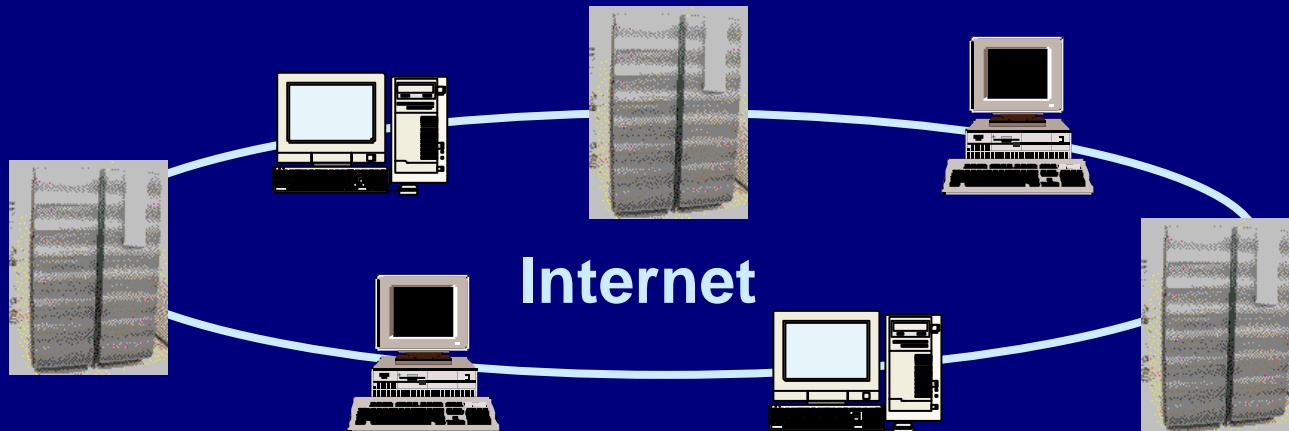
高性能広域計算

n ネットワーク技術の発展

計算・情報資源を利用した超広域並列分散計算

→ 高性能広域計算 (Global Computing)

Ninf, Globus, NetSolve, Legion etc.



Global Computing Infrastructure

高性能広域計算のためのスケジューリング



n 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

- 計算サーバの性能 / 負荷
- ネットワークのトポロジ / バンド幅 / 混雑度

→ 高性能広域システムのためのスケジューリング

n 既存のスケジューリングシステム

MetaServer (Ninf), Agent (NetSolve), AppLeS, Prophet

➡ 有効性を示す評価の枠組みがない

- 様々な実行環境を想定した大規模評価実験が困難
- 再現性のある公平な評価実験が不可能



研究の目的と発表内容

- 高性能広域計算のシミュレーションモデルの設計・シミュレーション環境の構築
 - 様々な環境設定、大規模評価実験
 - 再現性がある評価実験
- 発表内容
 - 高性能広域計算の概要
 - 高性能広域計算システムシミュレータの概要
 - シミュレーションモデルの有効性の評価
 - スケジューリング手法の評価

