Ninf-G/MPIハイブリッドによる大規模一般化固有値問題の並列解法

小瀧義久[1]，櫻井鉄也[1,3]，梅田宏明[2,3]，
稲富雄一[2,3]，渡邊寿雄[2,3]，長嶋雲兵[2,3]

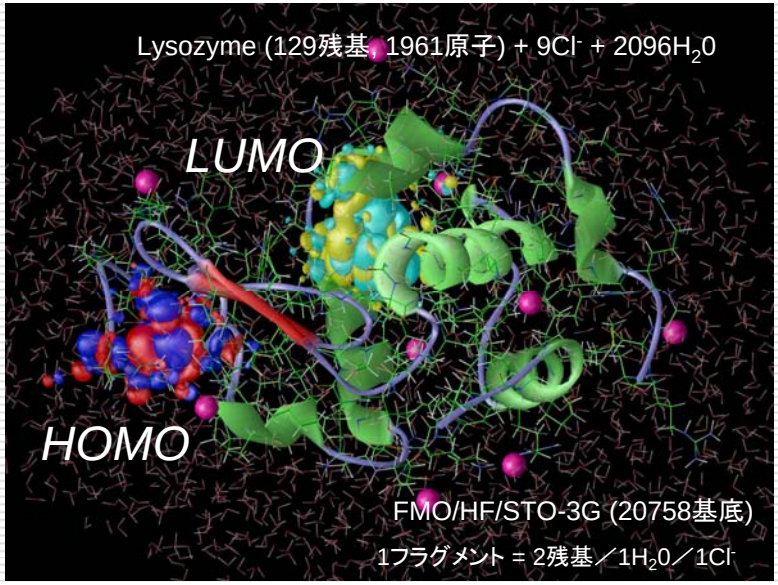
[1] 筑波大学，[2] 産業技術総合研究所，
[2] 科学技術振興機構

発表の構成

- 分子軌道計算
- 櫻井-杉浦法
- Ninf-Gによる並列化
- Ninf-G/MPIによる並列化
- まとめと今後の課題

分子軌道計算

- $FC=SC\varepsilon$
- 計算対象:リゾチーム
- リゾチームの行列
  - 次元数:20,758
  - 非零要素数:10,010,416
    (約2.3%, 約100MB,
    1.0e-7で疎行列化)



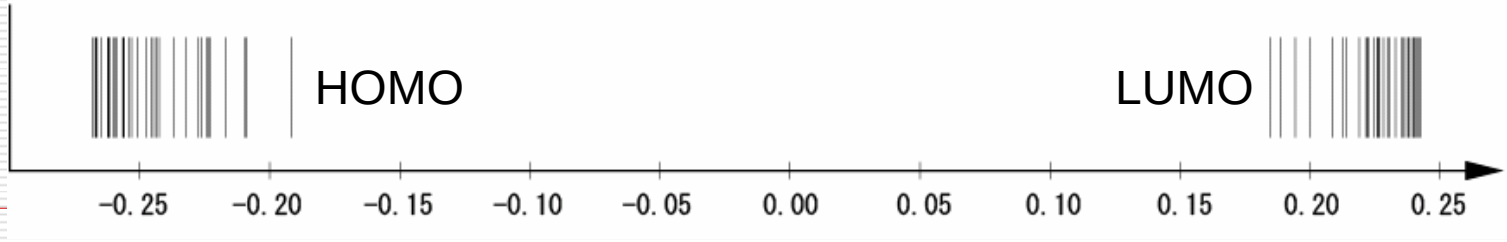
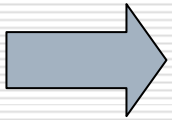


fig. リゾチームの0付近のエネルギー図.

一般化固有値問題の解法

- 従来法
  - Shift-Inverse + Arnoldiなど
  - リゾチームの計算には
    従来法(Householder + bisection)では逐次計算で約4時間
  - MPI化は既になされている（ライブラリが存在）
  - グリッド化は難しい

