Empowered by Innovation NEC

NIMSナノシミュレーションワークショップ2012

PHASEの性能最適化

2012年10月29日

日本電気株式会社
加藤　季広

はじめに

- PHASEは、様々なプラットフォームで高い性能が得られるよう、性能最適化が適用されています
  - 対応プラットフォーム
    - Windows、PC-Linux (X86)
    - SR (POWER)
    - 京コンピュータ、FX10 (SPARC)
    - 地球シミュレータ、NEC SXシリーズ
- ここでは、PHASEに適用されている性能最適化について、その概要を説明させていただきます

© NEC Corporation 2011

旧地球シミュレータ向け性能最適化

- ベクトル長(主に最内ループ長)がなるべく長くなるようにする
- MPIによる分散メモリ並列化と共有メモリ並列化(SMP)のハイブリッド並列を適用
  - 並列化軸が共通の場合もある
  - 波動関数の固有状態を分割
  - 波動関数のFFTは局所的に実行(非並列)
- ファイル入出力も並列化
- 大規模Si系(N=10,648個)を対象とする
  - 非局所ポテンシャルの射影演算子 N_p = 42,592 = 4N
  - 波動関数の基底平面波 M = 792,555
  - 状態数 N_e = 24,576 ≒ 2N
    - MPI/SMP並列化するのに十分な数
  - FFTメッシュ数 300x300x300

© NEC Corporation 2011

負荷の高い処理

O(N^2M)の項

- **非局所ポテンシャルと波動関数の積を作る** $V_{NL}|\Psi_{k\nu}\rangle = \sum_I \sum_{n,m} D_{nm}^{\zeta(I)} |\beta_n^I\rangle\langle\beta_m^I|\Psi_{k\nu}\rangle$
  - **射影演算子と波動関数の内積を作る** $\langle\beta_m^I|\Psi_{k\nu}\rangle \equiv f_{mk\nu}^I$
  - **非局所ポテンシャルと波動関数の積を完成する** $V_{NL}|\Psi_{k\nu}\rangle = \sum_I \sum_{n,m} D_{nm}^{\zeta(I)} |\beta_n^I\rangle f_{mk\nu}^I$
- **波動関数の規格直交化(修正グラムシュミット法)** $\langle\Psi_{k\mu}|S|\Psi_{k\nu}\rangle = \delta_{\mu\nu}$
- Subspace-Diagonalization(**波動関数のユニタリ変換**) $\langle\Psi_{k\mu}|H|\Psi_{k\nu}\rangle = \varepsilon_{k\mu}\delta_{\mu\nu}$

O(NMlogM)の項

- **波動関数の**FFT $\Psi_{k\nu}(\mathbf{G}) \xrightarrow{\text{FFT}} \Psi_{k\nu}(\mathbf{r})$
  - **局所ポテンシャルと波動関数の内積**
  - **電荷密度分布の構成(欠損電荷に由来しない項)**

