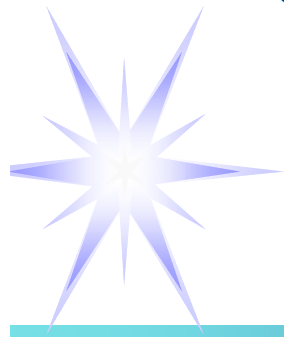


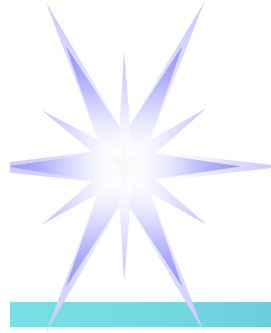
ネットワーク数値情報システム Ninf: マルチクライアント環境での性能



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}

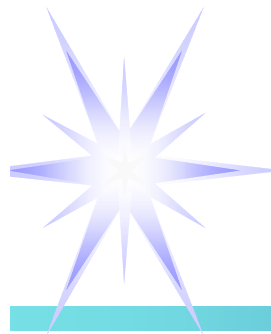
^{*1} お茶の水女子大学, ^{*2} 東京大学, ^{*3} 東京工業大学,
^{*4} 電子技術総合研究所, ^{*5} 新情報処理開発機構

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



発表内容

- Ninf の概要
- Linpack を用いたシングルクライアント環境での性能
- Linpack を用いたマルチクライアント環境での性能
- まとめと今後の課題



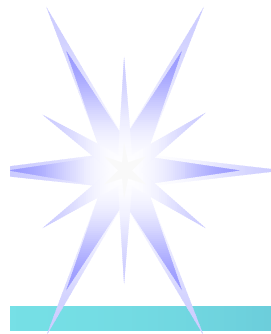
はじめに

ネットワーク数値情報システム **Ninf**

(Network Information Library towards
a Globally High Performance Computing)

広域分散並列計算技術を支援するシステム

- 広域分散並列処理を有効に行う条件
 - 信頼性のある通信手続き
 - 圧倒的に高いサーバの計算性能

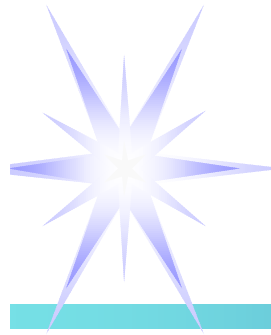


本研究の目的

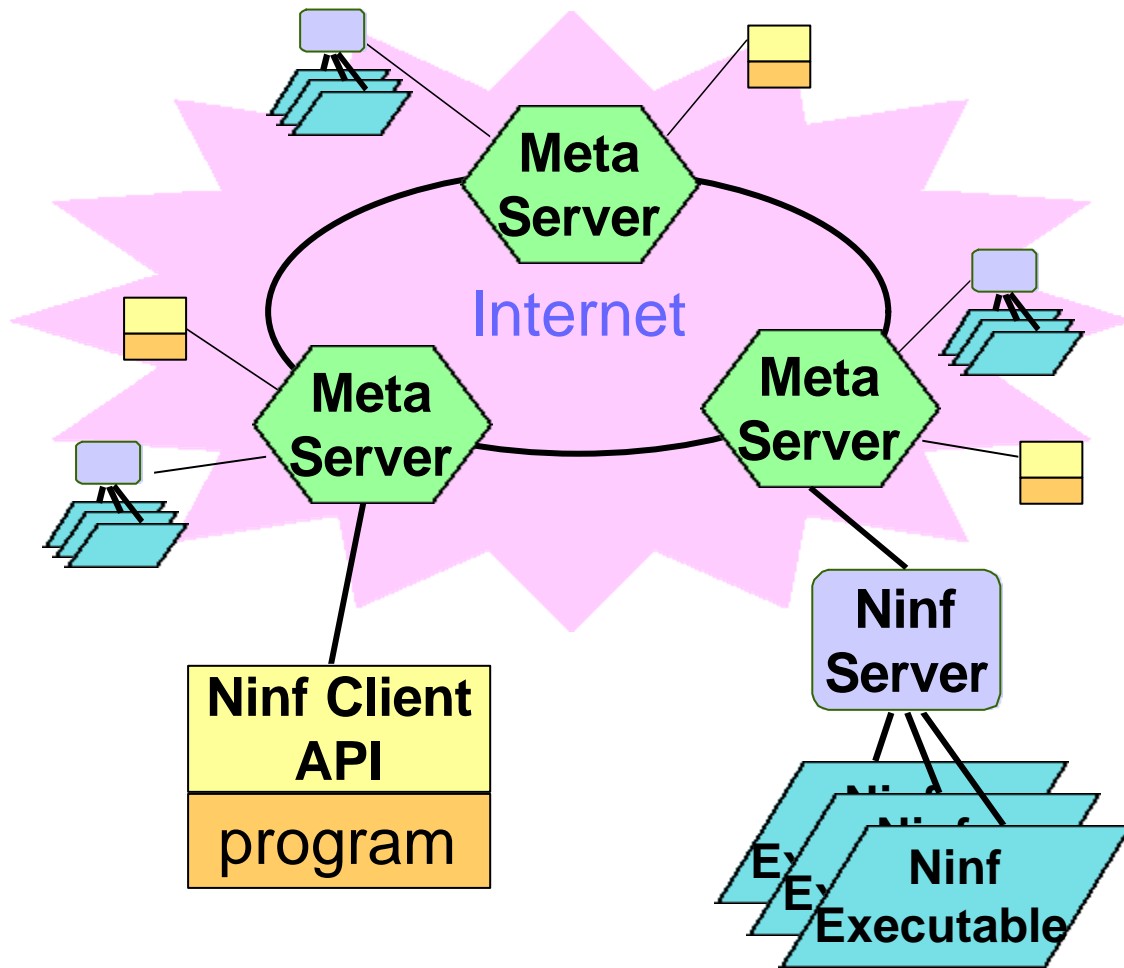
- より現実的なNinf システムの運用
 - クライアントからの計算要求の頻度
 - 個々の計算規模
 - 並列計算機では**ライブラリの選択**
 - Scaler job の Parallel execution
 - Parallel job の Single execution