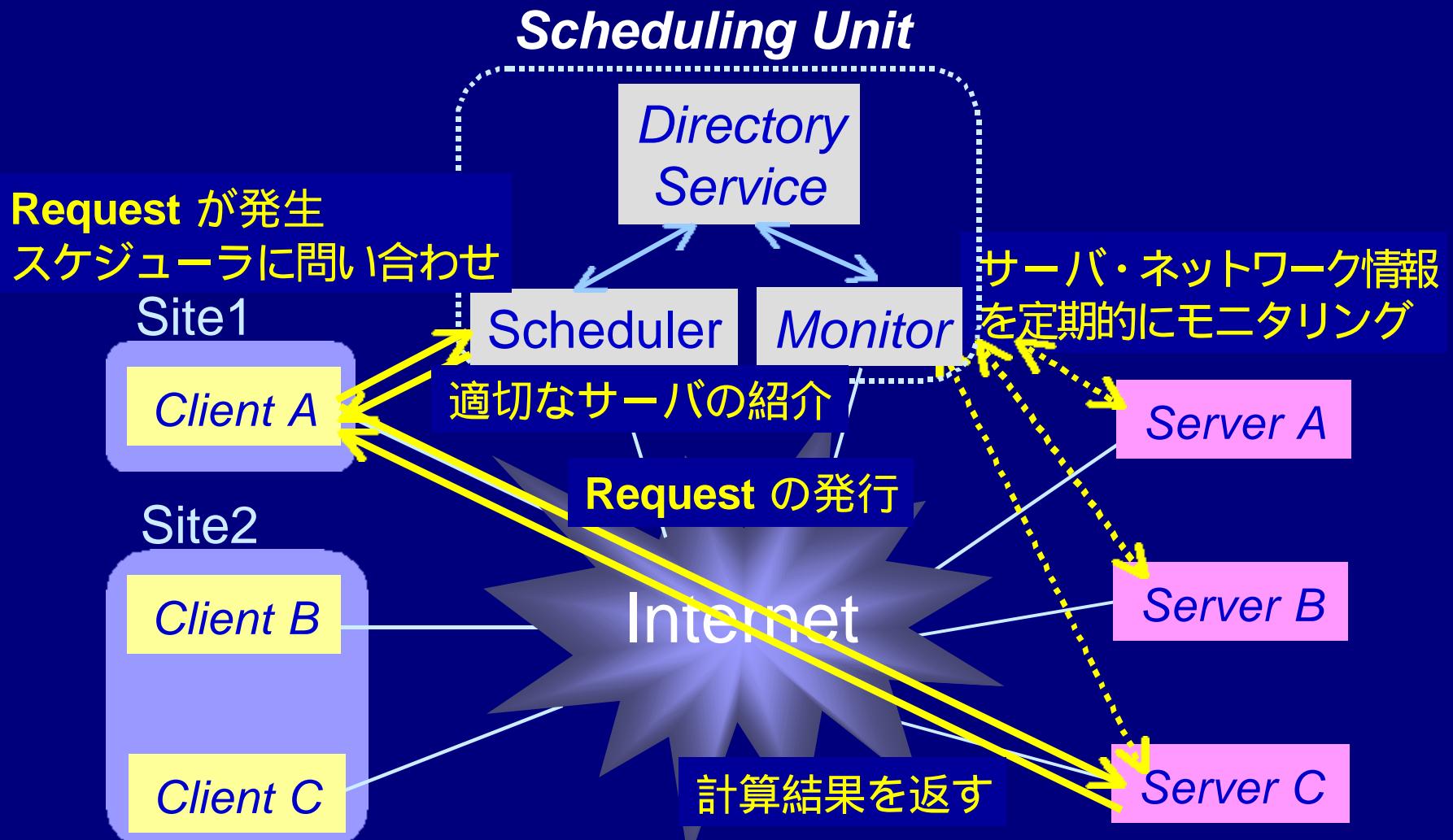


高性能広域計算システムの概要

- クライアント
- 計算サーバ
- スケジューリングユニット
 - スケジューラ (ex. AppLeS, Prophet)
 - 各システムのポリシーに従ったスケジューリングを行う
 - ディレクトリサービス (ex. Globus-MDS)
 - 資源情報の集中型データベース
 - モニタ (ex. NWS)
 - 計算サーバ / ネットワークのモニタリングと予測