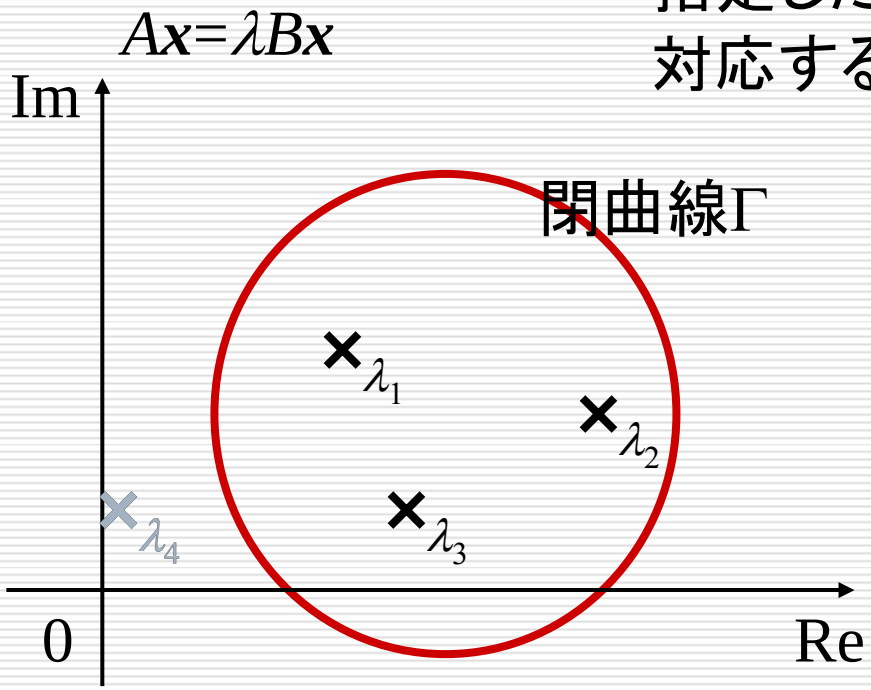


櫻井-杉浦法

  - グリッド環境に適した一般化固有値問題の解法

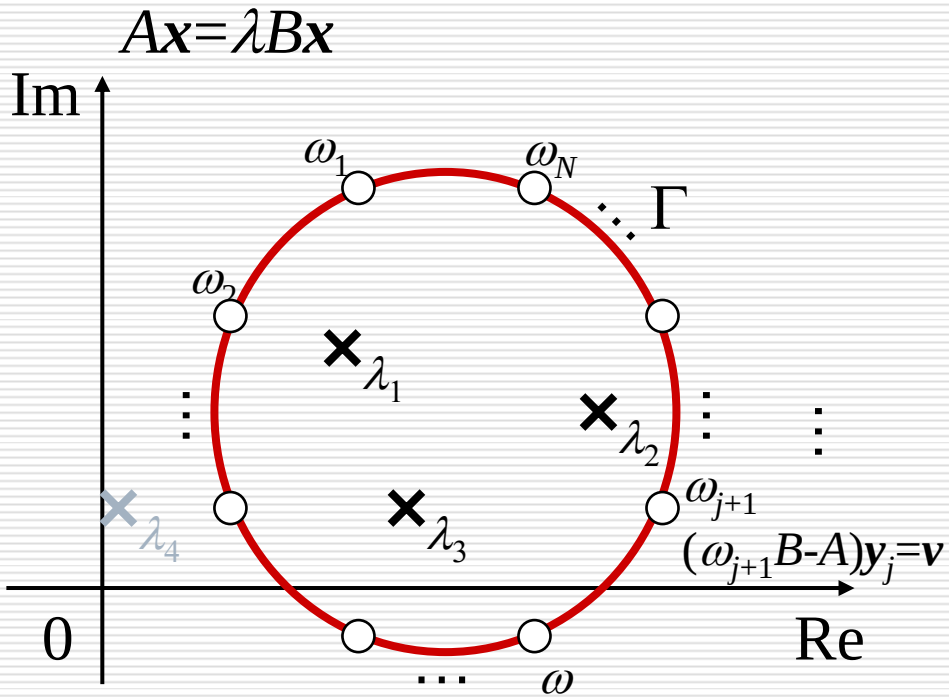
櫻井-杉浦法(1/4)

指定した閉曲線Γ内にある固有値及び
対応する固有ベクトルを求める方法



×：固有値λ

櫻井-杉浦法(2/4)