マルチクライアント環境での測定で検証



Ninfシステムアーキテクチャ



- Ninf サーバ
- Ninf クライアント API
- メタサーバ

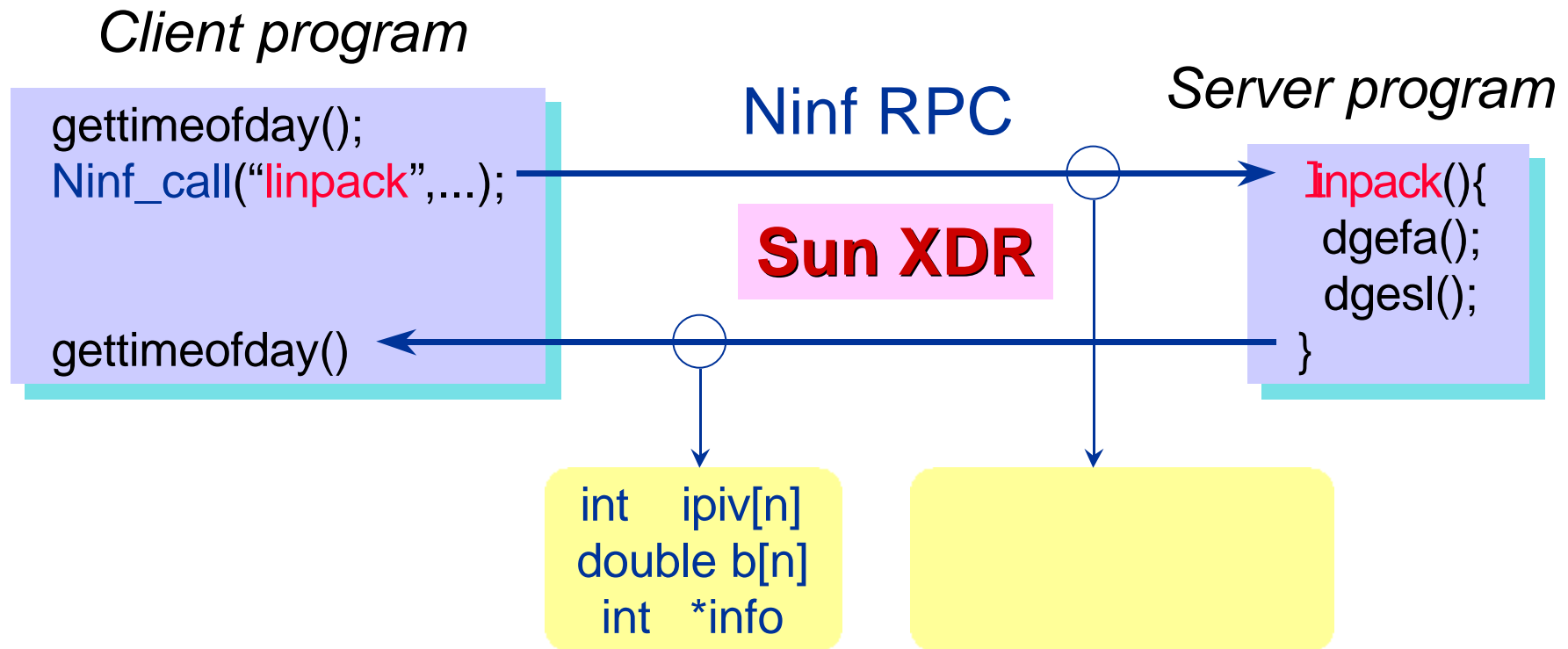


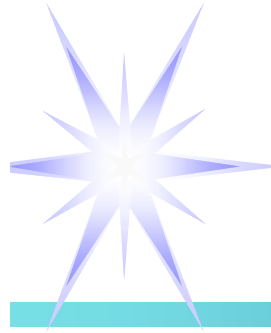
Ninf RPC (Remote Procedure Call)

により実現



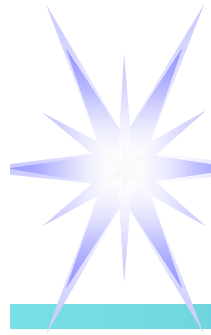
Ninf RPC による Linpac Benchmarkプログラムの実行



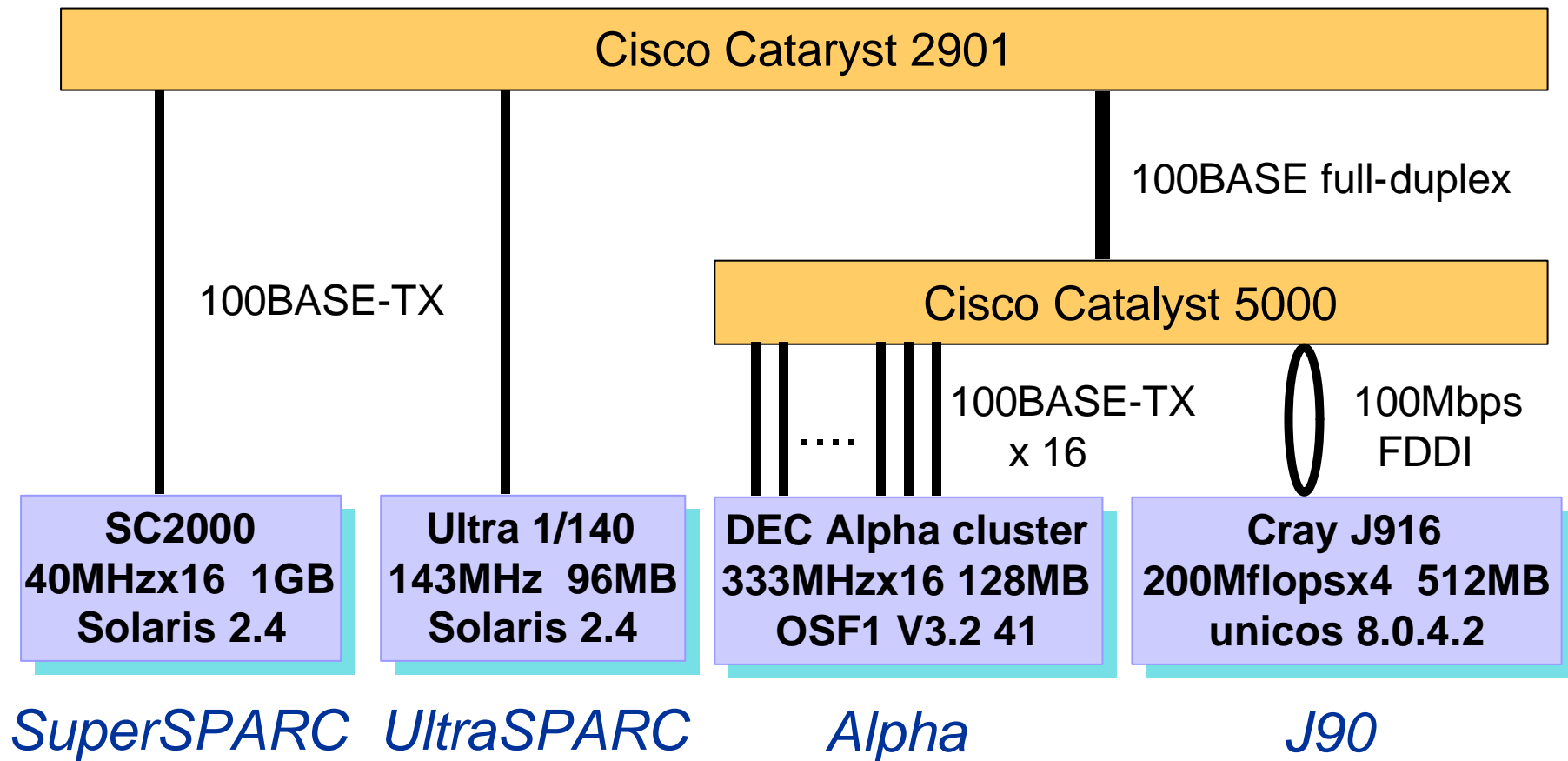


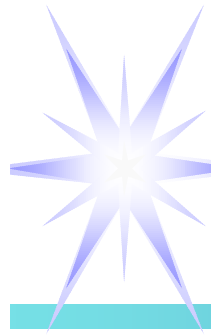
Linpack Benchmark

- ▶ 倍精度のLinpack Benchmark :
 - ▶ ガウスの消去法で密行列の連立一次方程式を求解
 - ▶ 演算量 : $2/3 n^3 + 2 n^2$
 - ▶ 通信量 : $8 n^2 + 20 n + O(1)$ [bytes]
 - ▶ Ninf_callの性能 :
 $(2/3 n^3 + 2 n^2) / (\text{通信時間} + \text{計算時間})$ [Mflops]