高性能広域計算の実行プロセス

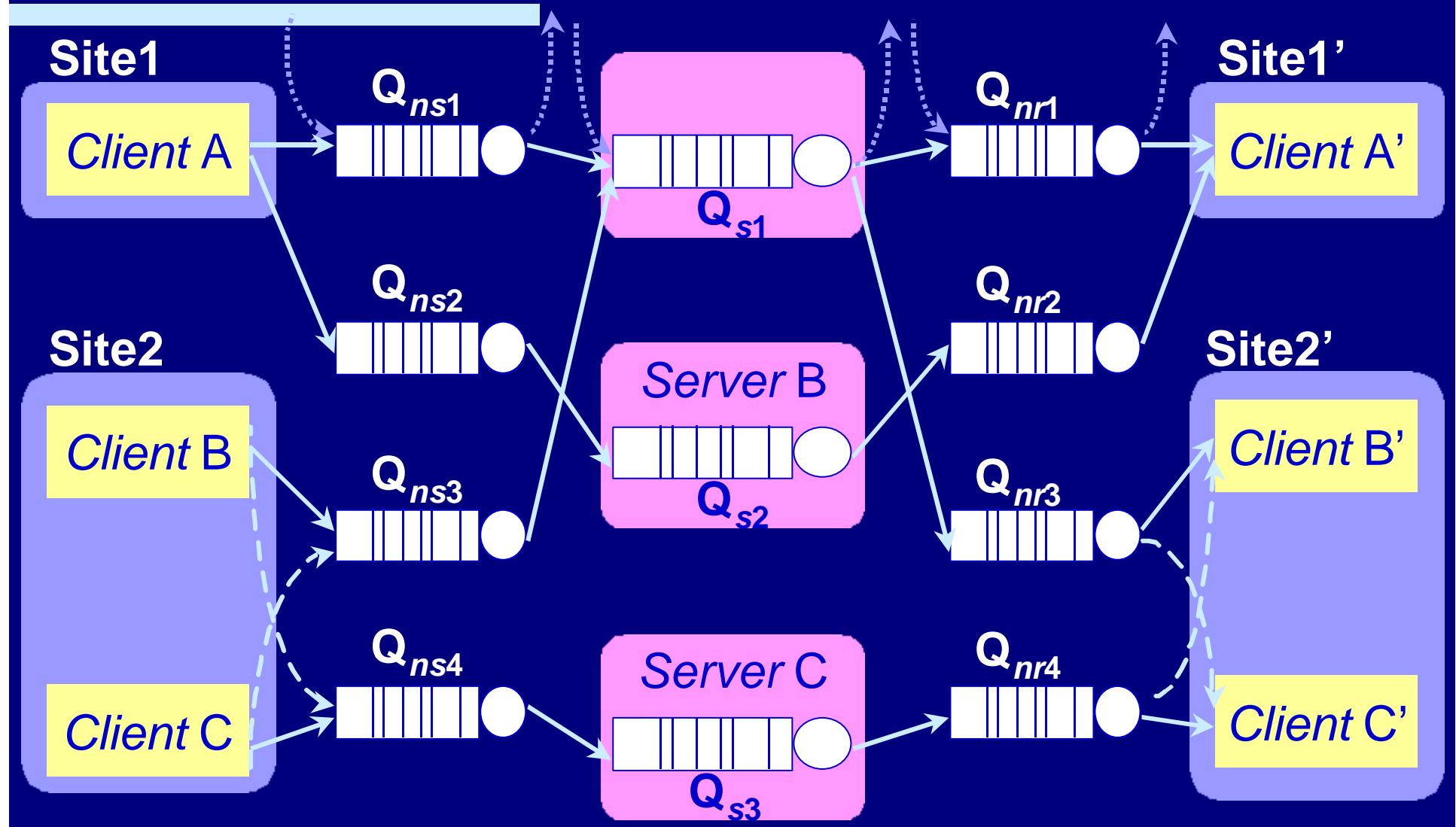




シミュレーションモデル

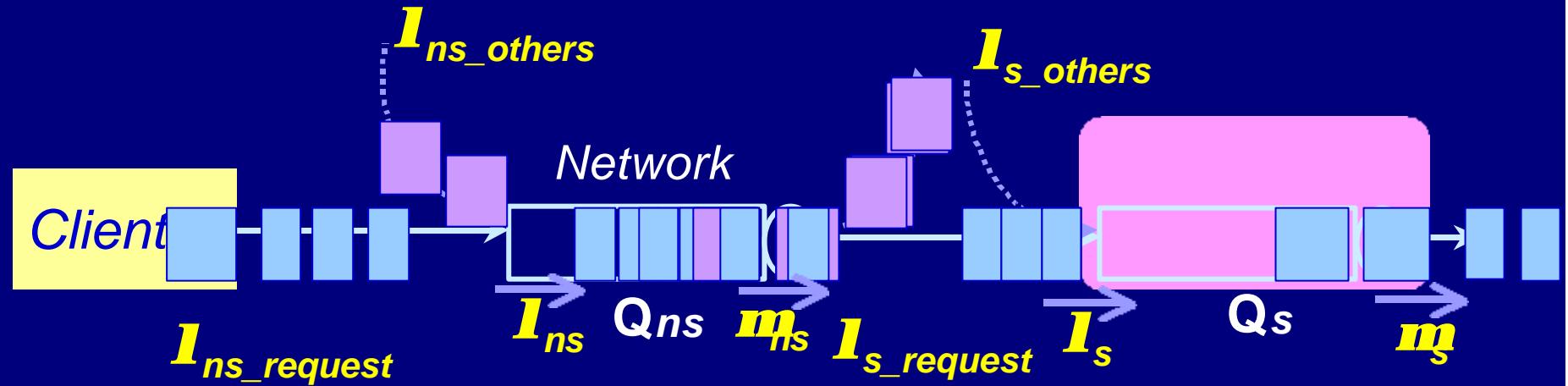
- シミュレーションモデルの条件
 - 多様なクライアント, サーバのネットワークトポロジ
 - サーバ: 性能, 負荷, 变動
 - ネットワーク: バンド幅, スループット(混雑度), 变動が表現可能であること
- ⑧ 待ち行列モデルを採用
- 本シミュレーションモデルの特徴
 - 様々な実行環境を想定した大規模評価実験
 - 再現性のある公平な評価が可能となる

