

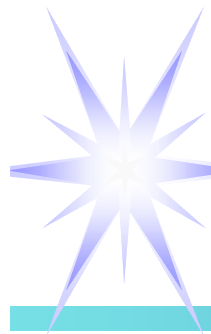
マルチクライアントによるネットワーク 数値情報システム Ninf の性能



竹房 あつ子^{*1}・小川 宏高^{*2}・松岡 聡^{*3}・中田 秀基^{*4}・
佐藤 三久^{*5}・関口 智嗣^{*4}・長嶋 雲兵^{*1}

^{*1} お茶の水女子大学, ^{*2} 東京大学, ^{*3} 東京工業大学,
^{*4} 電子技術総合研究所, ^{*5} 新情報処理開発機構

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



はじめに

Ninf (Network Infrastructure for Global Computing):
広域分散並列計算技術(=Global Computing)を支援するシステム
➤ *C.f., NetSolve, Legion, RCS, Javelin etc.*

Global Computing での問題点

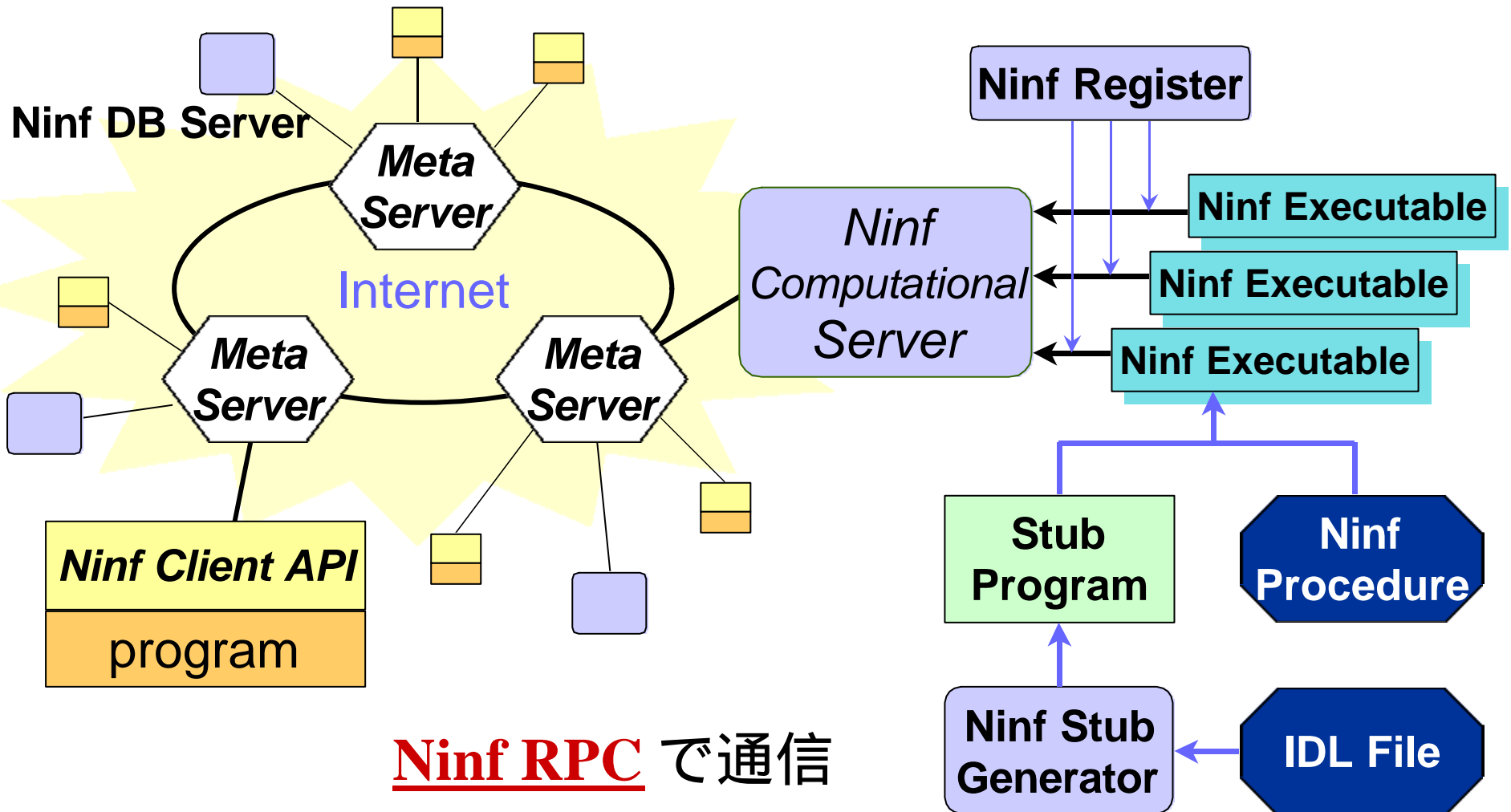
- 通信性能 (Throughput, Latency)
- 負荷分散による計算サーバの適切な選択
- 複数クライアントからの要求に対するサーバの耐久性
- リモートライブラリの設計と再利用性

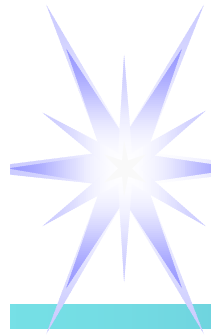


Ninf システム / Global Computing の様々なパラメタの
もとでの性能評価

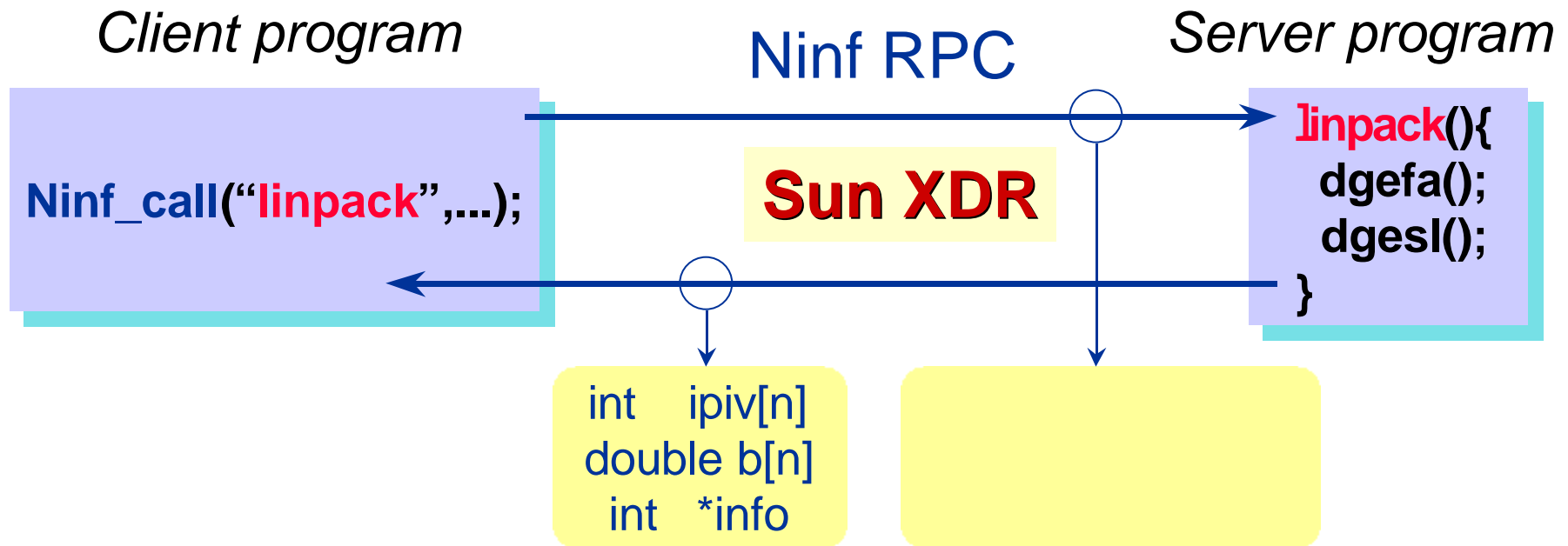


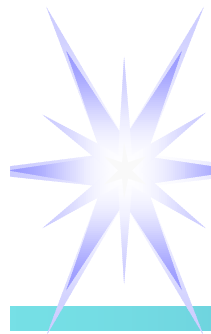
Ninfシステムアーキテクチャ





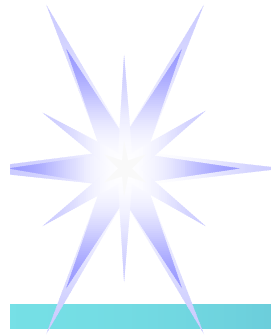
Ninf_call の実行例





評価の概要

- シングルクライアントでの評価：
 - 基本性能の評価 LAN環境, Linpack
- マルチクライアントでの評価：
 - 計算サーバ(J90)の耐久性 (同時に多数のNinf_call)
 - リモートライブラリの再利用性
 - タスクパラレル / データパラレル
 - ネットワークの通信条件による影響
 - LAN / WAN (単一サイト) / WAN (複数サイト)
 - 問題の性質, 計算と通信の比率による影響
 - Linpack / EP