×：固有値λ

Γ上にN個の節点
ω_j $(j=1,N)$
を設定

↓

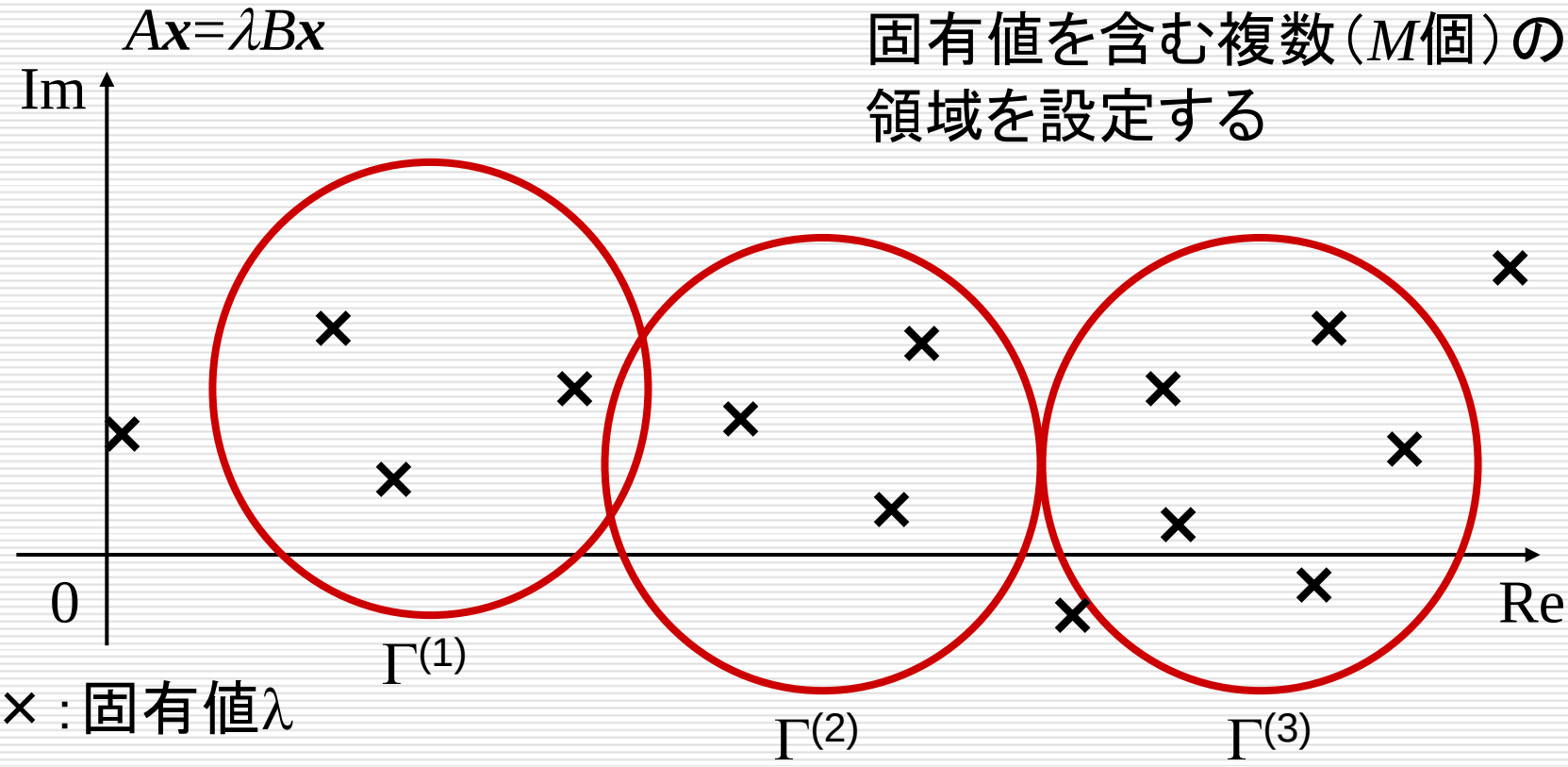
N個の連立一次方程式
$(\omega_jB-A)\boldsymbol{y}_j=\boldsymbol{v}$ $(j=1,N)$
を解く

↓

$\boldsymbol{y}_j(j=1,N)$を利用して，
領域内の固有値λ_i，
固有ベクトル$\boldsymbol{q}_i$を計算

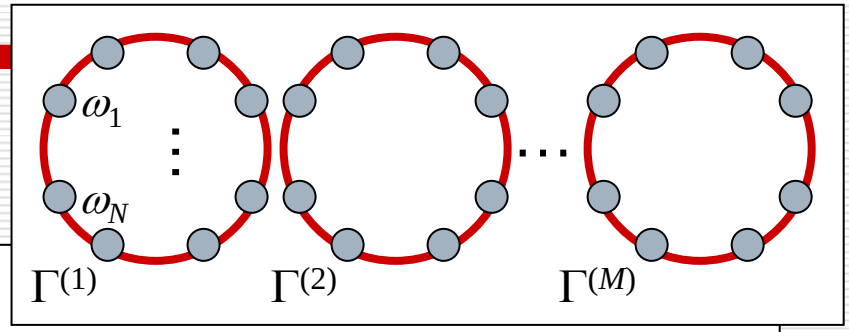
櫻井-杉浦法(3/4)

多くの固有値を求める場合は
固有値を含む複数（M個）の
領域を設定する



櫻井-杉浦法(4/4)

櫻井杉浦法の疑似コード



broadcast A, B

for(i=1, M){ /*領域に関する並列性*/

　for(j=1, N){ /*節点に関する並列性*/

　　$\boldsymbol{y}_j^{(i)}=(\omega_j^{(i)}B-A)^{-1}\boldsymbol{v}^{(i)}$

　}

　$\boldsymbol{\lambda}^{(i)}=\mathrm{f}(\boldsymbol{y}_j^{(i)})$

　$\boldsymbol{q}^{(i)}=\mathrm{g}(\boldsymbol{y}_j^{(i)})$

}

全ての連立一次方程式は
独立に解ける

▽

Ninf-G向きである

分子軌道計算への応用(1/2)