待ち行列による高性能 広域計算シミュレーションモデル





待ち行列モデル



■ Q_{ns} へのデータの到着率: $I_{ns} = I_{ns_request} + I_{ns_others}$

$I_{ns_request}$: Requestのパケット, I_{ns_others} : 外乱のデータ

■ Q_s へのジョブの到着率: $I_s = I_{s_request} + I_{s_others}$

$I_{s_request}$: Requestのジョブ, I_{s_others} : 外乱のジョブ



クライアントでの処理

- $I_{request}$ の確率でRequest を発行
- スケジューラにサーバを問い合わせる
 - Request に関する情報を提供
 - 演算数 , 転送データ量(往路 / 復路)
- ↓ スケジューラは適切なサーバを割り当てる
- Request を割り当てられたサーバに発行
 - 転送データを論理パケットサイズに分割
- ↓ サーバで処理が終了
- Q_{nr} からデータを受け取る



クライアントに関するパラメータ

- Q_{ns} に論理パケットを投入する確率

$$I_{packet} = T_{net} / W_{packet}$$

T_{net} : ネットワークのバンド幅

W_{packet} : 論理パケットサイズ

- パラメータの設定例

$T_{net}=1.0[\text{MB}/\text{s}]$, $W_{packet}=0.01[\text{MB}]$ である場合

$$I_{packet} = 1.0/0.01 = \underline{\underline{100}}$$



ネットワークでの処理

- n I_{ns_others} で通信スループットを表現
→ Q_{ns} に M/M/1/N待ち行列を採用
- n Q_{ns} には論理パケットと外乱のデータが到着
 - バッファが一杯のとき、各データは再送される
- n 各データは Q_{ns} のサーバ上で [サイズ/バンド幅] 時間処理され、 Q_{ns} から出る



ネットワークに関するパラメータ

n 外乱データの到着率 - スループットを決定

$$\begin{aligned} \mathbf{I}_{packet} / (\mathbf{I}_{ns_others} + \mathbf{I}_{packet}) &= T_{act} / T_{net} \\ \Leftrightarrow \mathbf{I}_{ns_others} &= (T_{net} / T_{act} - 1) \times \mathbf{I}_{packet} \end{aligned}$$

n Q_{ns} のバッファ長 - レイテンシを決定

$$\begin{aligned} W_{packet} \times N / T_{net} &\approx T_{latency} \\ \Leftrightarrow N &\approx T_{latency} \times T_{net} / W_{packet} \\ \text{ただし } N &\geq 2 \end{aligned}$$



ネットワークに関する パラメータの設定例

- $T_{net}=1.0[\text{MB}/\text{s}]$, $W_{packet}=0.01[\text{MB}]$ ($I_{packet}=100$)の条件下, 通信スループット $T_{act}=0.1[\text{MB}/\text{s}]$, レイテンシ $T_{latency}=0.1$ を再現する場合

- 外乱のデータの到着率

$$\begin{aligned}I_{ns_others} &= (T_{net}/T_{act}-1) \times I_{packet} \\&= (1.0/0.1-1) \times 100 = \underline{\underline{900}}\end{aligned}$$

- バッファ長

$$N \approx T_{latency} \times T_{net} / W_{packet} = 0.1 \times 1.0 / 0.01 = \underline{\underline{10}}$$



サーバでの処理

- 処理の応答時間を表す
→ Q_s に M/M/1待ち行列を採用
- Requestの受付
 - すべての論理パケットが Q_{ns} から出た時点で Q_s に投入される
- Q_s のサーバ上で各ジョブは [演算数 / サーバの性能] 時間処理
- Request の処理結果をクライアントに送信
 - 転送データを論理パケットサイズに分割
 - I_{packet} の確率で Q_{nr} に論理パケットを投入



サーバに関するパラメータ

- n 外乱のジョブの到着率 - サーバの稼働率を決定

$$I_{s_others} = T_{ser} / W_{s_others} \times U$$

T_{ser} : サーバの性能

W_{s_others} : 外乱のジョブの平均演算数

U : サーバの稼働率

- n パラメータの設定例

- $T_{ser}=100[\text{Mflops}]$, $W_{s_others}=0.01 [\text{MB}]$ のとき , $U=0.1$ を再現する場合

$$I_{s_others} = 100/0.01 \times 0.1 = \underline{\underline{1000}}$$



シミュレータの特徴

- n シミュレーション設定が pluggable
 - クライアント , サーバ , ネットワークの構成
 - スケジューリングモデル
 - ネットワーク / サーバでの処理方法
 - 乱数分布
(Abstract Factory パターンを用いた)
- n 各オブジェクトに対して独立の乱数系列が指定可能
→ 柔軟なシミュレーション設定が可能
- n Java で実装