Benchmarks

➤ Linpack :

通信量が多い

➤ 演算量 : $2/3 n^3 + 2 n^2$ [flops]

➤ 通信量 : $8 n^2 + 20 n + O(1)$ [bytes]

➤ Ninf_callの性能 :

$$\frac{(2/3 n^3 + 2 n^2)}{(\text{通信時間} + \text{計算時間})} \text{ [flops]}$$

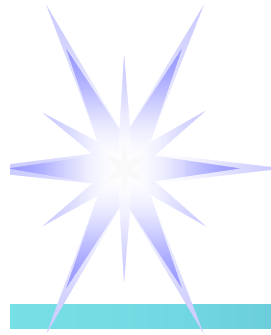
➤ EP (NASPB Kernel) :

通信量が極少 , 計算主体

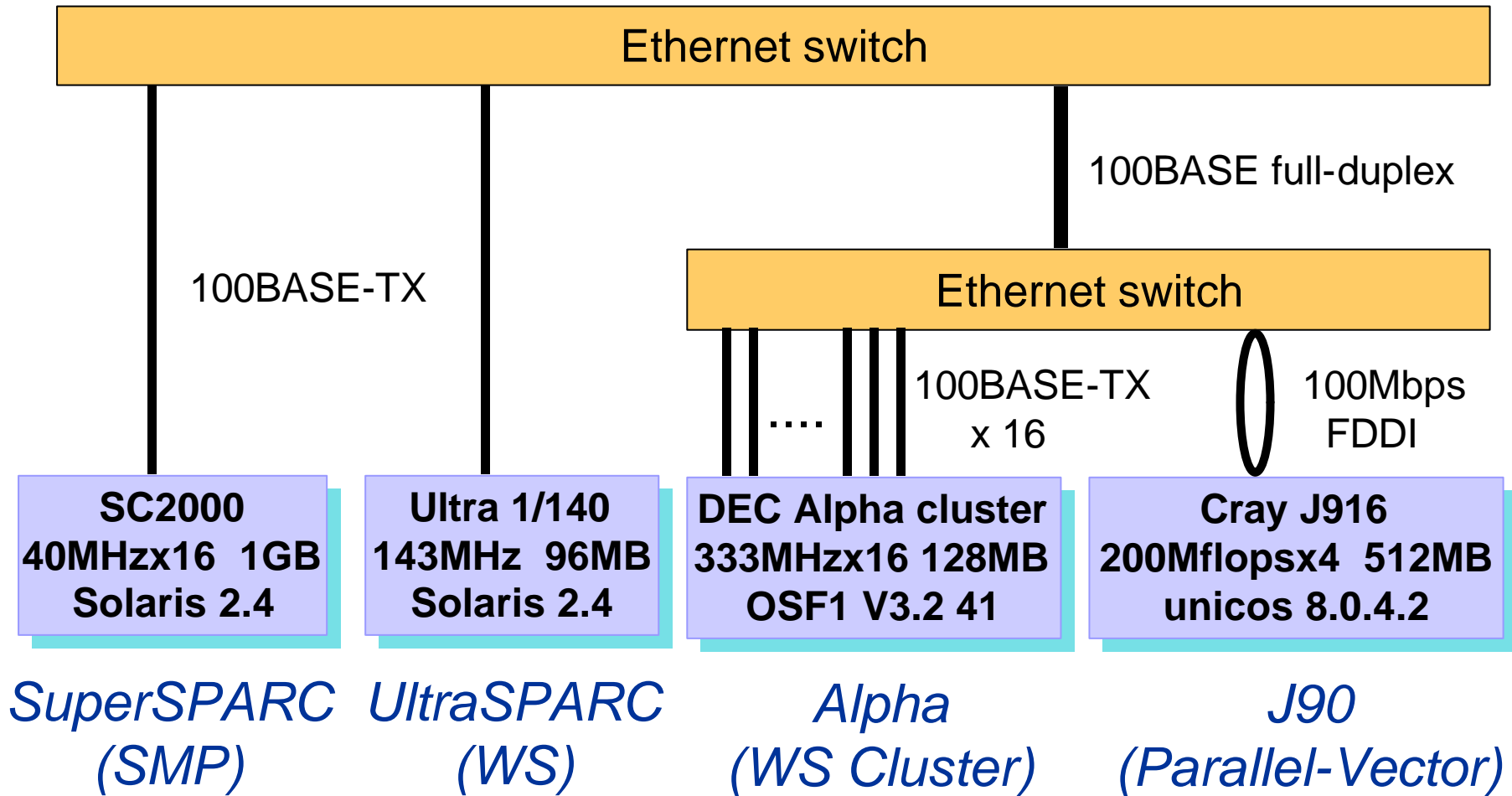
➤ 演算量 : 2^{n+1} [ops]

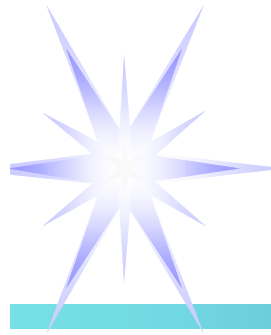
➤ 通信量 : $O(1)$ [bytes]

➤ Ninf_callの性能 : $\frac{2^{n+1}}{(\text{通信時間} + \text{計算時間})} \text{ [ops]}$



LANの計測環境





シングルクライアントによる評価

▶ Linpack Benchmarkのルーチン

J90(4PE) : libSci ライブラリ

(sgetrf, sgetrs) **4PE版lib**

その他 : 村田ライブラリ(glub4, gslv4)

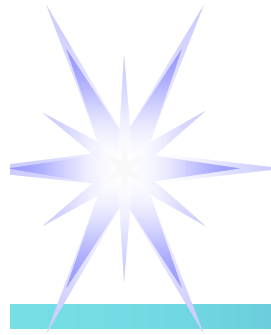
ブロック化・2段2列同時消去(Ninfには不利)

▶ 計測条件

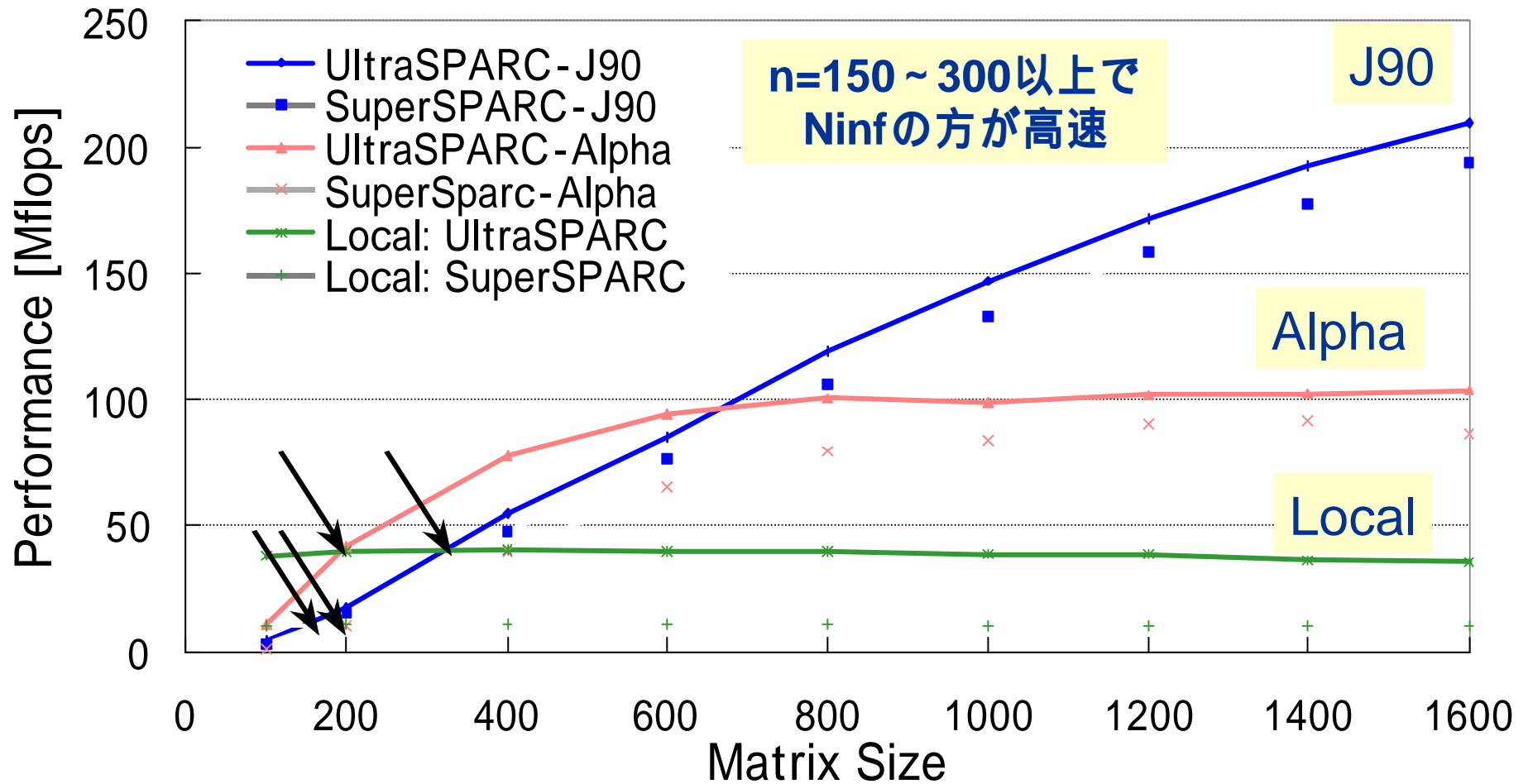
Ninf_call の回数 : 20回

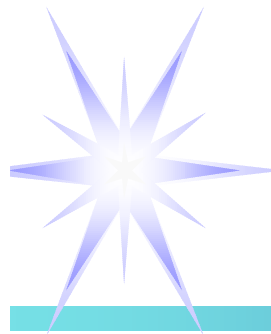
採用した値 : 最高性能値

Client	Local	Remote (Ninf_call)		
		Ultra	Alpha	J90
SuperSPARC				
UltraSPARC		-		
Alpha		-	-	