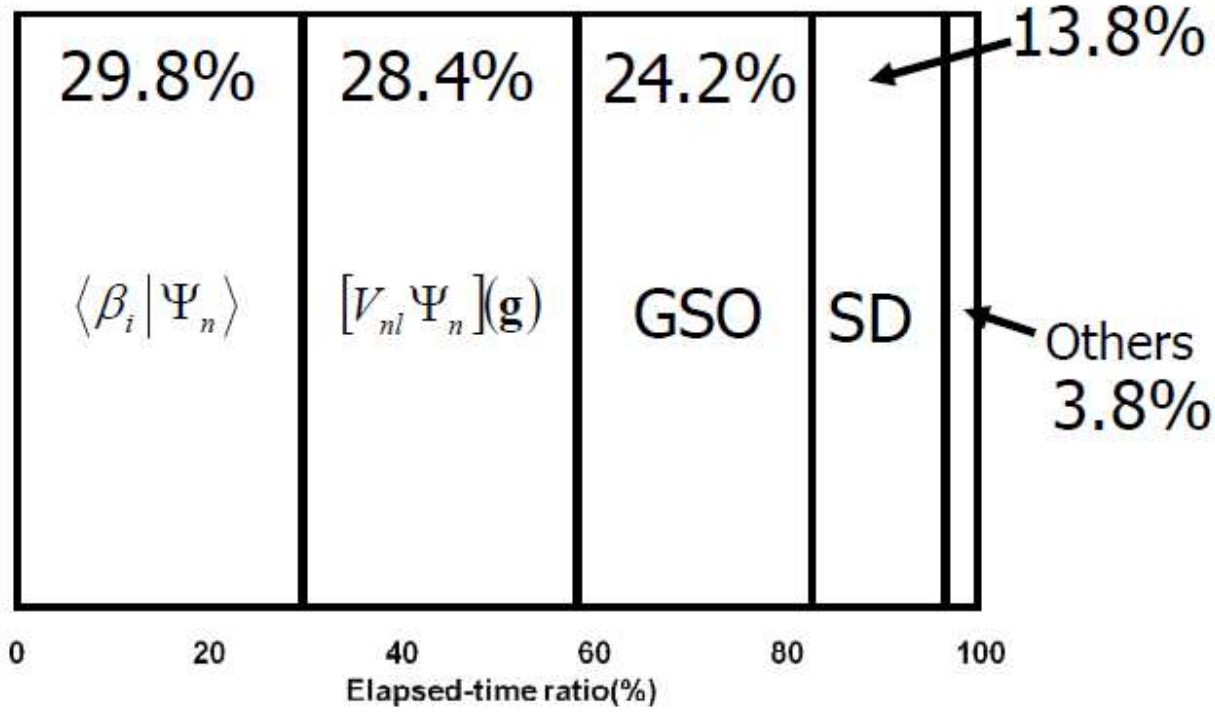
O(NM)の項

- **電荷密度の欠損電荷に由来する項(**hardpart**と呼ぶ)**
- … $\rho_H(\mathbf{G}) = \sum_I \sum_{\tau\ell m} \sum_{\tau'\ell'm'} h^I_{\tau\ell m,\tau'\ell'm'} e^{-i\mathbf{G}\cdot\mathbf{R}_I} \sum_{\ell''} i^{-\ell''} Q^{\zeta(I),\ell''}_{\tau\ell\tau'\ell'}(|\mathbf{G}|) d^{\ell''}_{lm,l'm'} Y_{\ell''m''}(\hat{\mathbf{G}})$