シミュレーションモデルの 有効性の評価



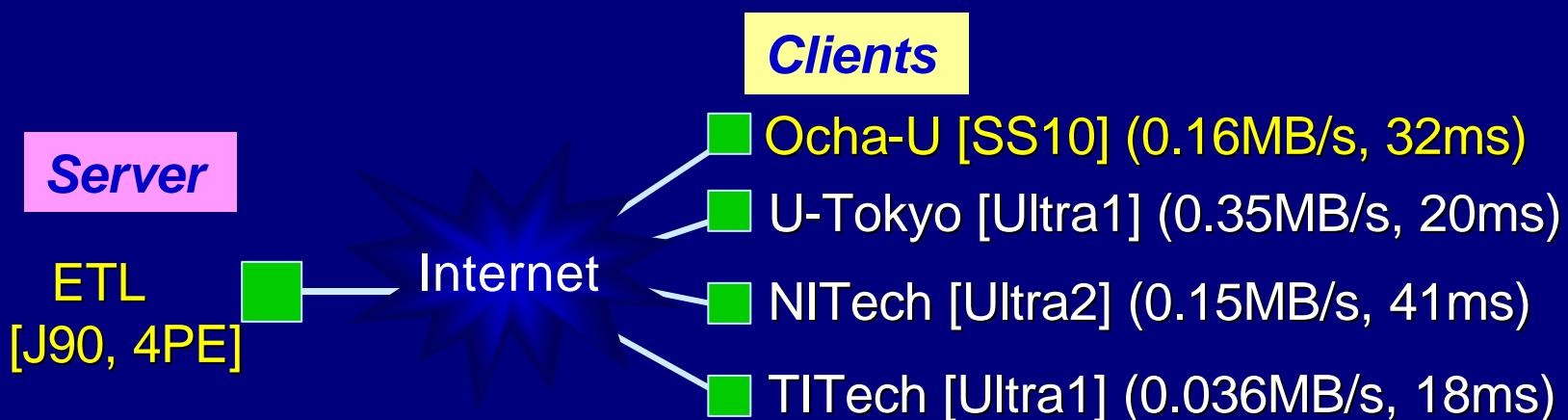
■ 実測(Ninfシステム)とシミュレーションの比較

- Linpack

演算数: $2/3n^3 + 2n^2$ [flops], 通信量: $8n^2 + 20n$ [bytes]

■ 評価環境

- サーバ: クライアント = 1 : 1, 1 : 4





モデルの評価での設定パラメータ

■ クライアント

- $I_{request} = 1 / [\text{Requestの所要時間} + interval]$
- $W_{packet} = 10, 50, 100 \text{ [KB]}$ (固定)

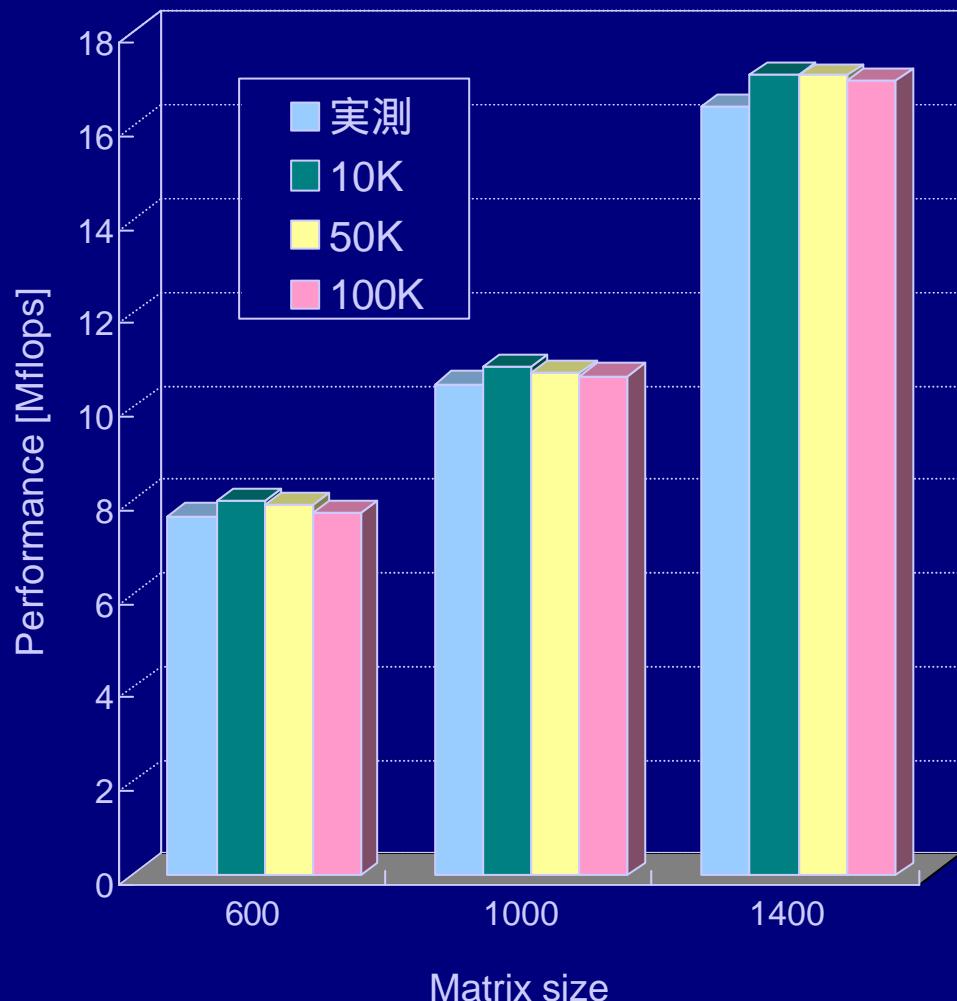
■ ネットワーク (FCFS)