- 分子軌道計算では，固有値は実軸上に分布
- 必要な固有値は，0付近の数個
- y_jが実軸に対して対称なので
  上半分の節点のみ計算すればよい

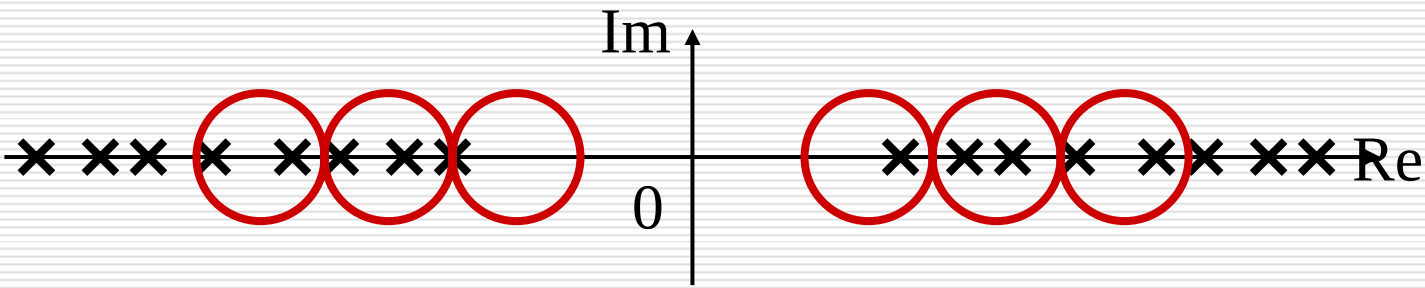


fig. 固有値の分布と領域設定の例.

分子軌道計算への応用(2/2)

- 領域数：32
  - −0.21〜−0.16（HOMO付近）に16領域
  - 0.16〜0.21（LUMO付近）に16領域
- 節点数/領域：32（うち上半分16個について計算）