計測環境





シングルクライアント環境での評価

➤ Linpack Benchmarkのルーチン

J90(4PE)上で libSci ライブラリ

(sgetrf, sgetrs)

1PE版, 4PE版ライブラリ

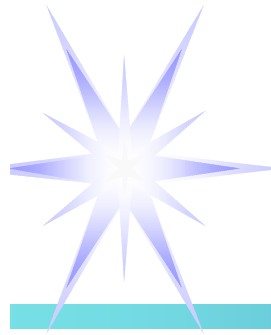
その他 : LAPACK (dgefa, dgesl)

➤ 計測条件

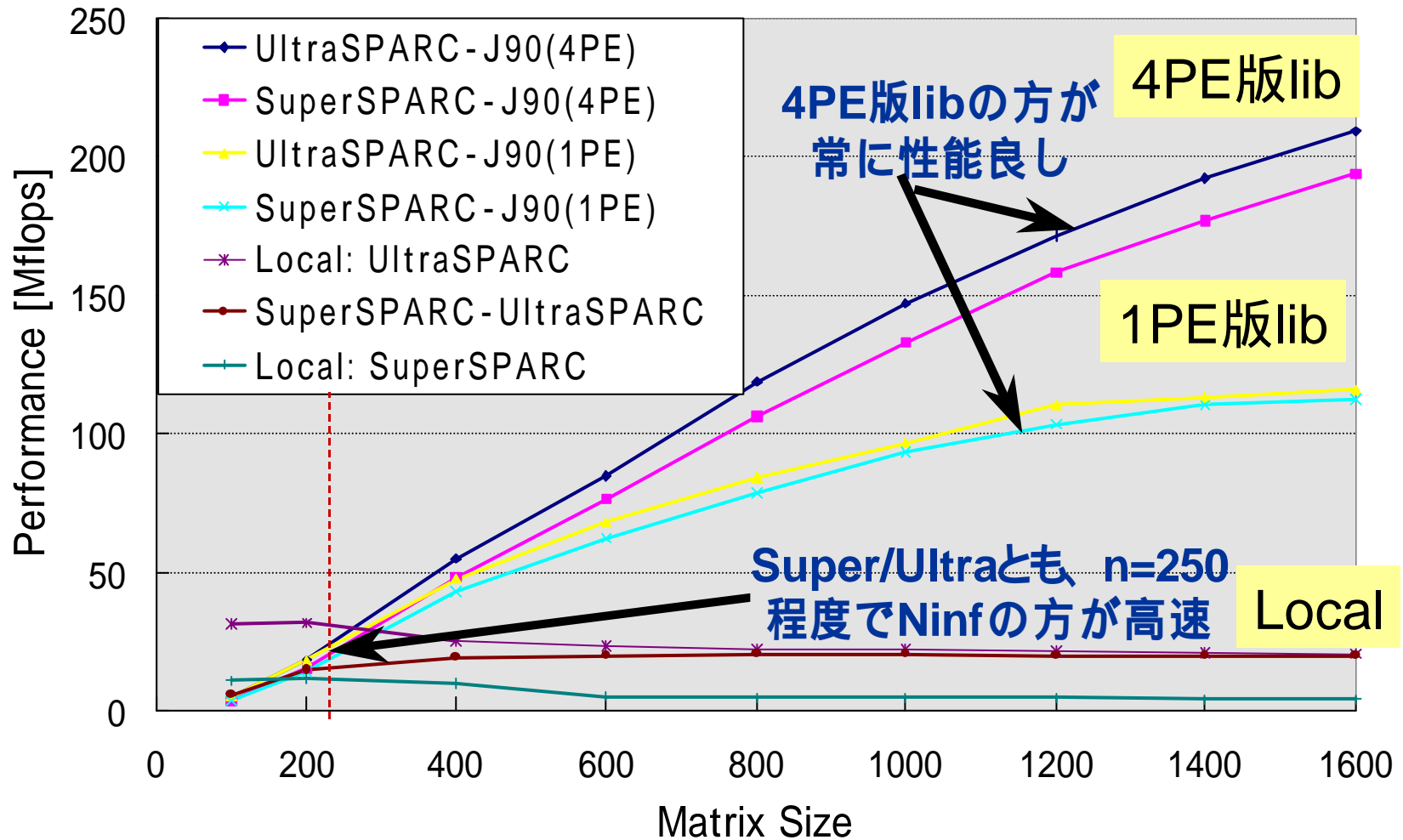
Ninf_call の回数 : 20回

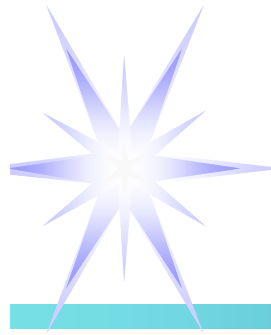
採用した値 : 最高性能値

Client	Local	Remote (Ninf_call)	
		Ultra	J90
		1PElib	4PElib
SuperSPARC			
UltraSPARC		-	
Alpha		-	