- バンド幅 $T_{net} = 1.5 \text{ [MB/s]}$
- 外乱のデータ : 平均サイズ = W_{packet} (指数分布)
 $I_{ns_others}, I_{nr_others}$ はポアソン到着

■ サーバ (FCFS) – 実測から得られたパラメータを適用

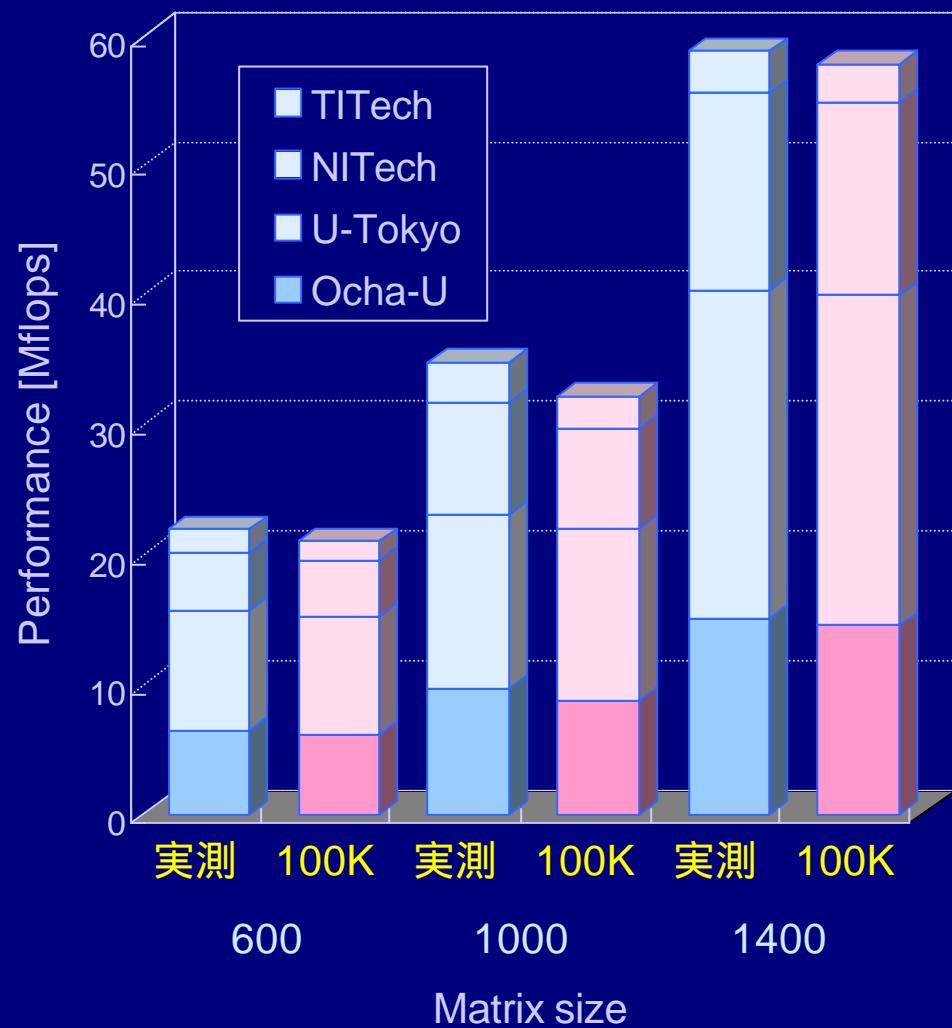
- 性能 $T_{ser} = 500 \text{ [Mflops]}$
- 外乱のジョブ: 平均演算数 = 10 [Mflops] (指数分布)
稼働率 4 [%] , ポアソン到着