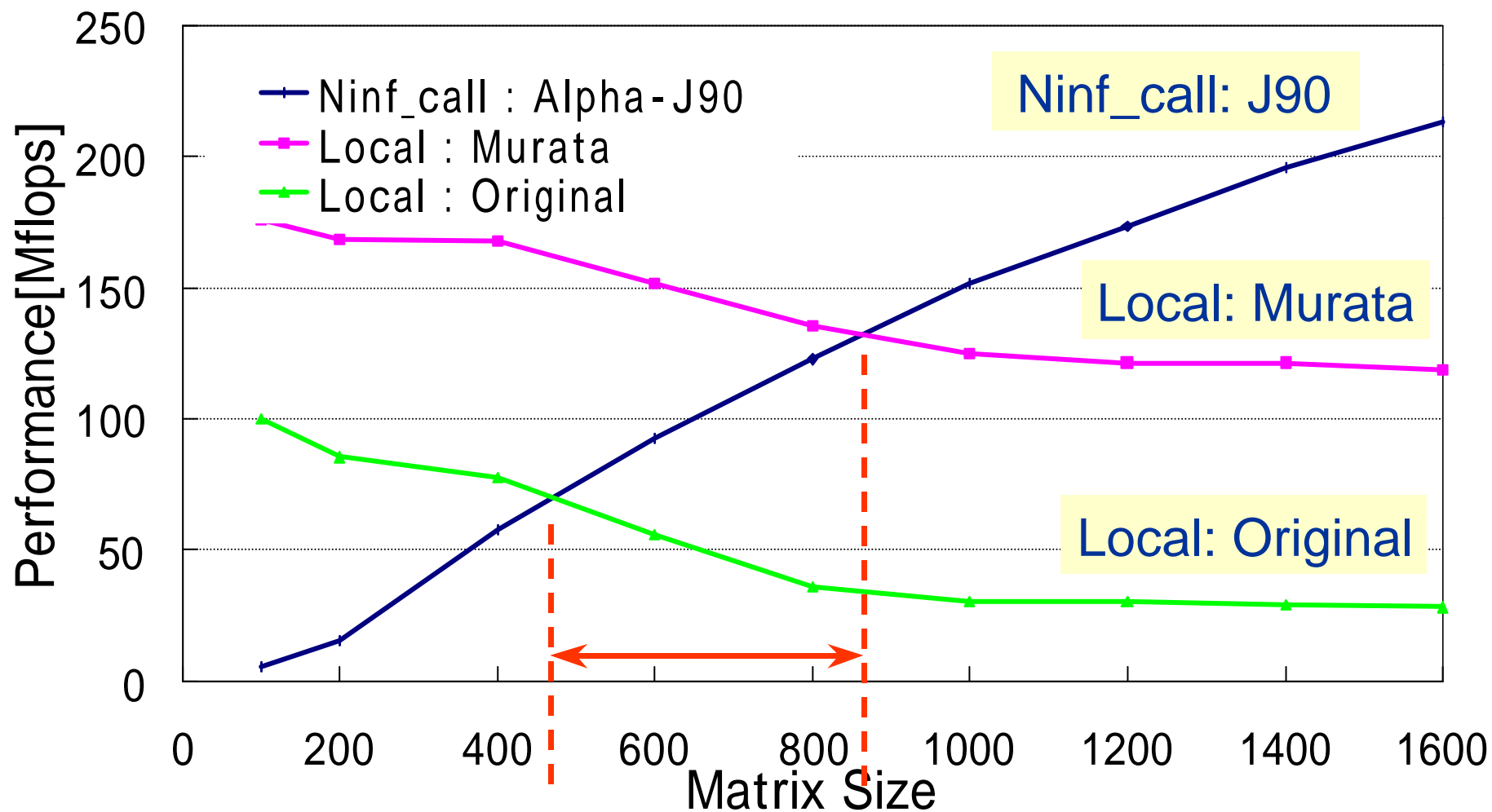


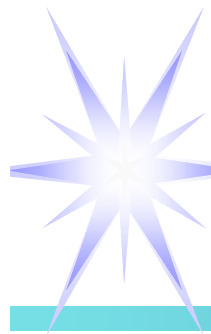
SPARCをクライアントとした性能





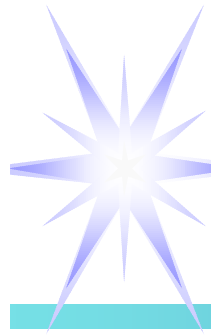
Alpha をクライアントとした性能





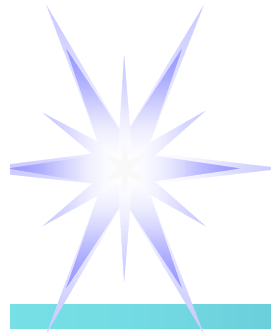
シングルクライアント環境での結果

- ▶ バンド幅が十分な場合(2 ~ 6MB/s)は問題の規模が小さくてもNinf_callの方が高速となる
 - ▶ SPARC をクライアントとした場合
問題サイズ **300** でローカルより Ninf を用いた方が高速
 - ▶ Alpha をクライアントとした場合
村田ライブラリ : 問題サイズ **800** / **Original** : 問題サイズ **500**
でローカルより Ninf を用いた方が高速
- ▶ クライアントマシン性能の影響はほとんどない
- ▶ 異機種間でもNinf_callの効率は著しく低下しない