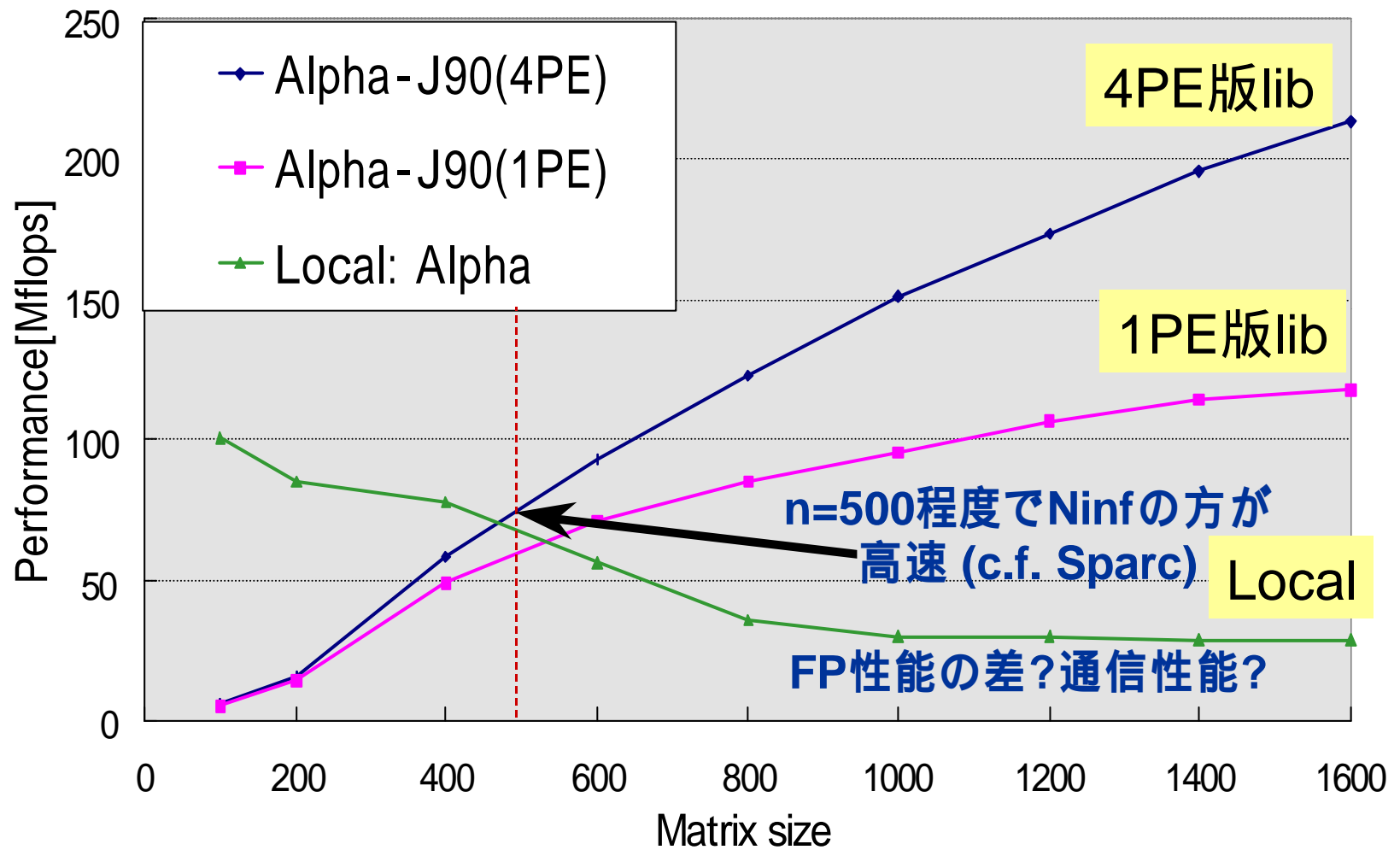


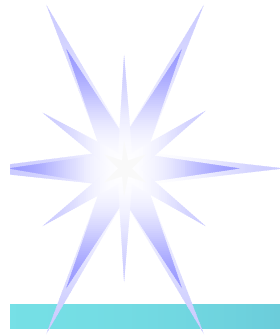
SPARCをクライアントとした性能



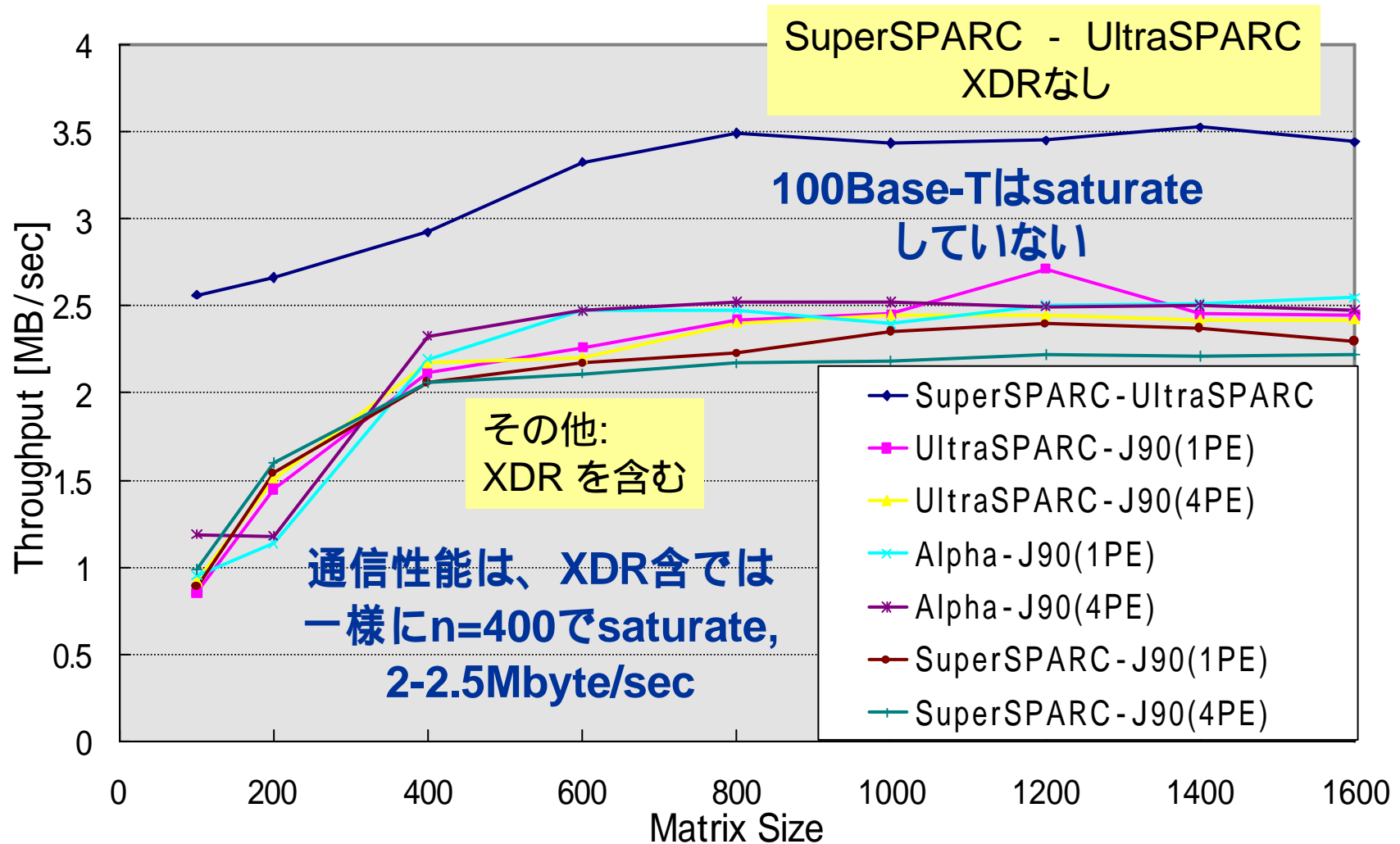


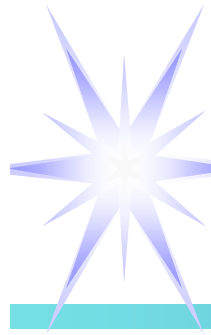
Alpha をクライアントとした性能





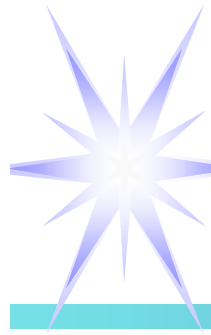
Ninf_call通信スループット





シングルクライアント環境での結果

- 問題サイズが大きくなると**通信のオーバーヘッドが隠蔽**
- **クライアントマシン性能が異なる**場合も同程度の性能向上
- データ表現が異なるプラットフォーム間でも**Ninf_callの効率**は著しく低下しない
- SPARC をクライアントとした場合
 - 問題サイズ200 ~ 400で Local 実行より Ninf を用いた方が高性能
 - 通信データ表現の変換のオーバーヘッドが小さい
- Alpha をクライアントとした場合
 - 問題サイズ400 ~ 600で Local 実行より Ninf を用いた方が高性能