シミュレーションモデルの 有効性の評価結果(1 : 1)



- 論理パケットサイズが異なる場合も実測と同程度の性能
- ④ **シミュレーションコストが削減可能**
- 問題サイズが変化しても実測と同程度

シミュレーションモデルの 有効性の評価結果(1 : 4)



- 異なるスループットを設定した場合も実測と同程度の性能
- ⑧ シミュレーションで実測同様の挙動を再現

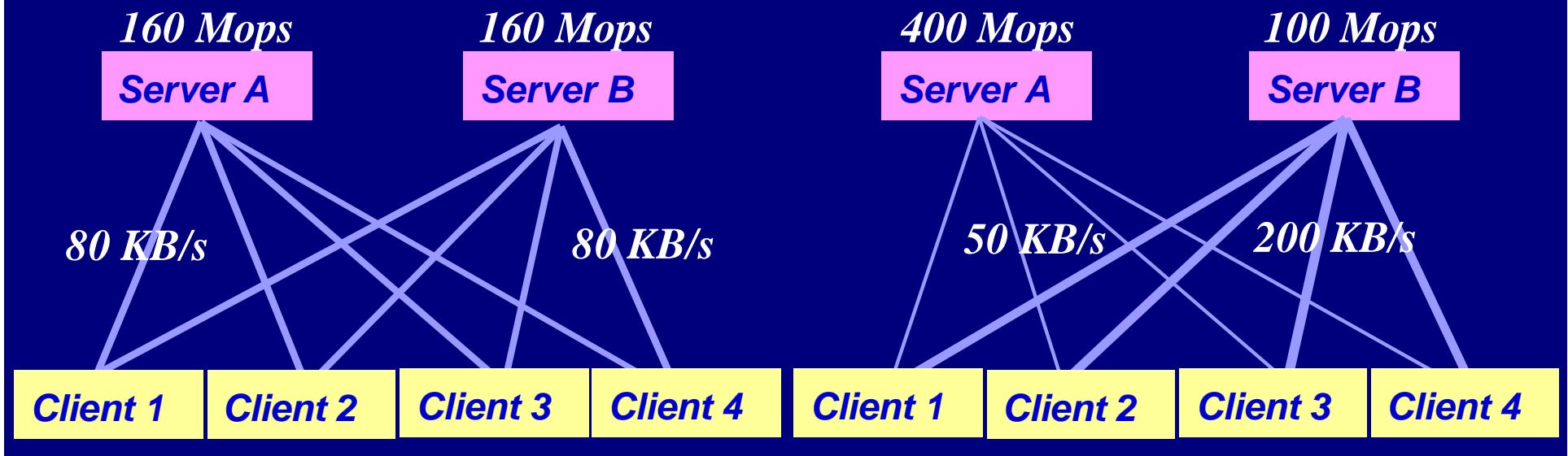


シミュレーションによる スケジューリング手法の評価

■ 3種類の基本的なスケジューリング手法の評価

- LRR : 各クライアントでRound-Robin
- GRR : 広域スケジューラでRound-Robin
- LOAD : 広域スケジューラで負荷 + 計算性能

■ 均質・不均質な環境で Linpack / EP による評価





スケジューリング手法の評価の設定パラメータ

n クライアント

- 問題サイズ : Linpack - 600 , EP - 2^{21}
- $I_{request} = 1 / (\text{最悪のRequest所要時間} + interval)$
(interval : Linpack 5 [sec], EP 20 [sec]), ポアソン到着
- 論理パケット $W_{packet} = 100$ [KB] (固定), ポアソン到着

n ネットワーク (FCFS)

- $T_{net} = 1.5$ [MB/s]
- 外乱のデータ : 平均サイズ = W_{packet} (指数分布)
 $I_{ns_others}, I_{nr_others}$ はポアソン到着

n サーバ (FCFS)

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



均質な環境での スケジューリング手法の評価結果

問題	スケジューラ	通信	計算	計	サーバの稼働率[%]	
		[sec]	[sec]	[sec]	A(160Mflops)	B(160Mflops)
Linpack	LRR	76.503	0.011	76.514	10.020	9.924
	GRR	76.910	0.011	76.921	9.992	9.941
	LOAD	75.407	0.012	75.419	10.000	9.934
EP	LRR	2.582	83.036	85.618	87.395	92.549
	GRR	2.591	76.161	78.752	89.988	91.119
	LOAD	2.482	89.613	92.095	88.245	88.833