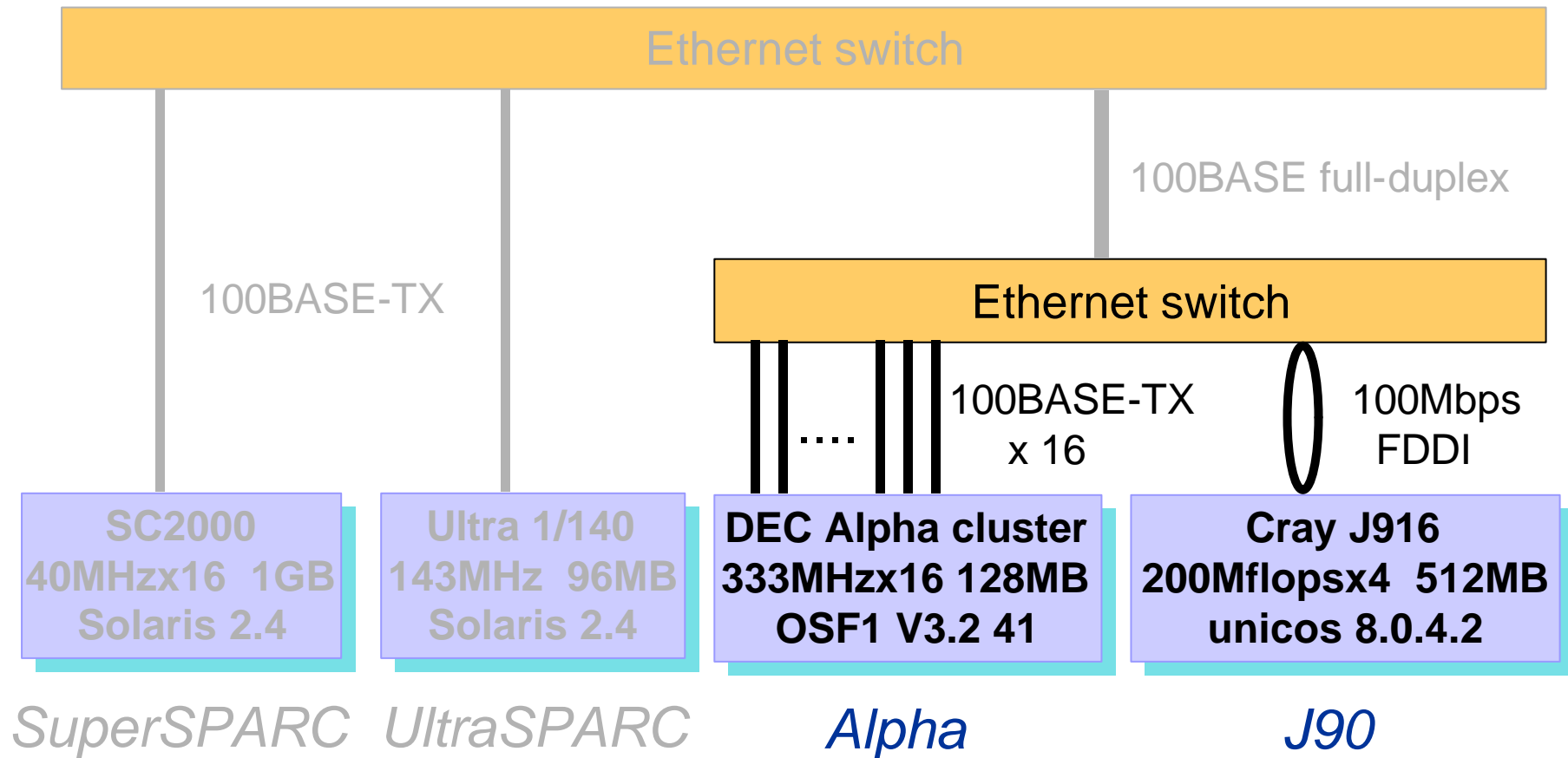


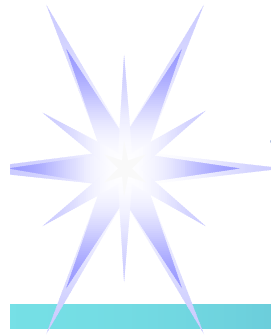
マルチクライアントによる評価

- ▶ 計算サーバの耐久性 (同時に多数のNinf_call)
- ▶ リモートライブラリの再利用性
 - タスクパラレル / データパラレル
- ▶ ネットワークの通信条件による影響
 - LAN / WAN (単一サイト) / WAN (複数サイト)
- ▶ 問題の性質, 計算と通信の比率による影響
 - Linpac / EP



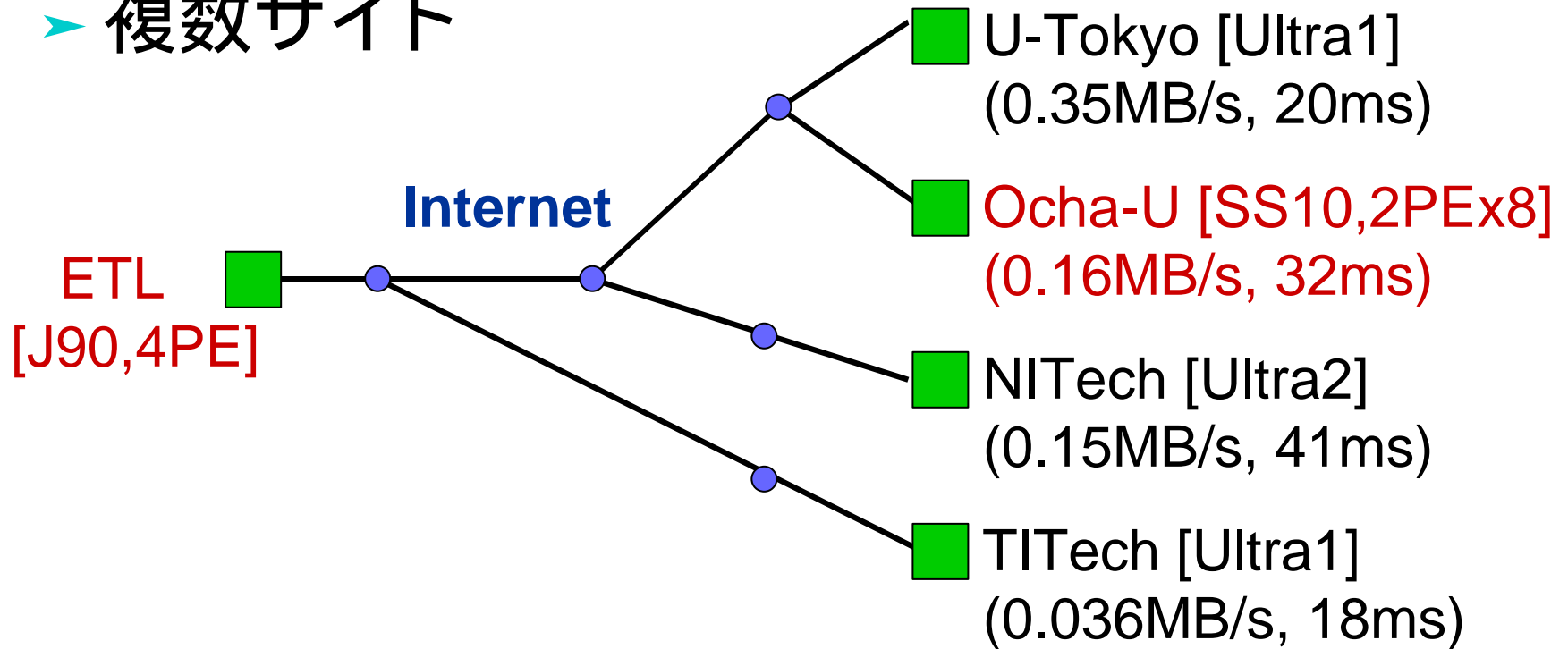
LANの計測環境

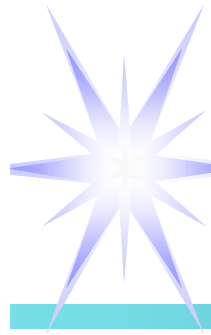




WANの計測環境

- 単一サイト
- 複数サイト





マルチクライアントでの評価条件

➤ Benchmarks

➤ Linpack: 1PE版lib - タスクパラレル

1PE でジョブを実行 / 4task同時に処理可能

4PE版lib - データパラレル

4PE でジョブを実行 / 同時に1taskのみ処理

➤ EP : 1PE版lib - タスクパラレル

➤ クライアントプログラムのモデル

Linpack / EP Benchmark を繰り返し呼び出す

➤ s [sec]毎に一定の確率 p で発生, クライアント数 c

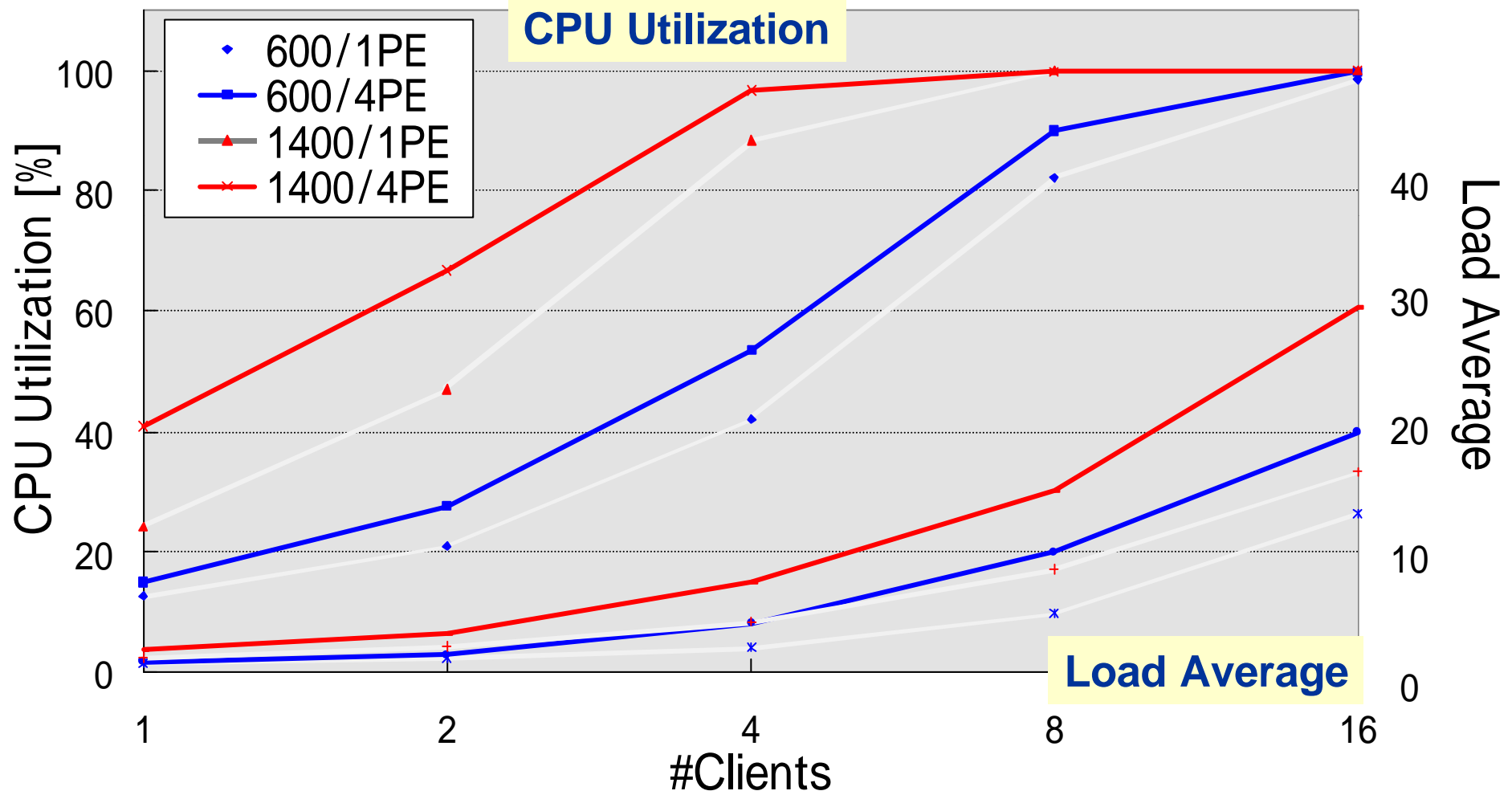
$s = 3$, $p = 1/2$, $c = 1, 2, 4, 8, 16$