© NEC Corporation 2011

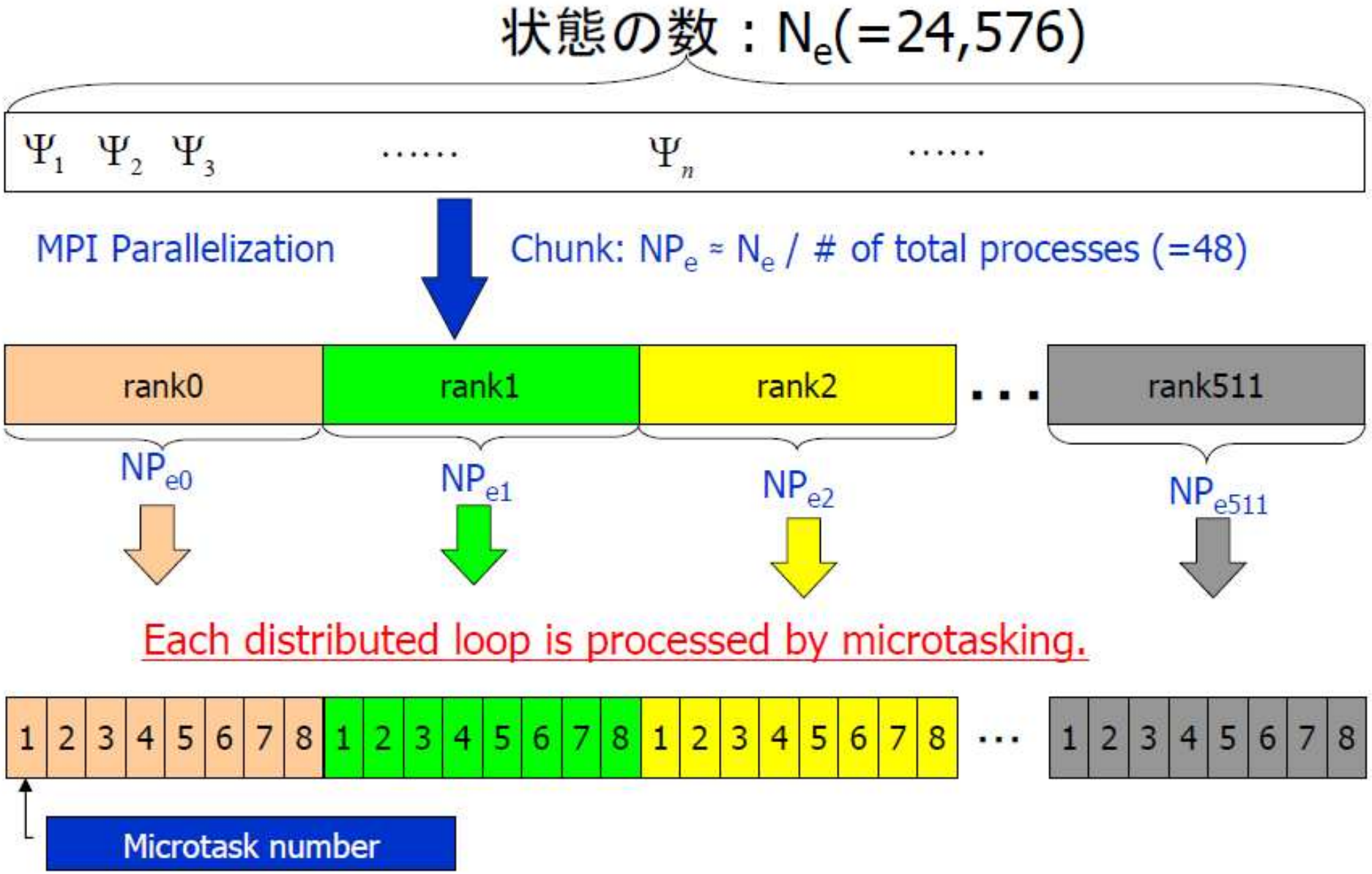
Si 10,000原子系の負荷分布

- **射影演算子と波動関数の内積**
- **非局所ポテンシャルと波動関数の積を完成する**
- **波動関数の規格直交化**(GSO)
- **部分対角化**=Subspace-Diagonalization (SD)
- FFT等