- Linpackではスケジューラの差がない
- EPではGRR , LRR , LOADの順に良い性能
 - 均質な環境であるため , GRRで適切に負荷分散できた



不均質な環境での スケジューリング手法の評価結果

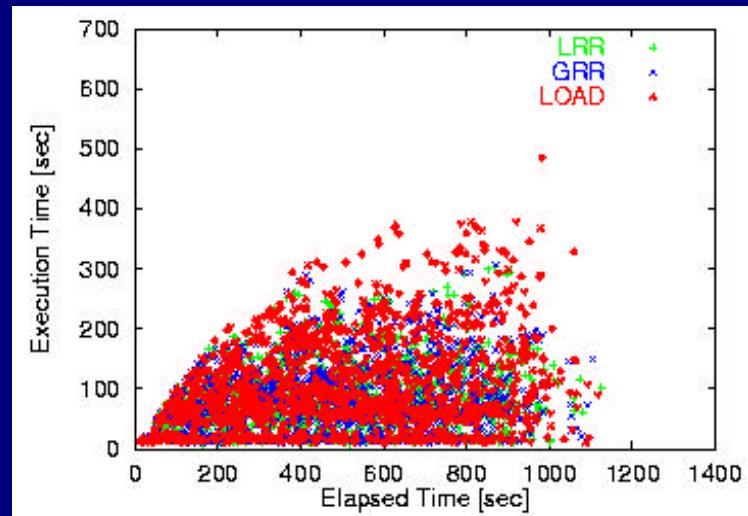
問題	スケジューラ	通信	計算	計	サーバの稼働率[%]	
		[sec]	[sec]	[sec]	A(400Mflops)	B(100Mflops)
Linpack	LRR	80.310	0.012	80.322	10.029	9.955
	GRR	80.442	0.012	80.454	10.031	9.960
	LOAD	108.661	0.014	108.675	10.031	9.959
EP	LRR	2.289	108.005	110.294	45.172	98.128
	GRR	2.290	97.522	99.812	46.014	97.741
	LOAD	1.182	32.685	34.497	63.671	79.117

- Linpack ではLOADの性能が悪い
 - 通信主体の問題なので、ネットワークがボトルネックとなつてサーバの負荷が上がらない [SC97]
- EP では LOAD で効率の良いスケジューリング

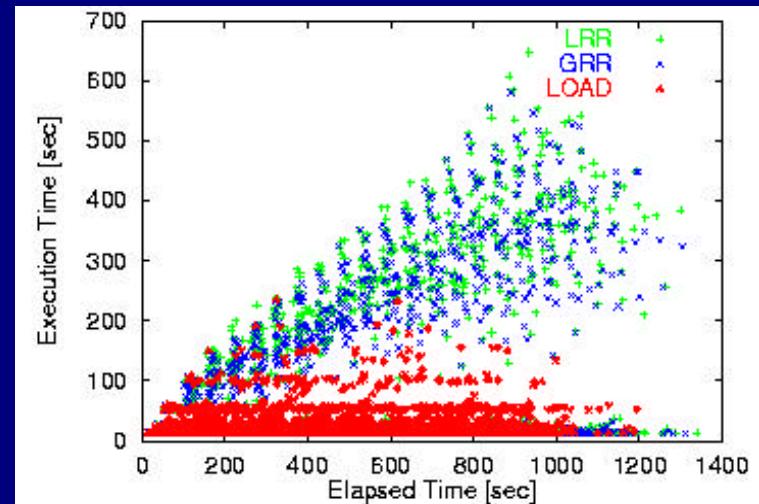
不均質な環境での スケジューリング手法の評価結果



Linpack



EP



- Linpack ではLOADの性能が悪い
- EP では LOAD で効率の良いスケジューリング
 - LRR , GRR はRequestの所要時間が長くなっていく



まとめ

- 高性能広域計算システムのシミュレーションモデルの設計・シミュレーション環境の構築を行った
- 実測値とほぼ等しい結果が得られ、シミュレーションモデルが有効であることを確認した
- 基本的なスケジューリング手法の評価では、実測における考察を反映した結果が得られた



今後の課題

- シミュレーションモデルの有効性の向上
 - 実際のネットワークにおける変動を考慮
 - 計算サーバでのジョブの処理方式の多様化
 - Round-Robin など
 - シミュレーションコストの削減
- 高性能広域計算システムにおける他のスケジューリング手法の評価
- より適切なスケジューリング手法の提案