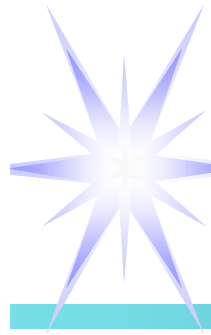


マルチクライアント環境での評価

- より現実的なNinf システムの運用
 - クライアントからの計算要求の頻度
 - 個々の計算の規模
 - 並列計算機では**ライブラリの選択**
 - Scaler job の Parallel execution
 - Parallel job の Single execution



マルチクライアント環境での測定で検証

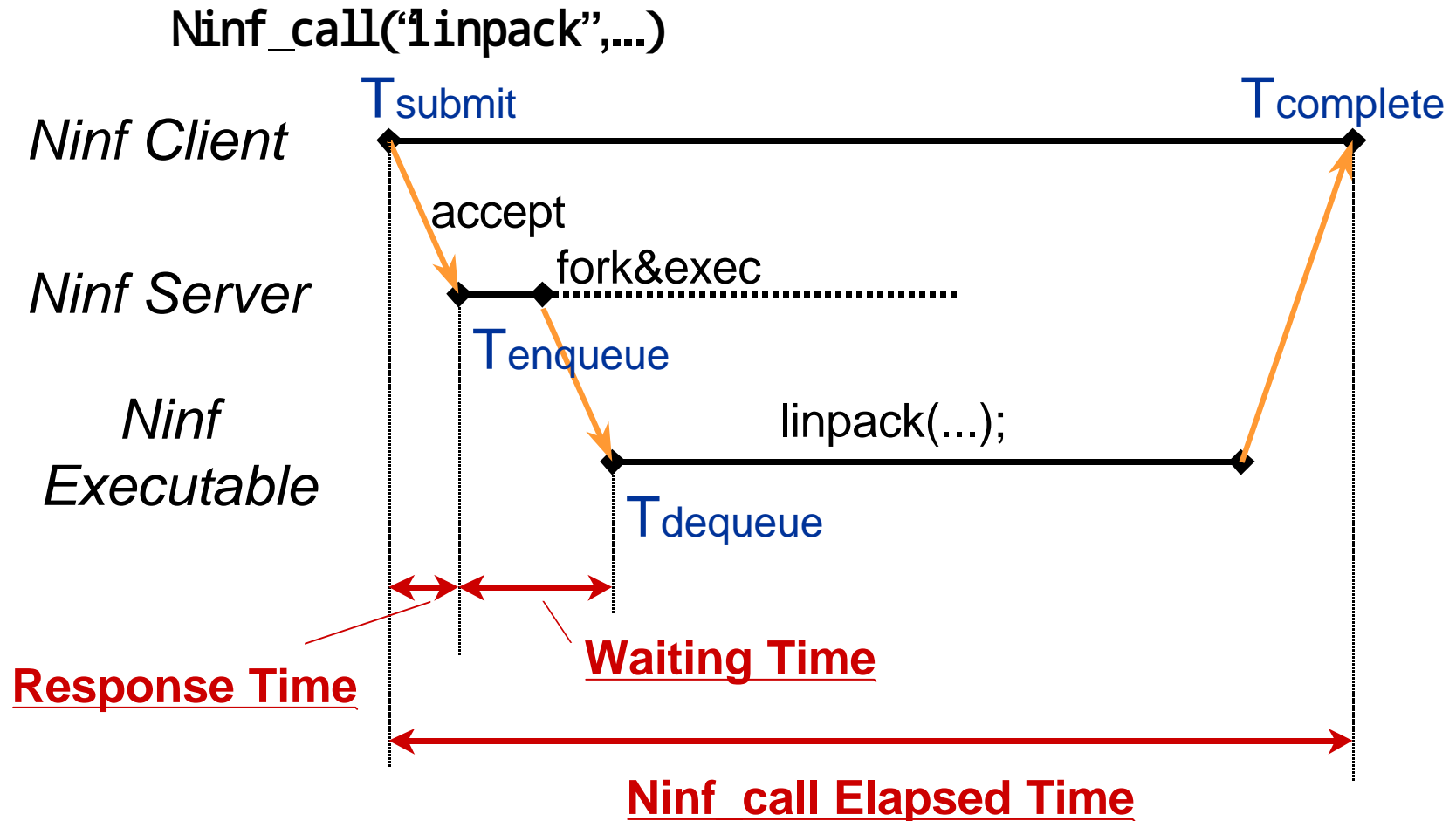


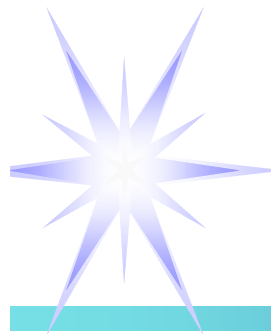
マルチクライアント環境での評価条件

- サーバ : J90(4 PE), クライアント : Alphaクラスタ
- クライアントプログラムのモデル
Linpack Benchmark を繰り返し呼び出す
 - s [sec]毎に一定の確率 p で発生
 - 問題サイズ n は試行の間一定
 - クライアント数 c
 $s=3, p=1/2, c=1,4,8,12,16, n=600,1000,1400$
- Linpackのルーチン : 1PE版lib , 4PE版lib(vector)