➤ 問題サイズ n は試行の間一定

Linpack : $n = 600, 1000, 1400$ / EP : $n = 24$ (Sample)

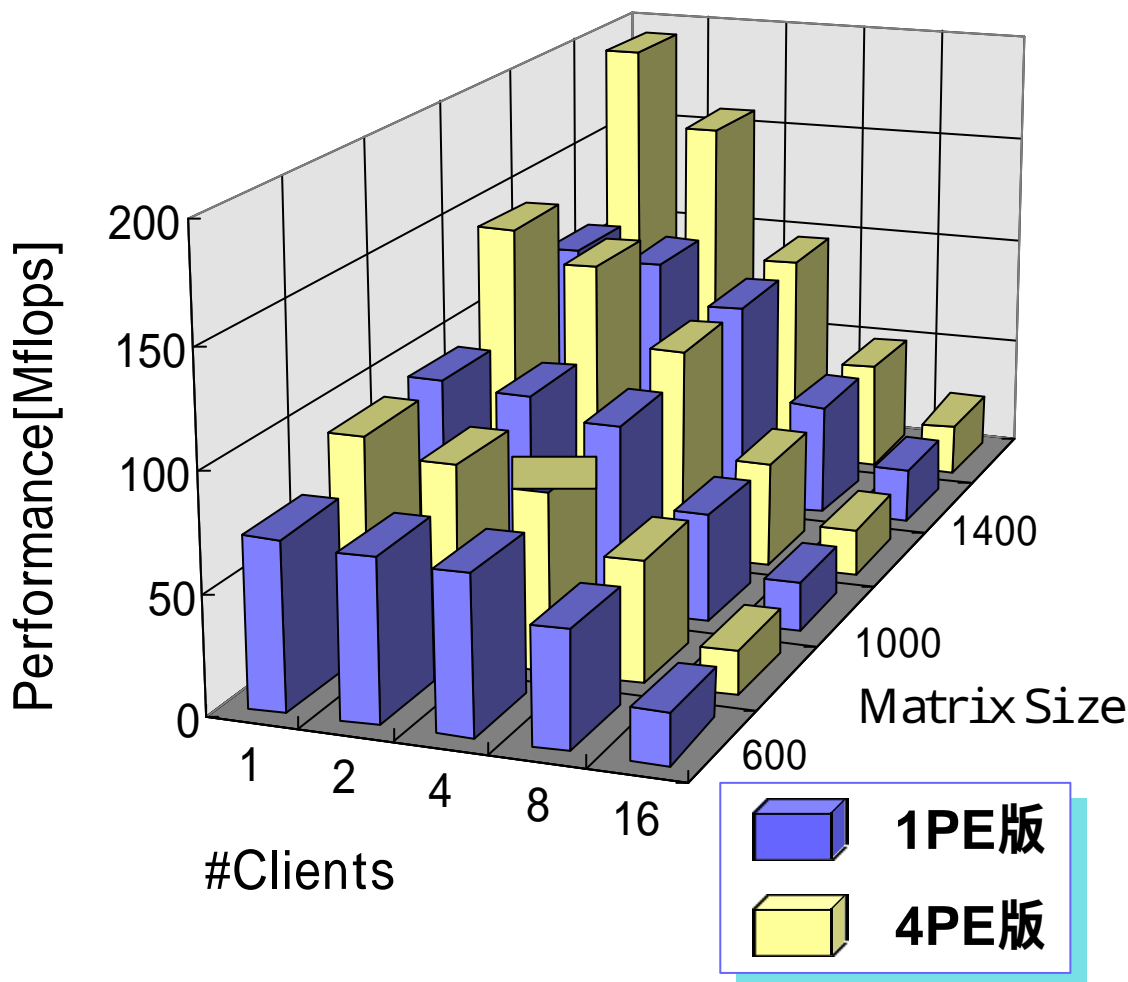


LANの測定におけるサーバの稼働率と 負荷平均値(Linpack)





LANの測定における各クライアントでの平均実効性能(Linpack)



➤ 1PE版に対する
4PE版の性能

クライアント数：

少ないとき 170 ~ 103 [%]

多いとき 105 ~ 88 [%]

➤ response time, waiting
timeは1PE版, 4PE版
の差異なし



4PE版libで有効運用



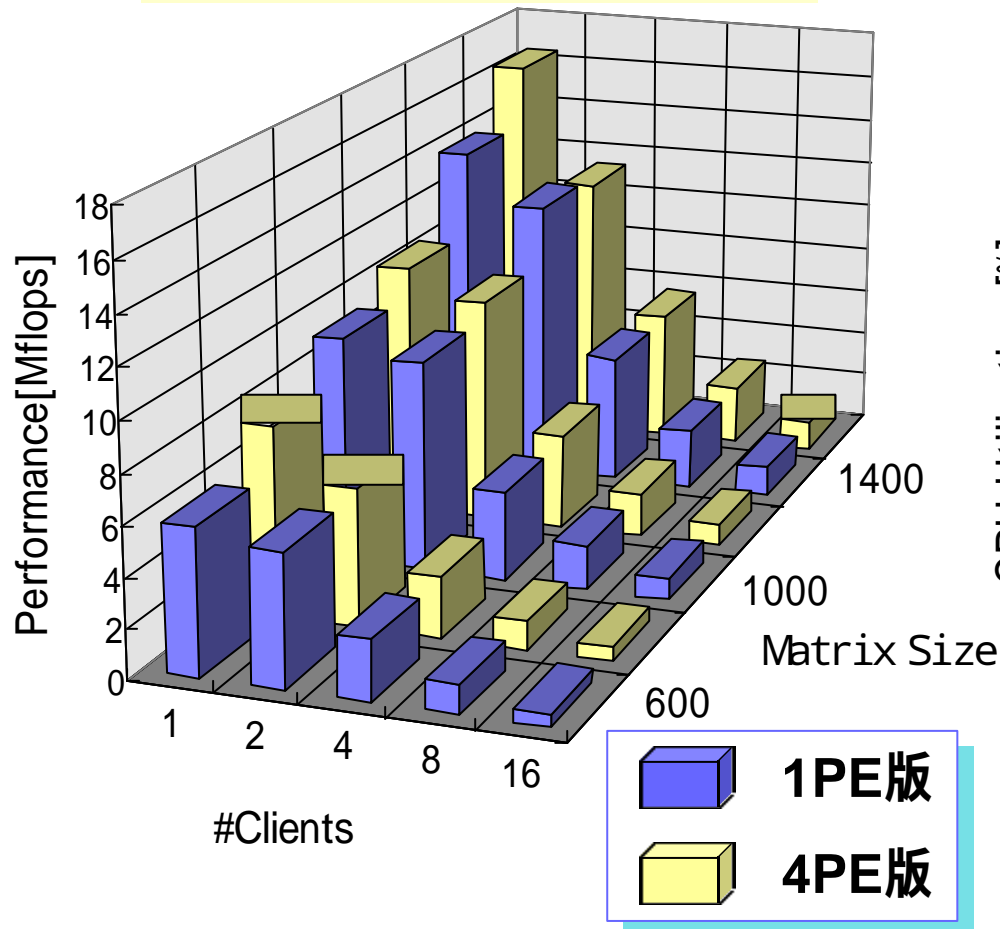
LANにおける マルチクライアントでの評価結果

- $n = 1400$, $c = 16$, 負荷 30 に達したが, 破綻せず機能した
Ninf ServerはCray J90のOS上でrobustに運用可能
- Client数 : 1 ~ 4の閑散な状態 - 4PE版が有利
 : 8 ~ 16の繁忙な状態 - 1PE版 / 4PE版の差が少ない
- 応答時間, 待ち時間は4PE版でも著しく低下しない
**Cray J90 で最高性能を実現する(データパラレル)ライブラ
リの逐次実行の方が効率が良い**