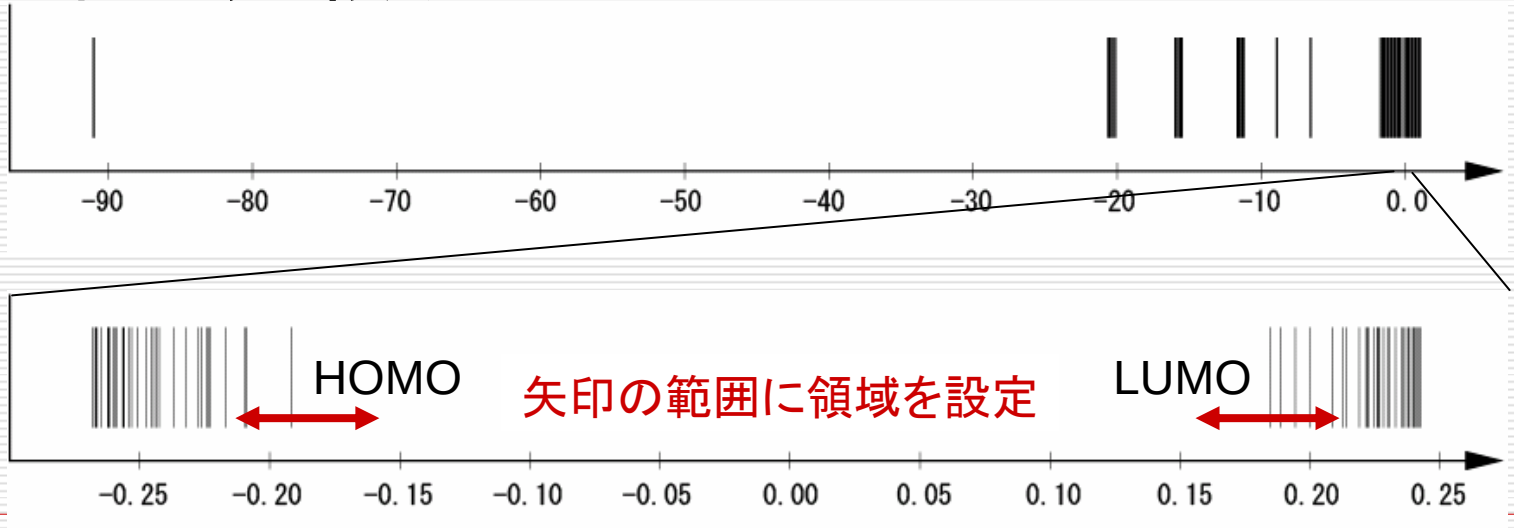


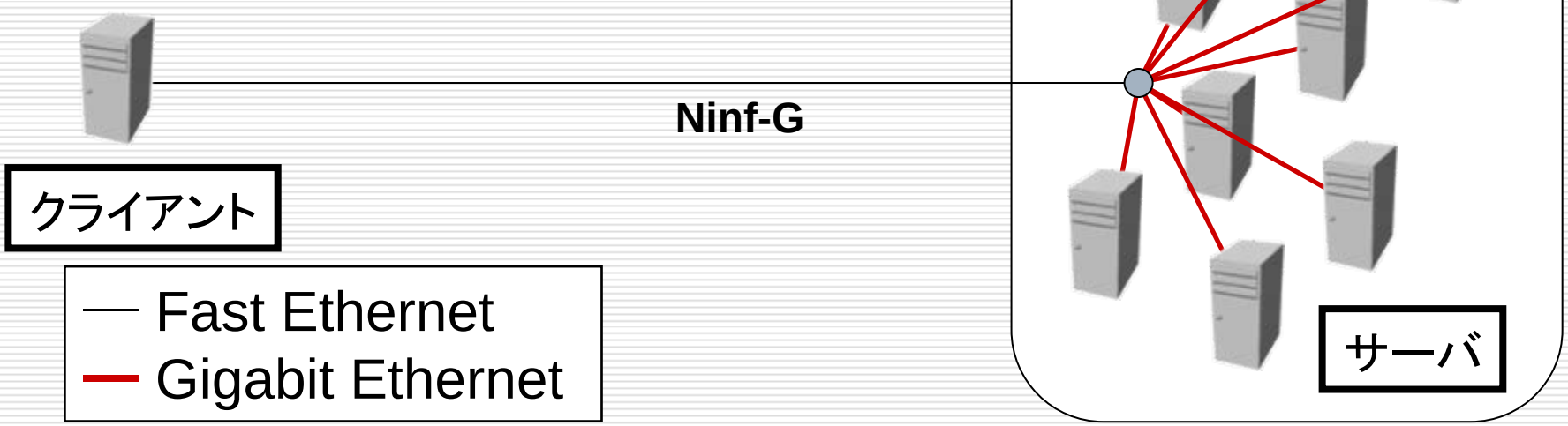
fig. リゾチーム全体，及び0付近のエネルギー図.

利用した計算機環境

- クライアント
  - Intel Pentium 4 CPU 3.00GHz / 2GB memory
  - Ninf-G ver. 2.4.0 / Globus ver. 3.2.1
- サーバ
  - AIST SuperCluster F-32
  - Intel Xeon CPU 3.06GHz Dual / 4GB memory * 260
  - Ninf-G ver. 2.4.0 / Globus ver. 3.3.3
- ネットワーク
  - クライアント-サーバ間：Fast Ethernet (100Mbps)
  - F-32クラスタ内：Gigabit Ethernet (1Gbps)

Ninf-Gによる並列化(1/3)

- Ninf-G単体での利用を想定
  - クライアントとサーバが一対一通信
  - Fast Ethernetで通信



Ninf-Gによる並列化(2/3)

櫻井杉浦法の疑似コード

broadcast A, B

for(i=1, M){ /*領域に関する並列性*/ — Ninf-Gで並列化

　for(j=1, N){ /*節点に関する並列性*/ — 並列化しない

　　$\boldsymbol{y}_j^{(i)}=(\omega_j^{(i)}B-A)^{-1}\boldsymbol{v}^{(i)}$

　}

　$\boldsymbol{\lambda}^{(i)}=\mathrm{f}(\boldsymbol{y}_j^{(i)})$

　$\boldsymbol{q}^{(i)}=\mathrm{g}(\boldsymbol{y}_j^{(i)})$

}

Ninf-Gによる並列化(3/3)