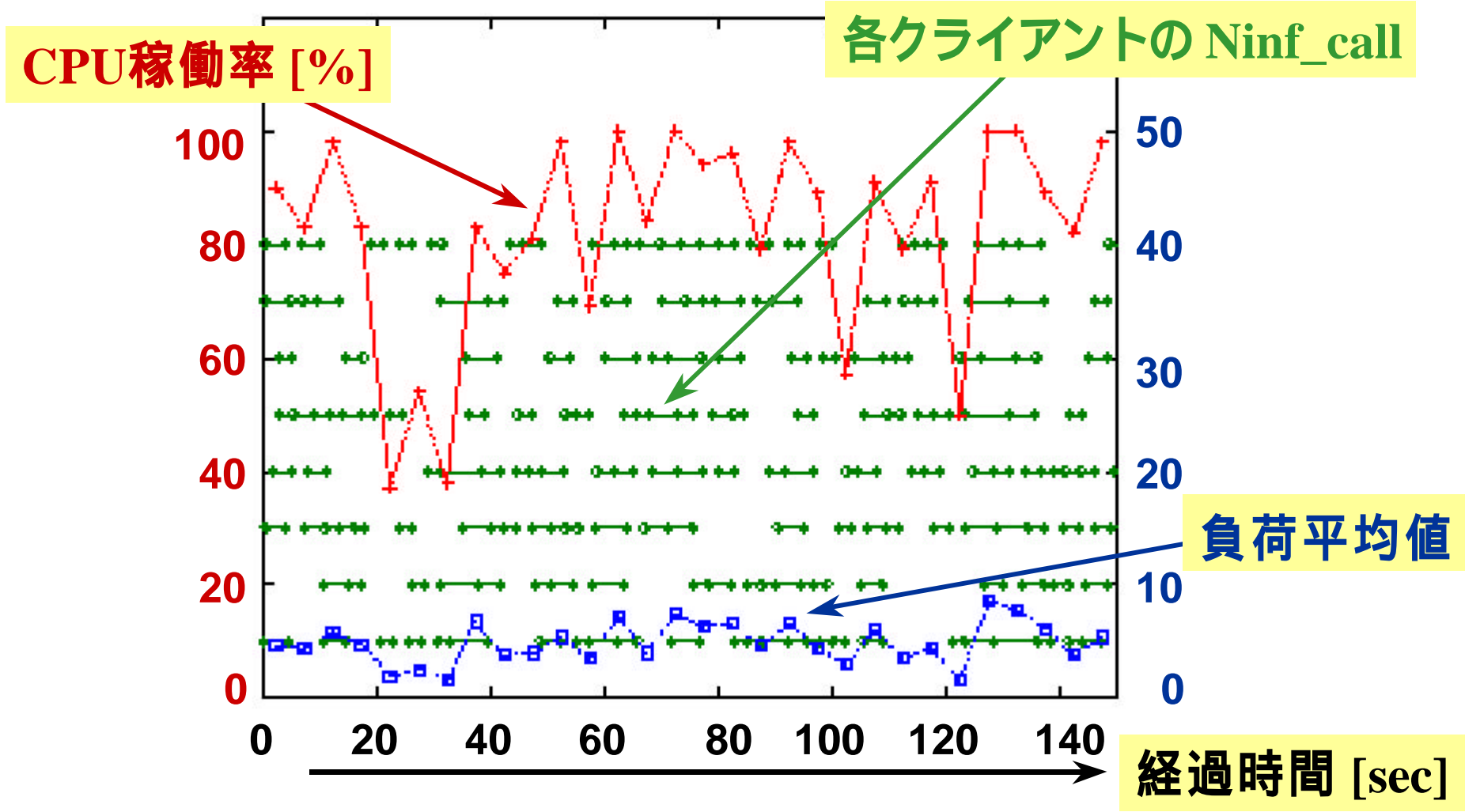


Ninf_callの測定のタイミング





グラフの見方



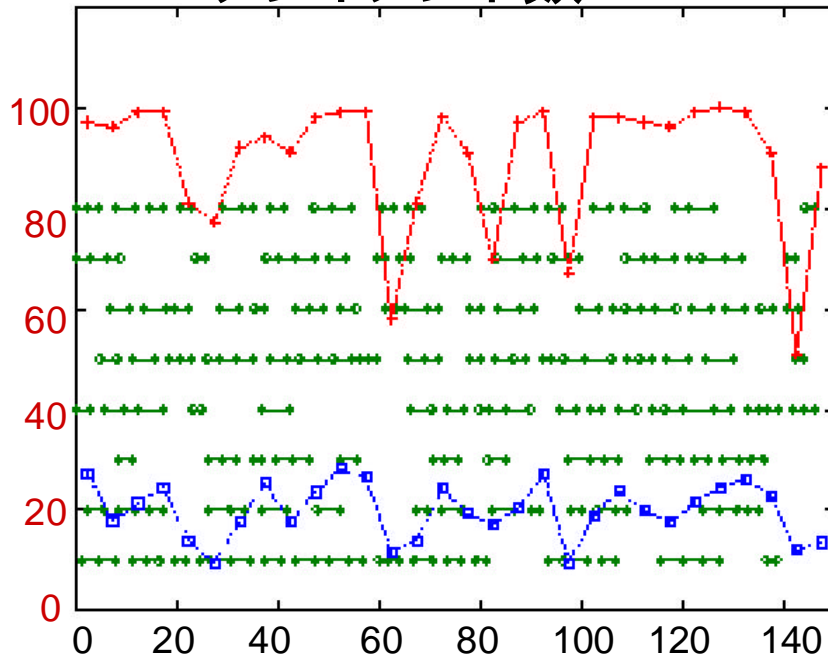


クライアント数による比較

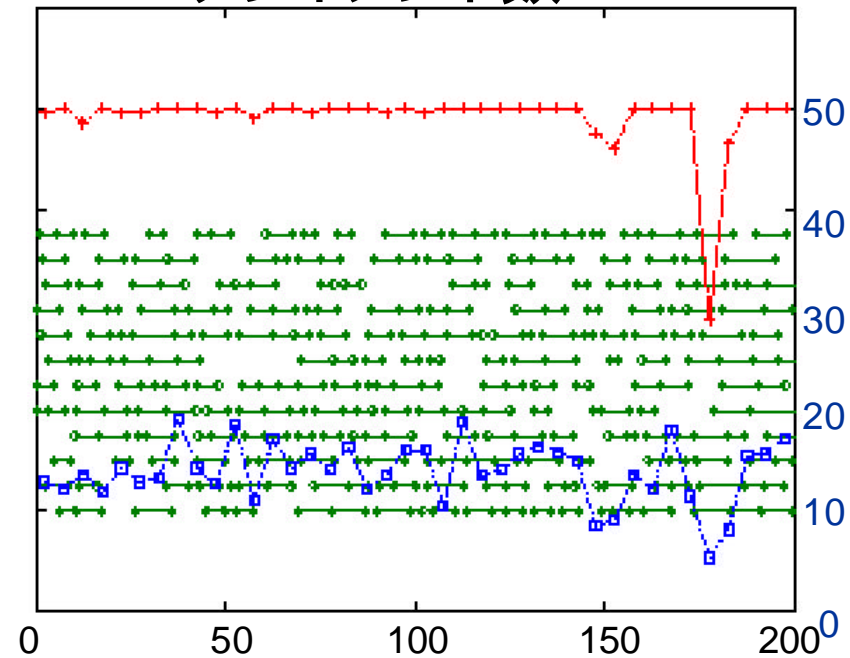
4PE版, 問題サイズ: 600

- 稼働率: 高
- 性能のばらつき(応答時間)

クライアント数: 8

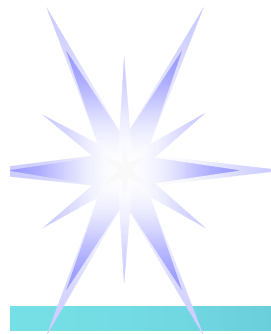


クライアント数: 12



稼働率 [%]
 負荷平均値
 各クライアントのNinf_call
 経過時間 [sec]

クライアント数	Performance[Mflops] max / min / mean	稼働率	負荷平均
8	89.35/24.31/51.51	90.00	9.98
12	71.83/11.26/29.83	98.15	13.96