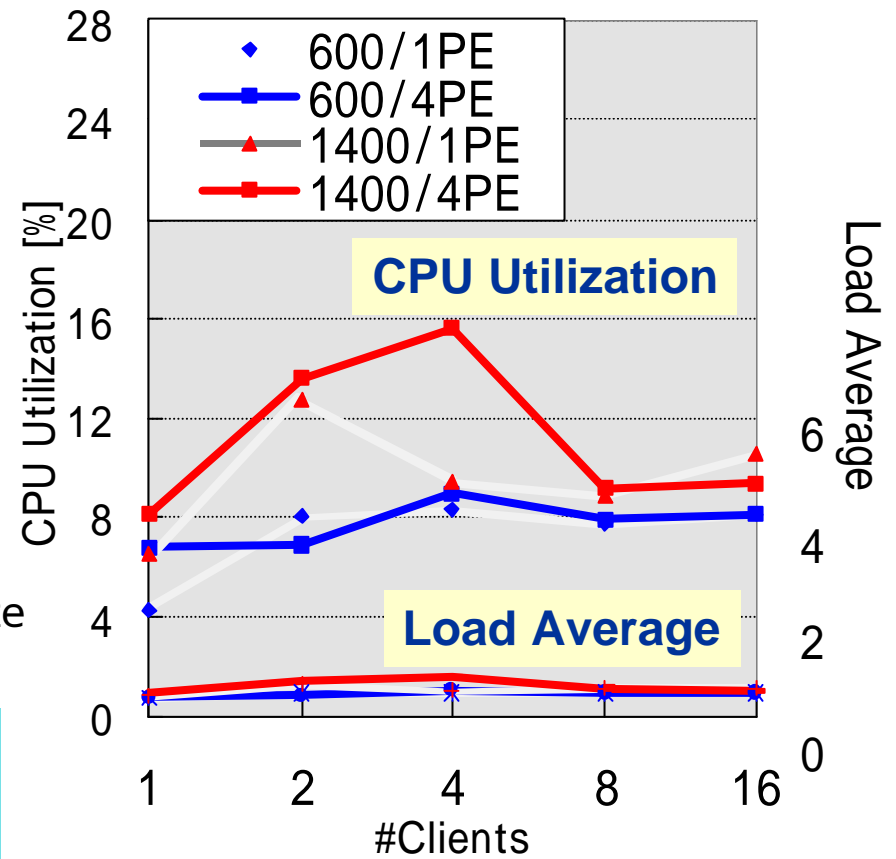


WAN:単一サイトでの測定結果 (Linpack お茶大-ETL間)

クライアントの平均実効性能



サーバの稼働率, 負荷平均値

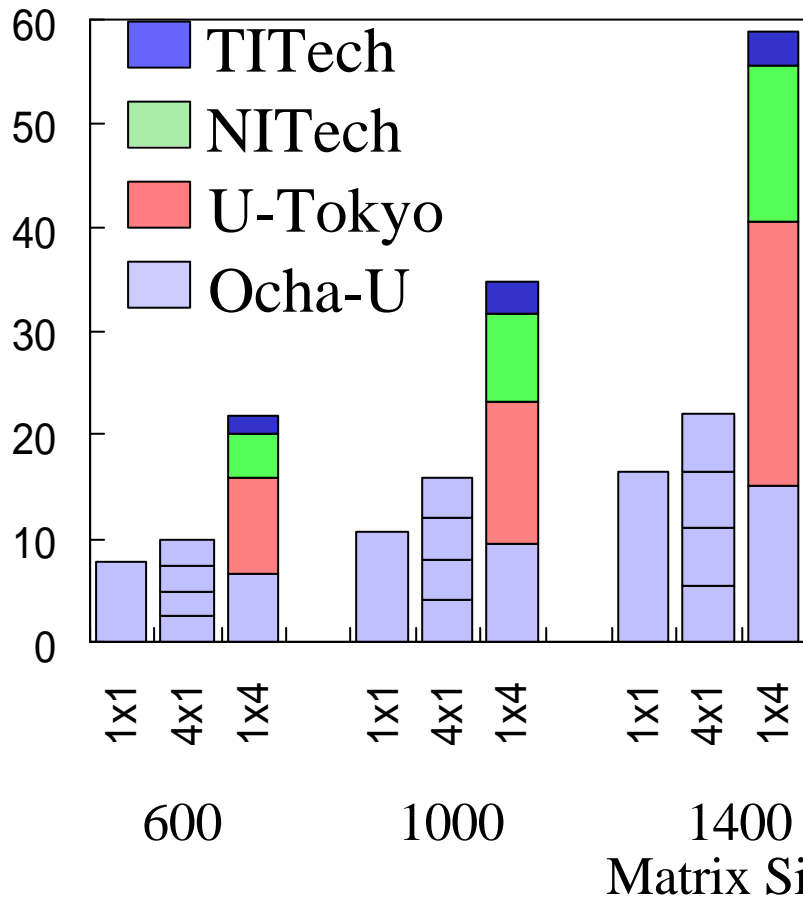




WANでの単一サイトと複数サイトの比較 (Linpack, クライアント数: 4, 4PE版lib)

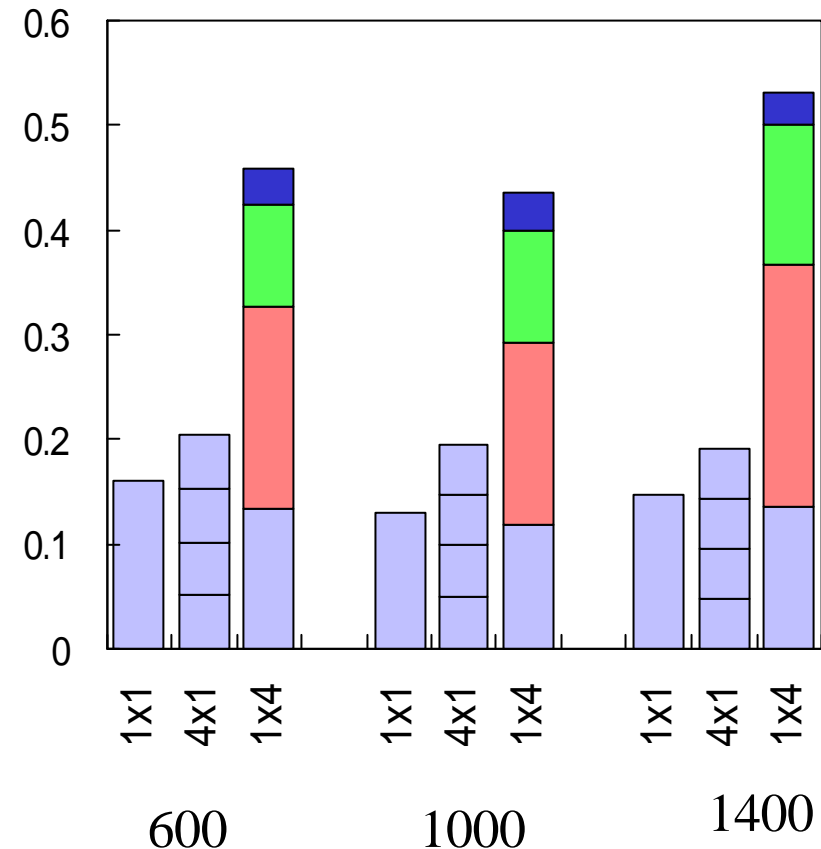
[Mflops]

実効性能の比較



[MB/s]

通信スループットの比較

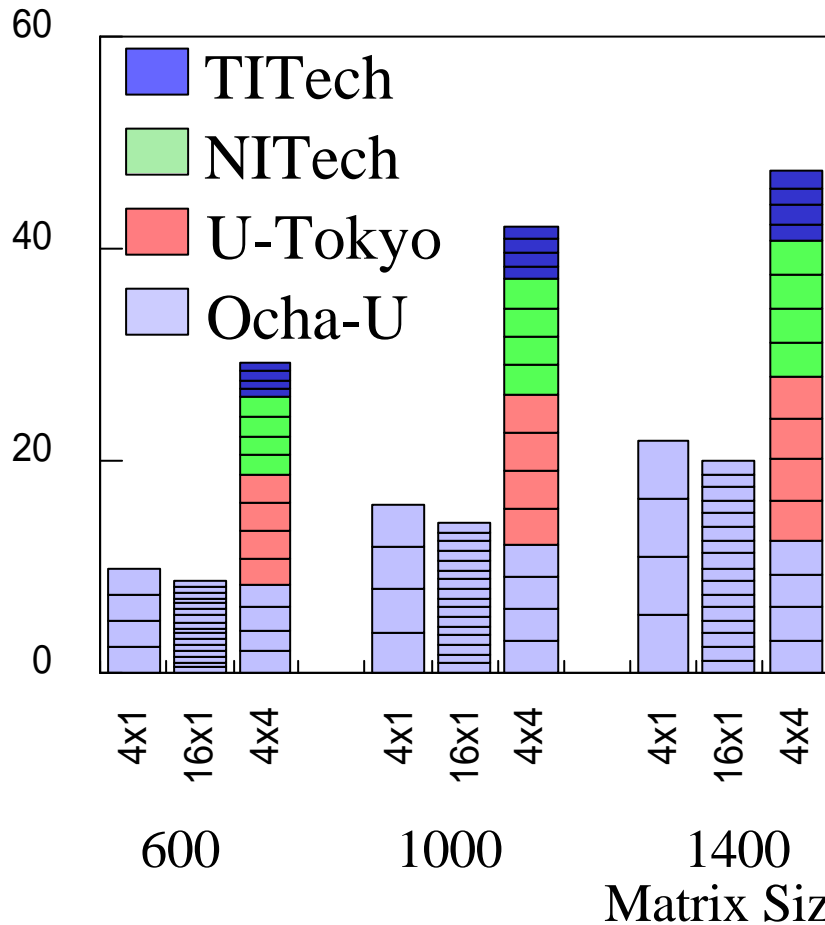




WANでの単一サイトと複数サイトの比較 (Linpack, クライアント数:16, 4PE版lib)