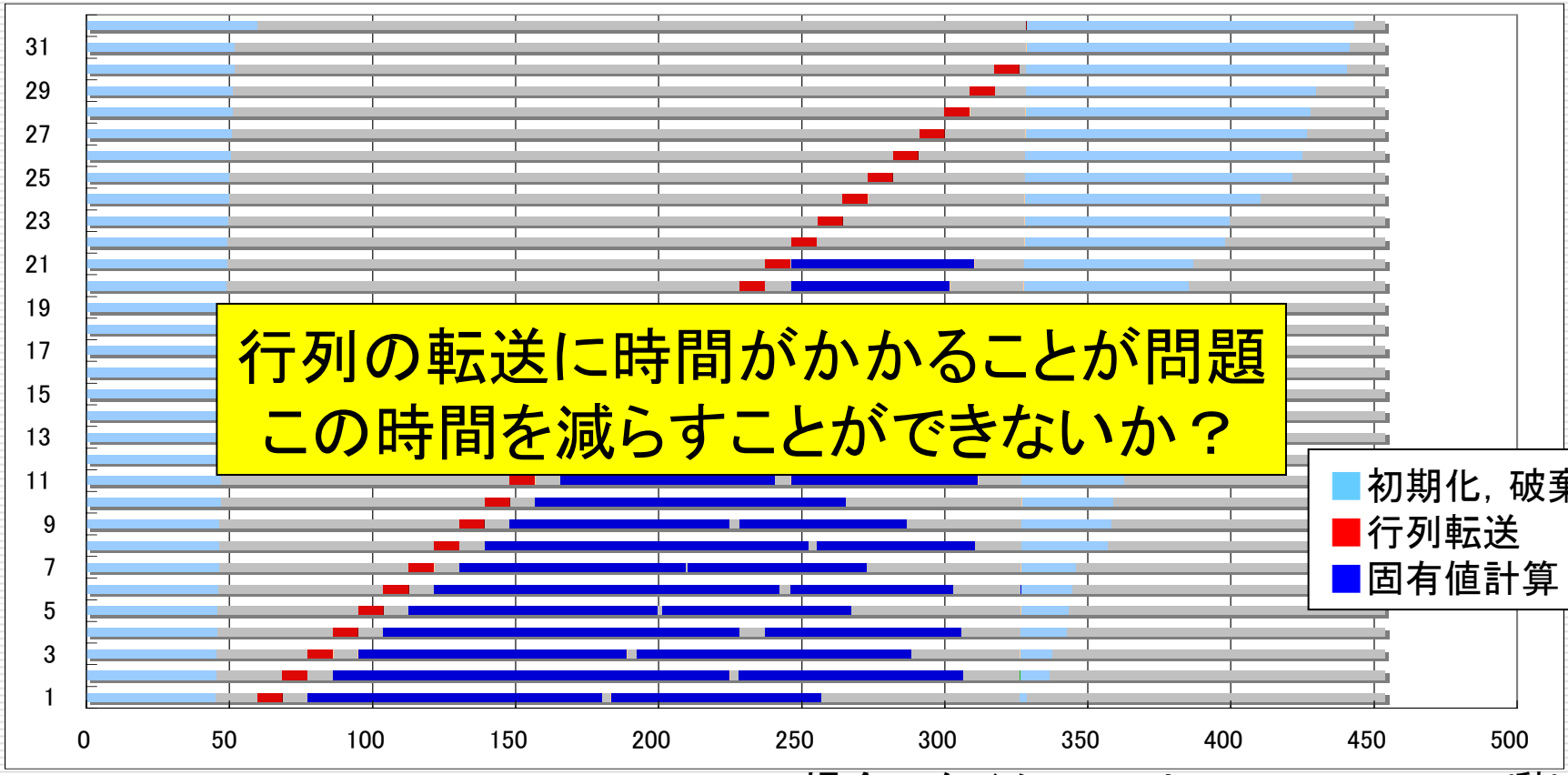
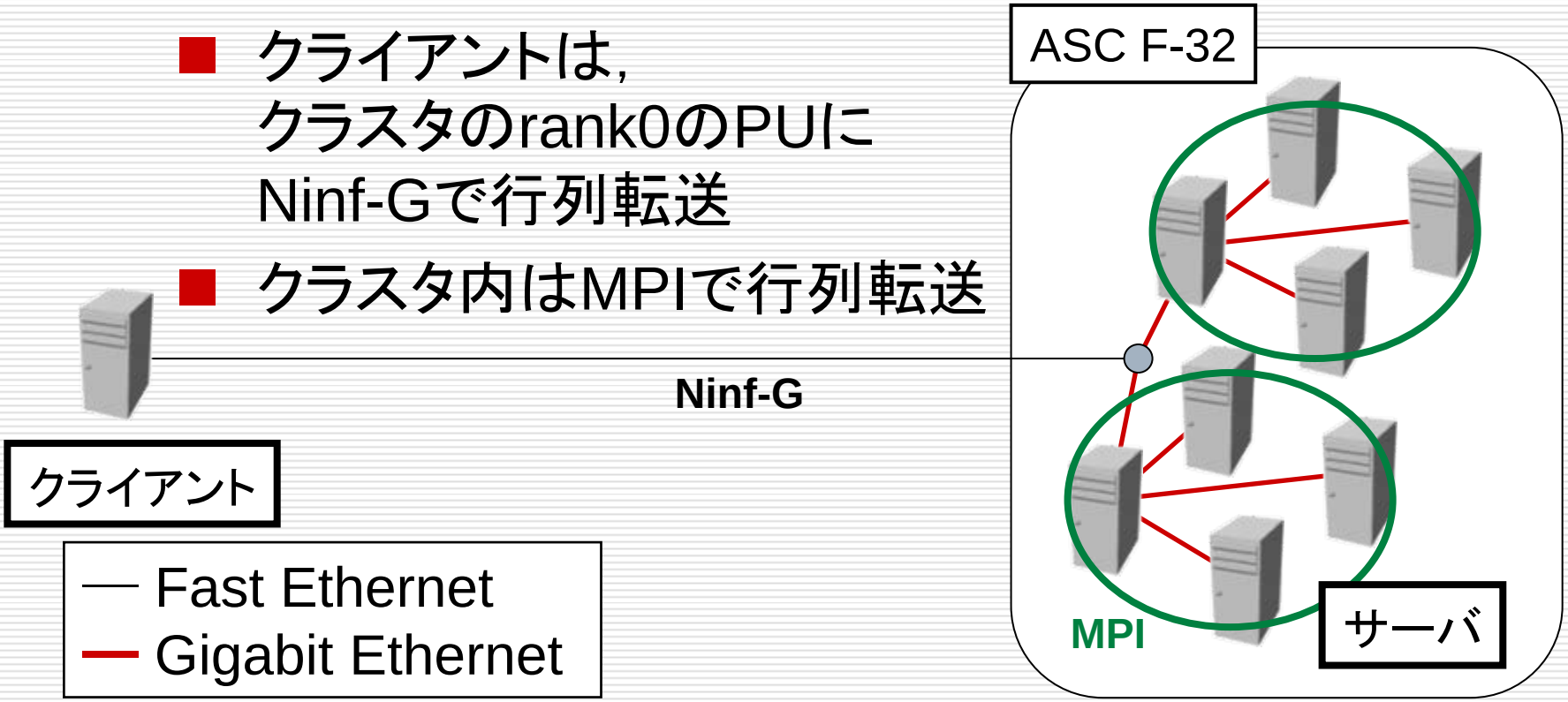


Fig.32PU(32handle)の場合のタイムチャート.