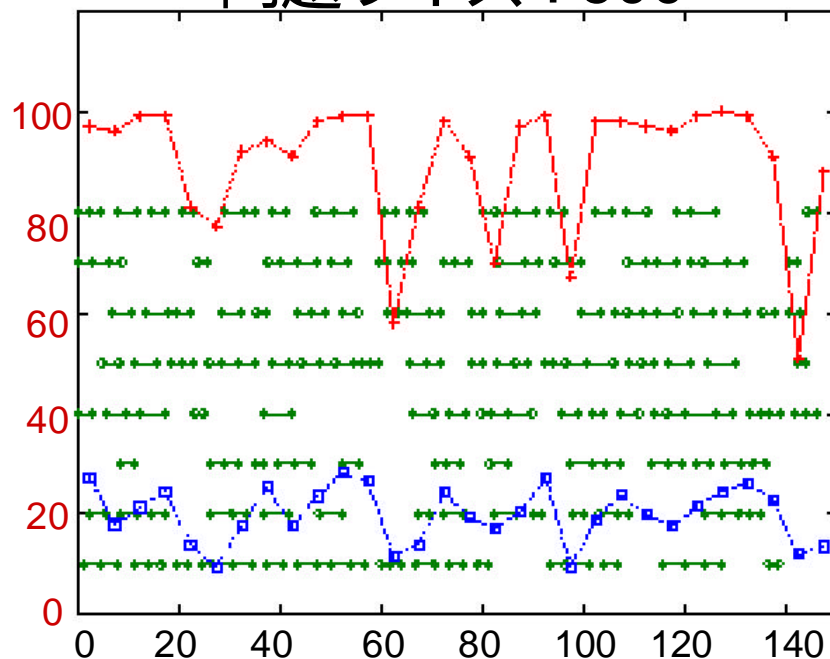


問題サイズによる比較

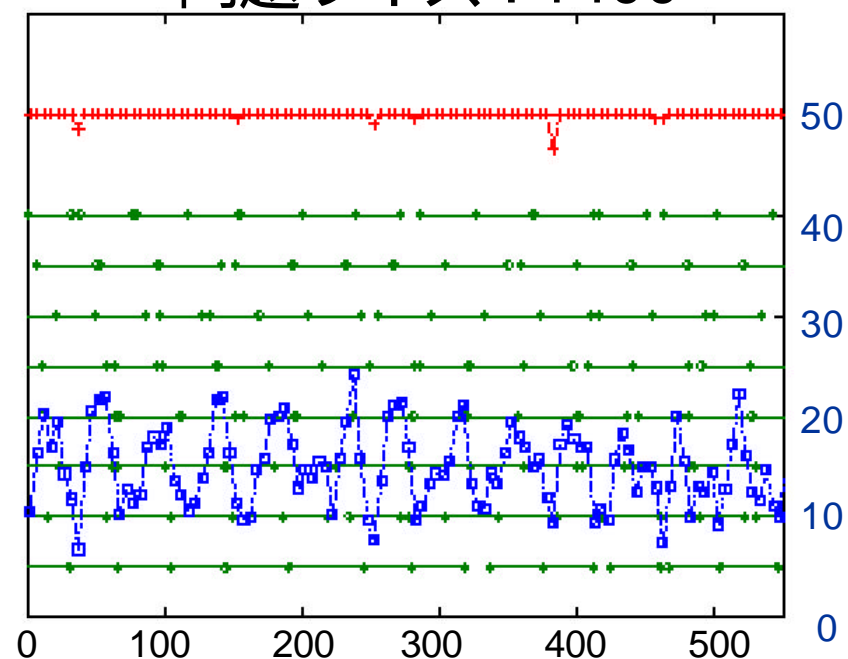
4PE版, クライアント数: 8

- ・稼働率: 高、負荷: 高
- ・稼働性・性能は維持

問題サイズ: 600

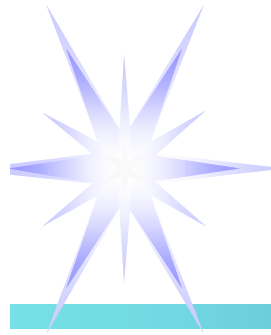


問題サイズ: 1400



稼働率 [%]
 負荷平均値
 各クライアントの Ninf_call
 経過時間 [sec]

問題サイズ	Performance[Mflops] max / min / mean	稼働率	負荷平均
600	89.35 / 24.31 / 51.51	90.00	9.98
1400	60.26 / 24.22 / 49.27	99.86	21.69