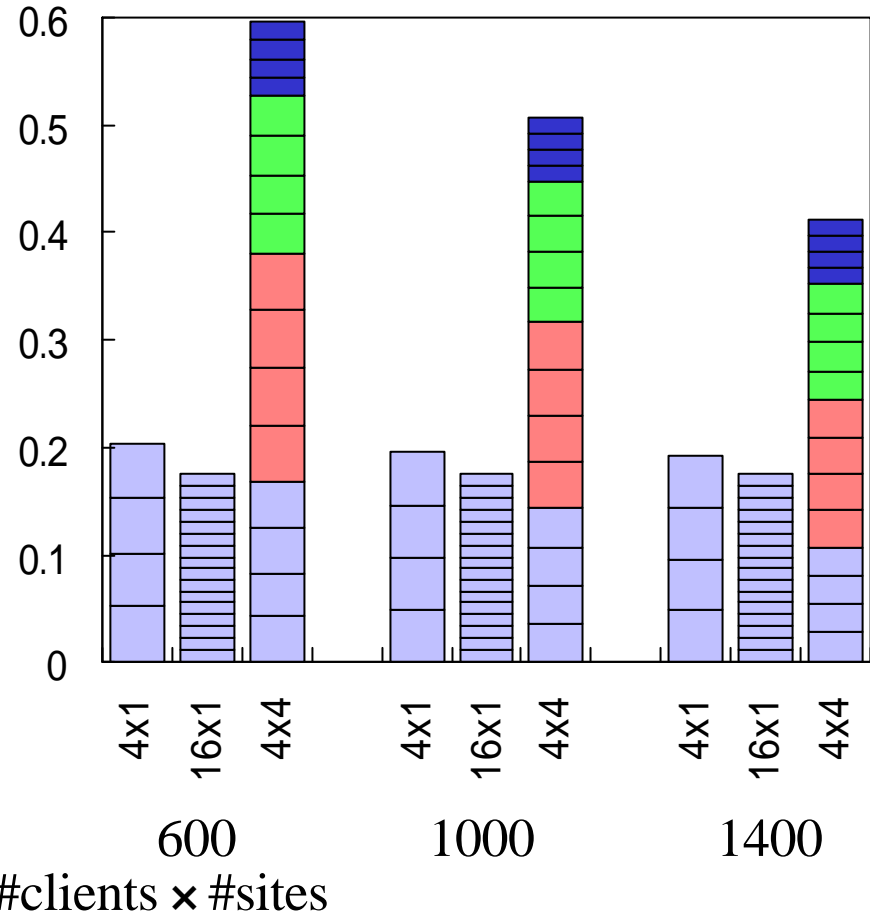
[Mflops]

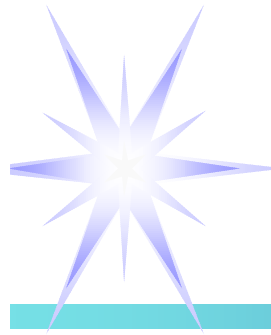
実効性能の比較



[MB/s]

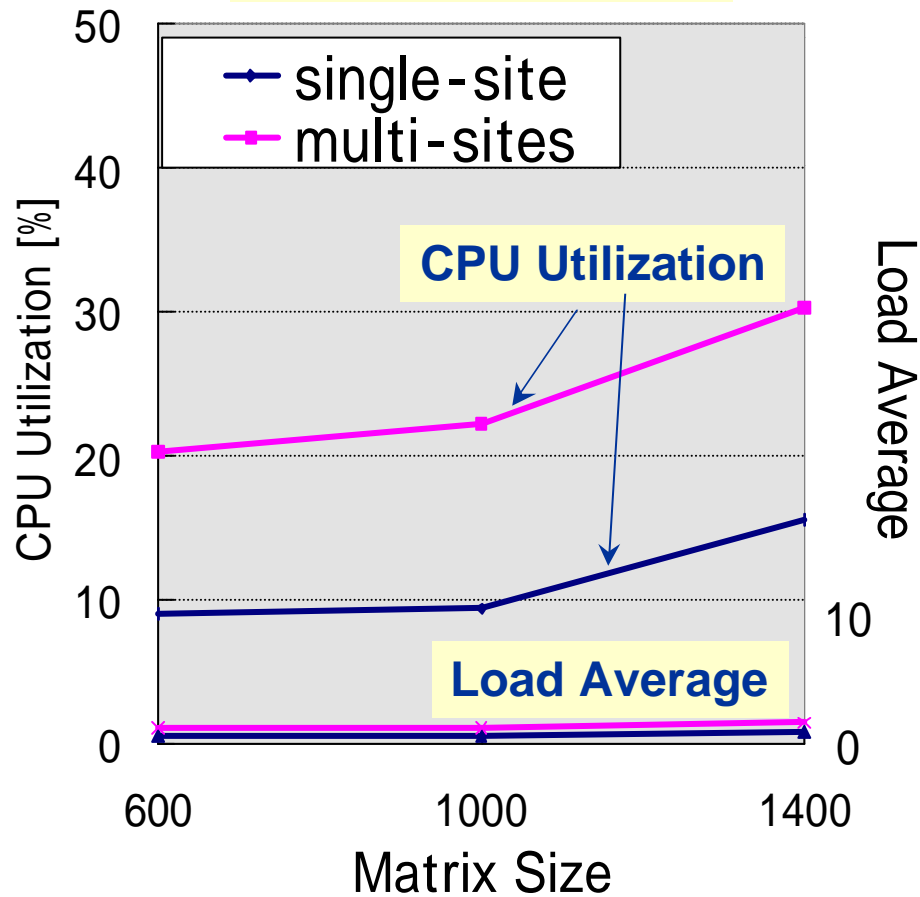
通信スループットの比較



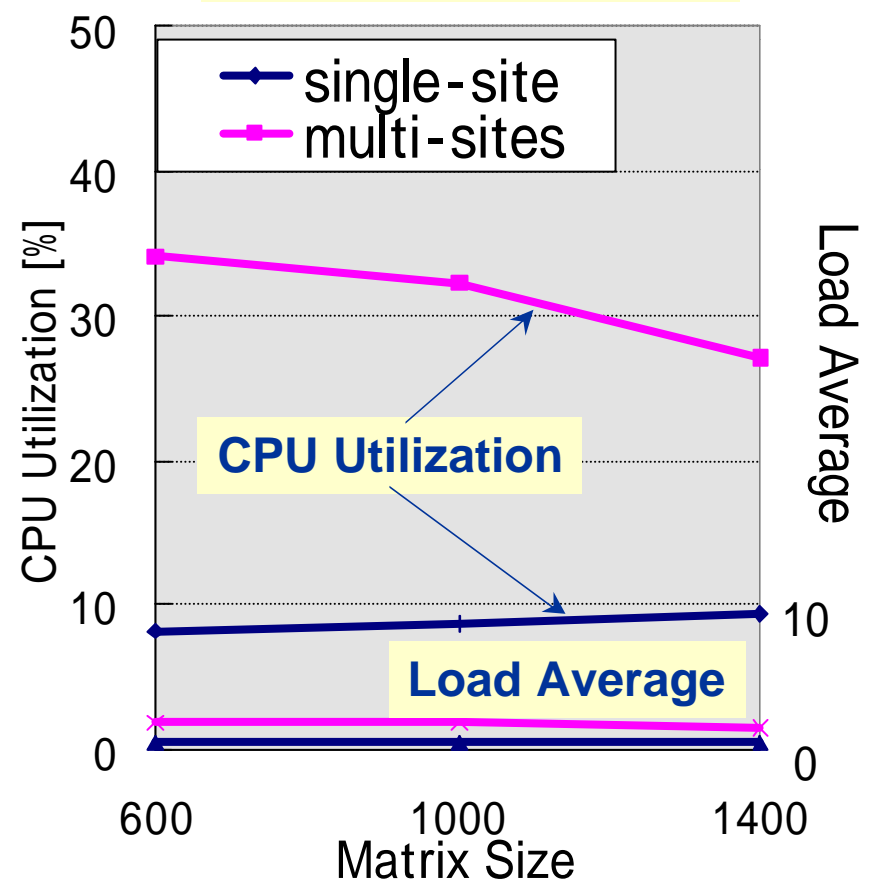


単一サイト/複数サイトでのサーバの稼働率・ 負荷平均値の比較 (Linpack, 4PE版lib)