Ninf-G/MPIによる並列化(1/3)

- 小規模クラスタを組み合わせる環境を想定
  - クライアントは，クラスタのrank0のPUにNinf-Gで行列転送
  - クラスタ内はMPIで行列転送



Ninf-G/MPIによる並列化(2/3)

櫻井杉浦法の疑似コード

broadcast A, B

for(i=1, M){ /*領域に関する並列性*/ Ninf-Gで並列化

　for(j=1, N){ /*節点に関する並列性*/ MPIで並列化

　　$\boldsymbol{y}_j^{(i)}=(\omega_j^{(i)}B-A)^{-1}\boldsymbol{v}^{(i)}$

　}

　$\lambda^{(i)}=\mathrm{f}(\boldsymbol{y}_j^{(i)})$

　$\boldsymbol{q}^{(i)}=\mathrm{g}(\boldsymbol{y}_j^{(i)})$

}

Ninf-G/MPIによる並列化(3/3)

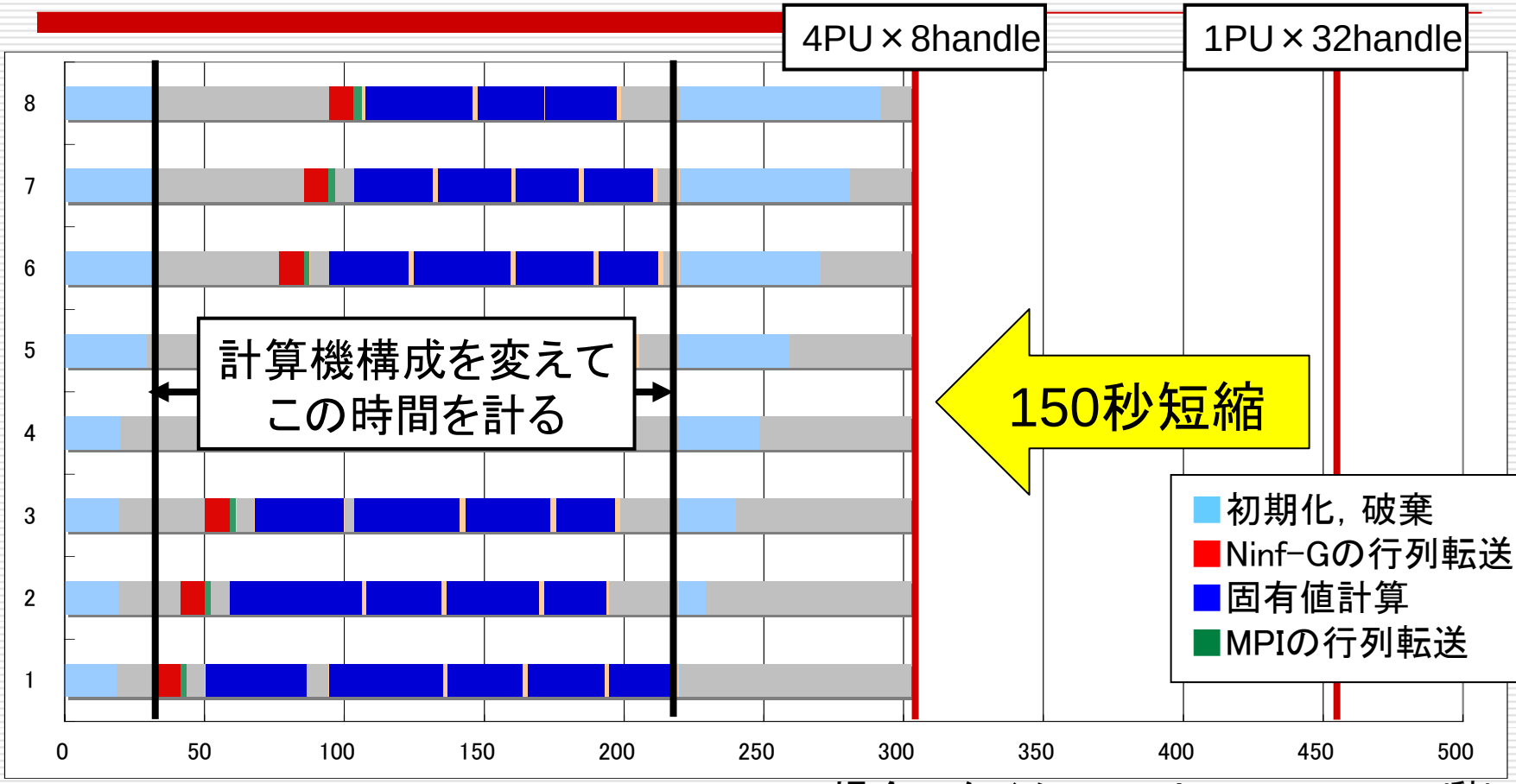


Fig.32PU(4PU×8handle)の場合のタイムチャート.