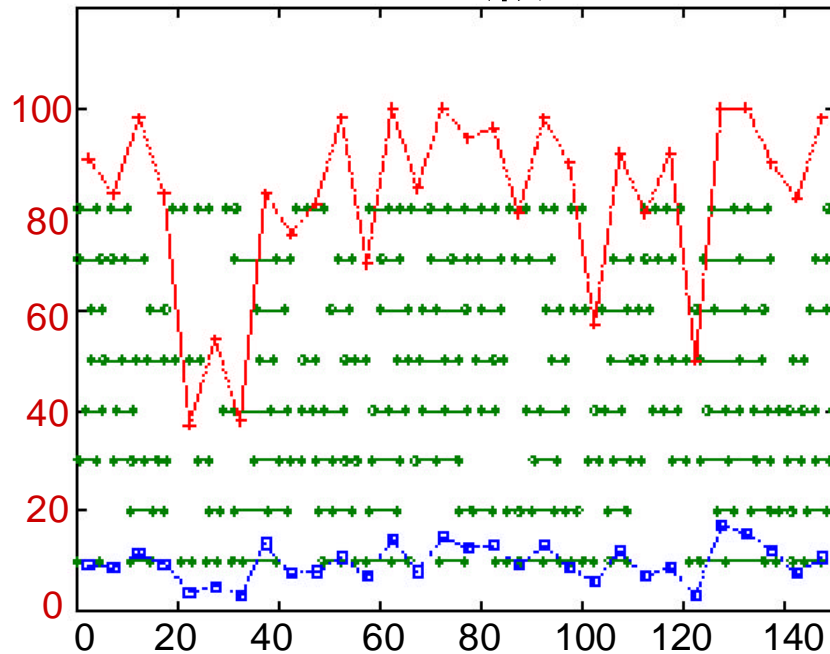


1PE版と4PE版の比較

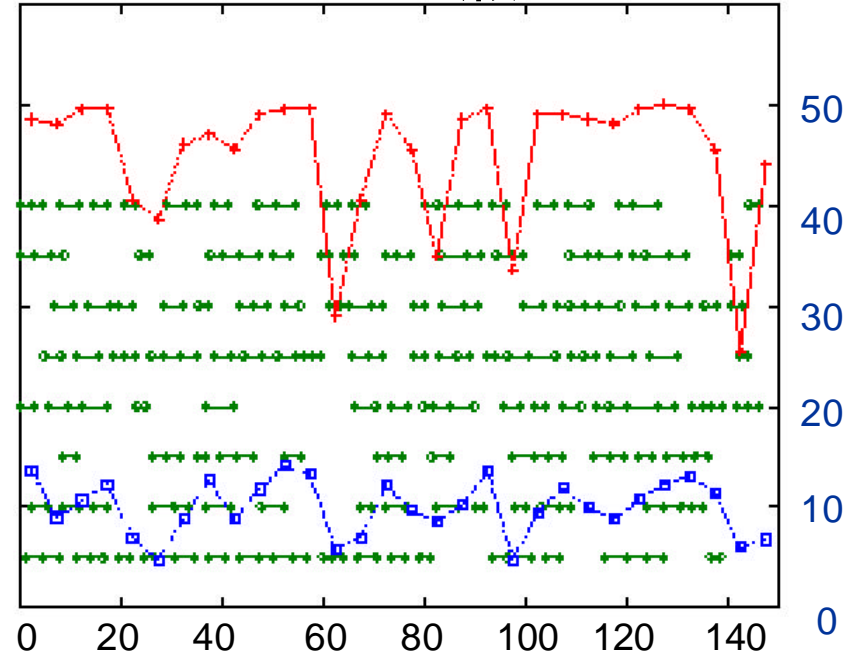
クライアント数: 8, 問題サイズ: 600

- ・4PE版 : 高稼働率、高負荷
- ・4PE版の方が性能が良い

1PE版

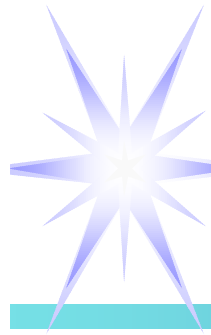


4PE版

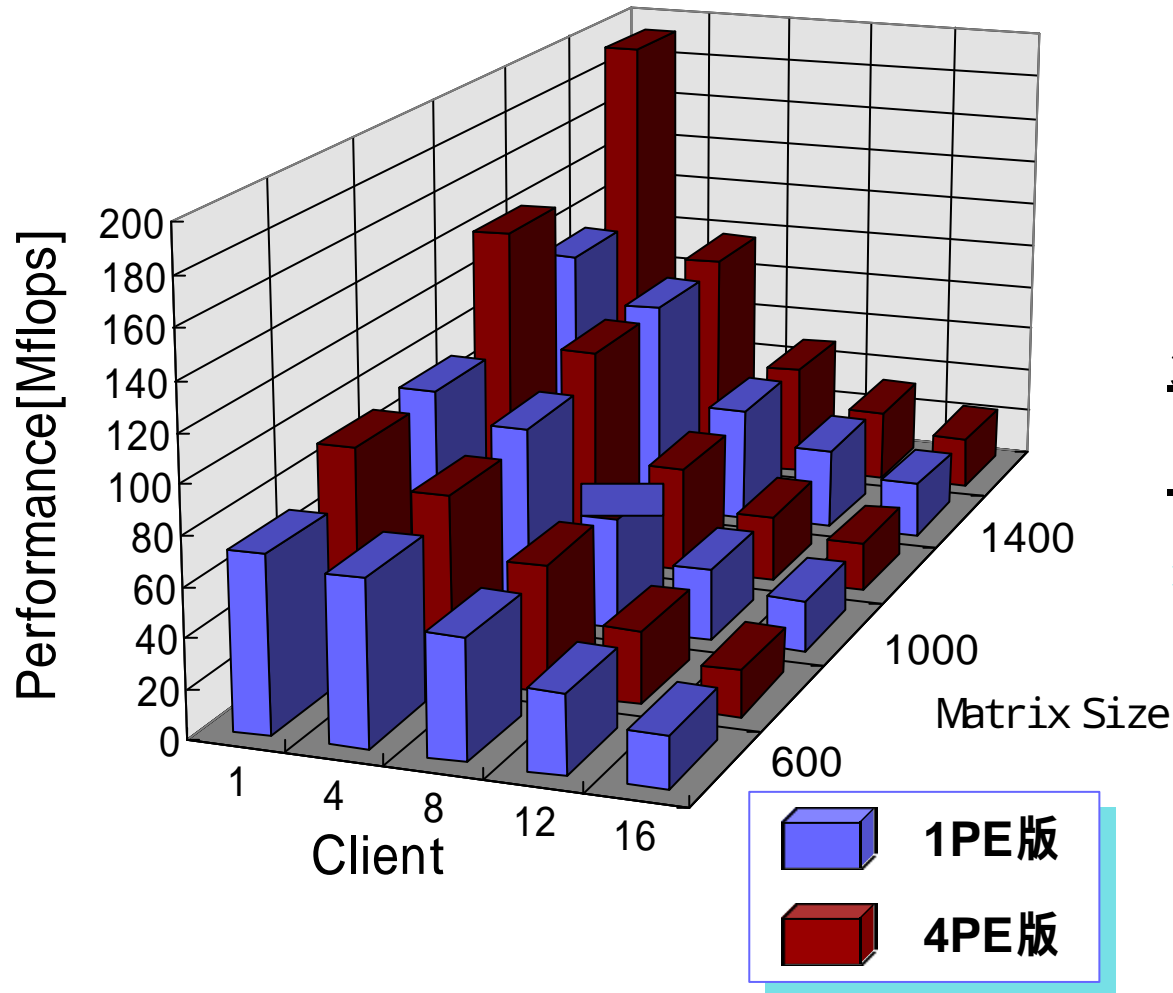


稼働率 [%]
 負荷平均値
 各クライアントのNinf_call
 経過時間 [sec]

	Performance[Mflops] max / min / mean	稼働率	負荷平均
1PE版	72.04 / 17.44 / 49.02	82.20	4.90
4PE版	89.35 / 24.31 / 51.31	92.00	9.98



各クライアントでの平均実行性能



➤ 1PE版に対する
4PE版の性能

クライアント数：

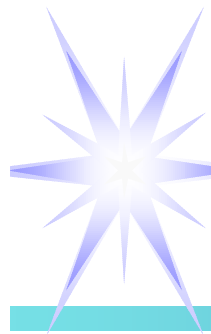
少ないとき 170 ~ 103 [%]

多いとき 105 ~ 88 [%]

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



4PE版libで有効運用



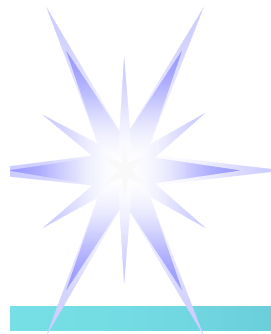
マルチクライアント環境での評価結果

- 問題サイズ1400のとき最大16クライアント，負荷平均値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

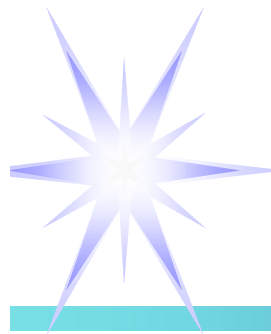
- Client数：1～8の閑散な状態 - 4PE版が絶対的に有利
：8～16の繁忙な状態 - 1PE版がわずかに有利
- 応答時間，待ち時間は4PE版が著しく低下しない

Cray J90 上で最高性能を実現する，最適化されたライブラリの方が効率がよい



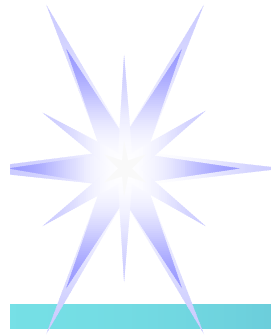
関連研究

- Remote Computation System (RCS) [ETH Zurich]
 - 複数のスーパーコンピュータを統一のインターフェースで利用するためのRPCを提供し，負荷分散に重点
 - PVM ベースで広域分散には適していない
- NetSolve [Univ. Tennessee]
 - Ninf_callに類似のAPIを提供し，Agentプロセスで負荷分散
 - クライアント側でもインターフェース情報を保持しなければならない
- Legion/Mentat [Univ. Virginia]
 - 独自の並列オブジェクト指向言語 Mentat を用い，複数の計算資源を統合した仮想計算機を実現
 - 最適化・機能追加が容易だが，既存システムとの連続性は維持できない
- メタコンピュータシステム [早稲田大]
 - 自動並列化Fortranコンパイラでサーバにコードを輸送し実行する
 - コードの安全性の保持，広域分散実行に適した粒度の抽出が難しい



本研究のまとめ

- Ninf システムで**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, マシン性能によって, **Ninf_callの実行性能は著しく低下しない**
- 超高性能計算機のOS上で**Ninf Serverがrobust**である
- 並列計算機の**高性能ライブラリがNinf Server上で有効利用**できる



今後の課題

- ▶ Linpack 以外の数値計算による検証
- ▶ Shortest Job First で Ninf Serverのジョブスケジューリングを行う
- ▶ メタサーバによる Network-Wide スケジューリング等を含めたNinf システムの有効性の検証のため、ネットワークシミュレータによる解析を行う