クライアント数: 4



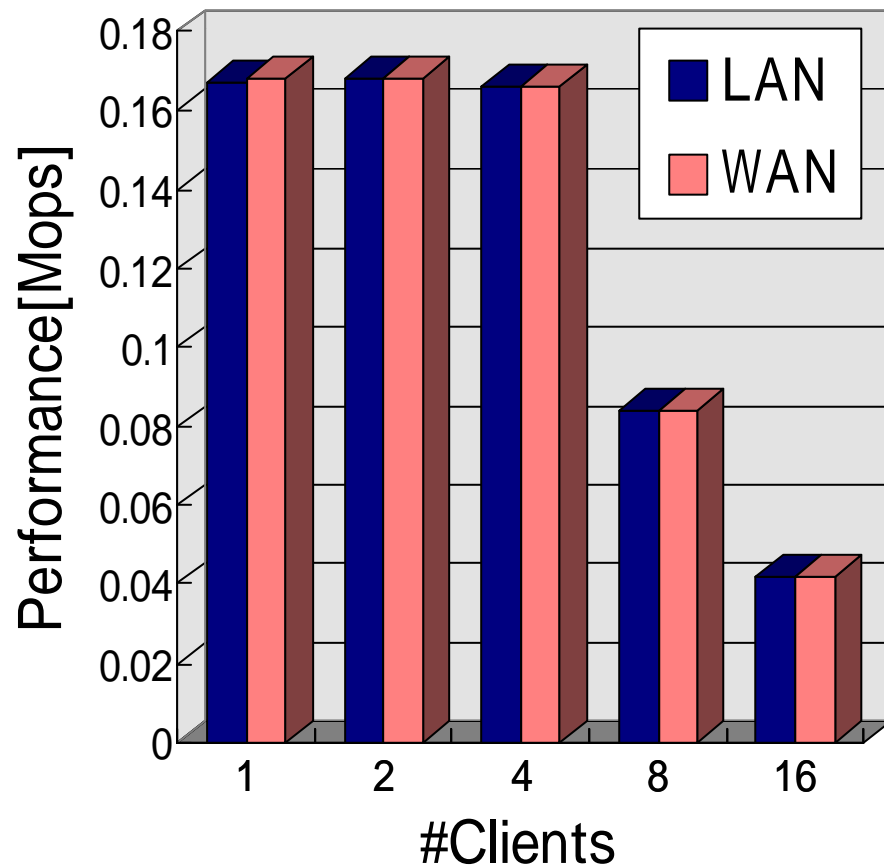
クライアント数: 16



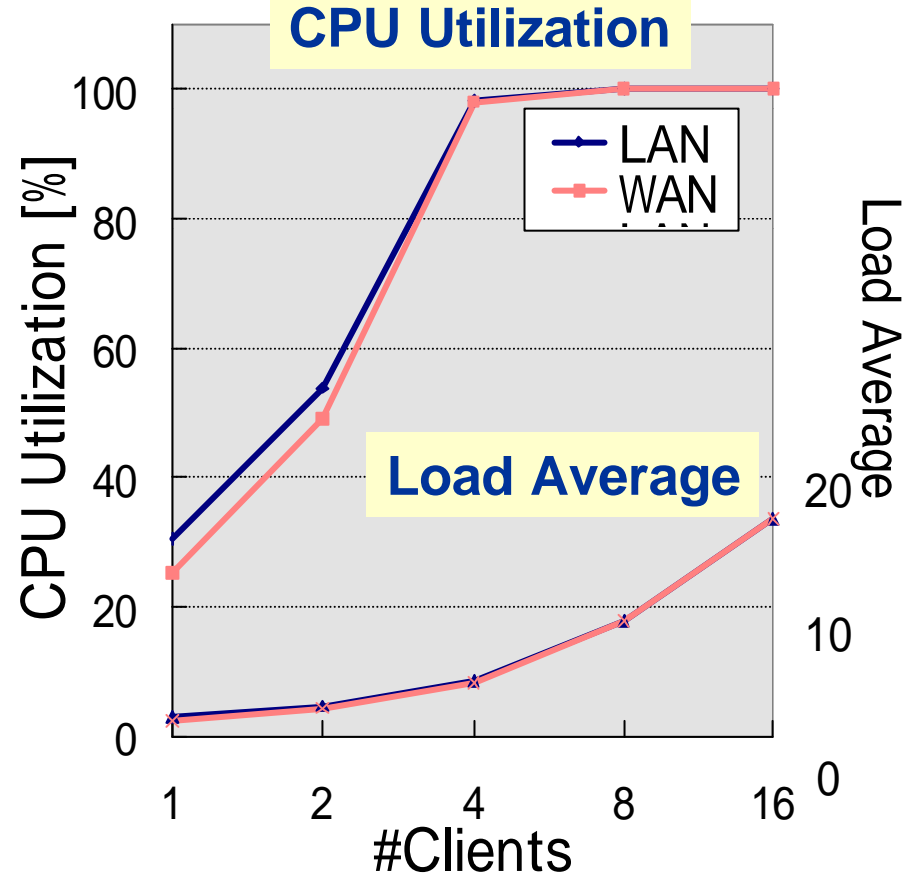


EPを用いた測定のLANとWAN の比較

クライアントの平均実効性能



サーバの稼働率, 負荷平均値

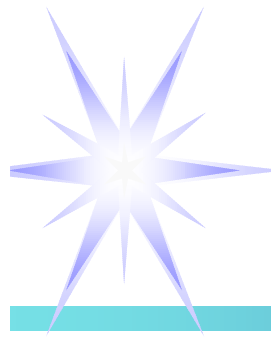




WANにおける マルチクライアント環境での評価結果

- ▶ **Linpack : 通信量が多い**
 - ▶ クライアント・サーバ間の**バンド幅が性能を決定する**
 - ▶ WANでは**通信経路がボトルネック**となり, サーバの稼働率 / 負荷が上がらない 負荷分散の良い指標にならない
 - ▶ 単一サイトから単一サーバに多数のジョブを投入すると著しく性能が低下する クライアントサイトのバンド幅の限界
 - ▶ 複数サイトから同数のジョブを投入すると総合性能が高くなる
- ▶ **EP : 通信量が極少**
 - ▶ **LAN / WAN の差がない**
 - ▶ **サーバの計算性能がボトルネックとなる**

負荷分散ではネットワークの**バンド幅**と**問題の性質**の考慮が重要



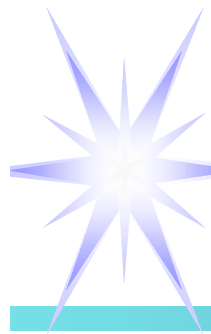
関連研究

RPC ベースシステム

- NetSolve [Casanova, Dongarra, Univ. Tennessee]
 - Ninf_callに類似のAPIを提供し, Agentプロセスで負荷分散を行う
 - サーバの平均負荷のみで負荷分散しており, ネットワークの混雑状態・バンド幅を考慮すべきである
 - その他、DB Server に対する API がない点などが Ninf と異なる
- RCS [Arbenz, ETH Zurich]

並列分散言語等を用いたシステム

- Legion/Mentat [Grimshaw, Univ. Virginia]
 - 独自の並列オブジェクト指向言語Mentatを用いたGlobal Computing system
 - 最適化・機能追加が容易だが, 既存ライブラリ資源の再利用性は?
- Javelin [Schauer et al., UCSB]
 - 汎用部品(Java, WWW)を用いて実現されたGlobal Computing system
 - 性能は?



本研究のまとめ

- Ninfでネットワークコンピューティングが有効運用できる
- Clientのプラットフォーム, マシン性能は, Ninf_callの実効性能にほとんど影響しない
- 現状のスーパーコンピュータはGlobal Computingを実現するために十分に耐久性のあるプラットフォームである
- ベクトル並列計算機ではデータパラレルlibが有効である
- 通信主体の問題ではネットワークのバンド幅 / 混雑度が性能に大きく影響する
通信主体/計算主体のジョブを適切にサーバに分散するスキームが重要



Global Computing を目的とするシステムに共通



今後の課題

- **Global Computing のための負荷分散**
 - メタサーバの強化
 - 性能保証
 - サーバのジョブハンドリング
 - さまざまな性質のアプリケーションによる評価
 - Global Computing のモデルを立てる シミュレータによる解析
 - **サーバの性能 / 負荷, 問題の性質(通信主体 / 計算主体), 計算規模, ネットワーク距離/位置/スループットが異なる条件での動的負荷分散**
- **MPPへの対応**
 - 1つのジョブに費やすPE数の選択と複数ジョブ間のスケジューリング
 - ネットワークサービスに対する考慮 API の拡張