計算機構成と実行時間

Table 実行時間(秒)と速度向上率.

		# handles					
		1	2	4	8	16	32
	1	2672 (1.0)	1382 (1.9)	721 (3.7)	412 (6.5)	284 (9.4)	267 (10.0)
	2	1769 (1.5)	934 (2.9)	483 (5.5)	286 (9.3)	213 (12.5)	
	4	1028 (2.6)	536 (5.0)	284 (9.4)	186 (14.4)		
	8	621 (4.3)	337 (7.9)	190 (14.1)			
	16	447 (6.0)	245 (10.9)				

- 1PUの場合2672秒(従来法では約4時間)
- Ninf-Gのみの場合, 32PUで267秒
- Ninf-G/MPIの場合, 同PU数で186秒まで高速化

結論

- 櫻井杉浦法をNinf-Gを用いて実装した
  1PUで2672秒→32PUで267秒
- Ninf-G/MPIハイブリッドによって
  行列転送時間を短縮し，高速化した
  →32PUで186秒

今後の予定

- MPIの機能を活用し，
  連立一次方程式についても並列計算を行う

前処理について(1/2)

櫻井杉浦法の疑似コード

broadcast A, B

for(i=1, M){ /*領域に関する並列性*/ Ninf-Gで並列化

for(j=1, N){ /*節点に関する並列性*/ MPIで並列化

$P^{(i)} = h(A, B, i)$ /*前処理行列を生成*/

$\boldsymbol{y}_j^{(i)} = (\omega_j^{(i)} B - A)^{-1} \boldsymbol{v}^{(i)}$ /*反復解法に$P^{(i)}$を利用*/

}

$\lambda^{(i)} = \mathrm{f}(\boldsymbol{y}_j^{(i)})$

$\boldsymbol{q}^{(i)} = \mathrm{g}(\boldsymbol{y}_j^{(i)})$

}

前処理について(2/2)

櫻井杉浦法の疑似コード

broadcast A, B

for(i=1, M){ /*領域に関する並列性*/ Ninf-Gで並列化

$P^{(i)} = h(A,B,i)$ /*前処理行列を生成*/ ←MPIで並列化されない

for(j=1, N){ /*節点に関する並列性*/ MPIで並列化

$\boldsymbol{y}_j^{(i)} = (\omega_j^{(i)} B - A)^{-1} \boldsymbol{v}^{(i)}$ /*反復解法に$P^{(i)}$を利用*/

}

$\lambda^{(i)} = \mathrm{f}(\boldsymbol{y}_j^{(i)})$

$\boldsymbol{q}^{(i)} = \mathrm{g}(\boldsymbol{y}_j^{(i)})$

}

速度向上に関する考察2

Table 実行時間(秒)と速度向上率.

		# handles					
		1	2	4	8	16	32
# PU/ handle	1	2672 (1.0)	1382 (1.9)	721 (3.7)	412 (6.5)	284 (9.4)	267 (10.0)
	2	1769 (1.5)	934 (2.9)	483 (5.5)	286 (9.3)	213 (12.5)	223 (12.0)
	4	1028 (2.6)	536 (5.0)	284 (9.4)	186 (14.4)	160 (16.7)	170 (15.7)
	8	621 (4.3)	337 (7.9)	190 (14.1)	136 (19.7)	144 (18.6)	
	16	447 (6.0)	245 (10.9)	145 (18.4)	112 (23.9)		

- 最速で，128CPUを使用し，112秒まで高速化
- 同CPU数の中では，16PU×8handleのものが最も速い

同CPU数で最速の組み合わせ

Table 実行時間(秒)と速度向上率.

		# handles					
		1	2	4	8	16	32
# PU/ handle	1	2672 (1.0)	1382 (1.9)	721 (3.7)	412 (6.5)	284 (9.4)	267 (10.0)
	2	1769 (1.5)	934 (2.9)	483 (5.5)	286 (9.3)	213 (12.5)	223 (12.0)
	4	1028 (2.6)	536 (5.0)	284 (9.4)	186 (14.4)	160 (16.7)	170 (15.7)
	8	621 (4.3)	337 (7.9)	190 (14.1)	136 (19.7)	144 (18.6)	
	16	447 (6.0)	245 (10.9)	145 (18.4)	112 (23.9)		

連立一次方程式について

- ソルバー：前処理付きCOCG法
  - 収束判定：1.0e-8
  - 最大反復回数：300
- 前処理：複素シフト付き不完全コレスキー分解
  - シフト量：0.02
  - ドロップ値：0.01

櫻井-杉浦法のパラメータ

- 小規模固有値問題の次元数：12
- ゴーストの判定：1.0e-6

FMO-MOについて