© NEC Corporation 2011

波動関数の各状態を分割

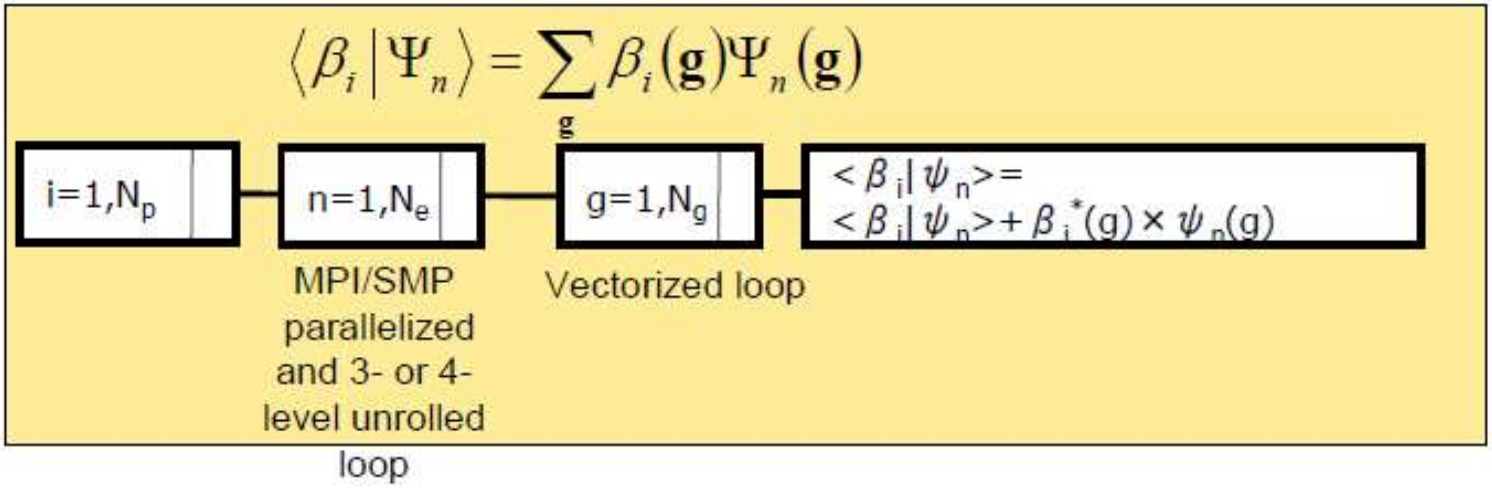


© NEC Corporation 2011

非局所ポテンシャルと波動関数の積

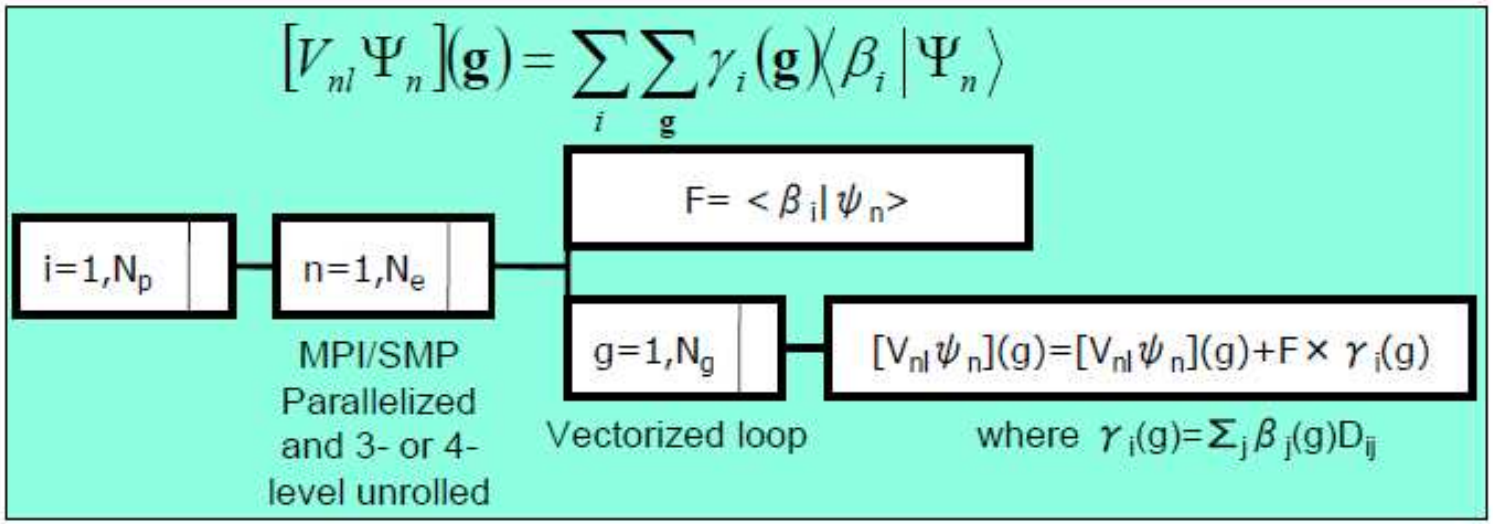
射影演算子と波動関数の内積

$$\langle \beta_i | \Psi_n \rangle = \sum_{\mathbf{g}} \beta_i(\mathbf{g}) \Psi_n(\mathbf{g})$$



非局所ポテンシャルと波動関数の積を完成する

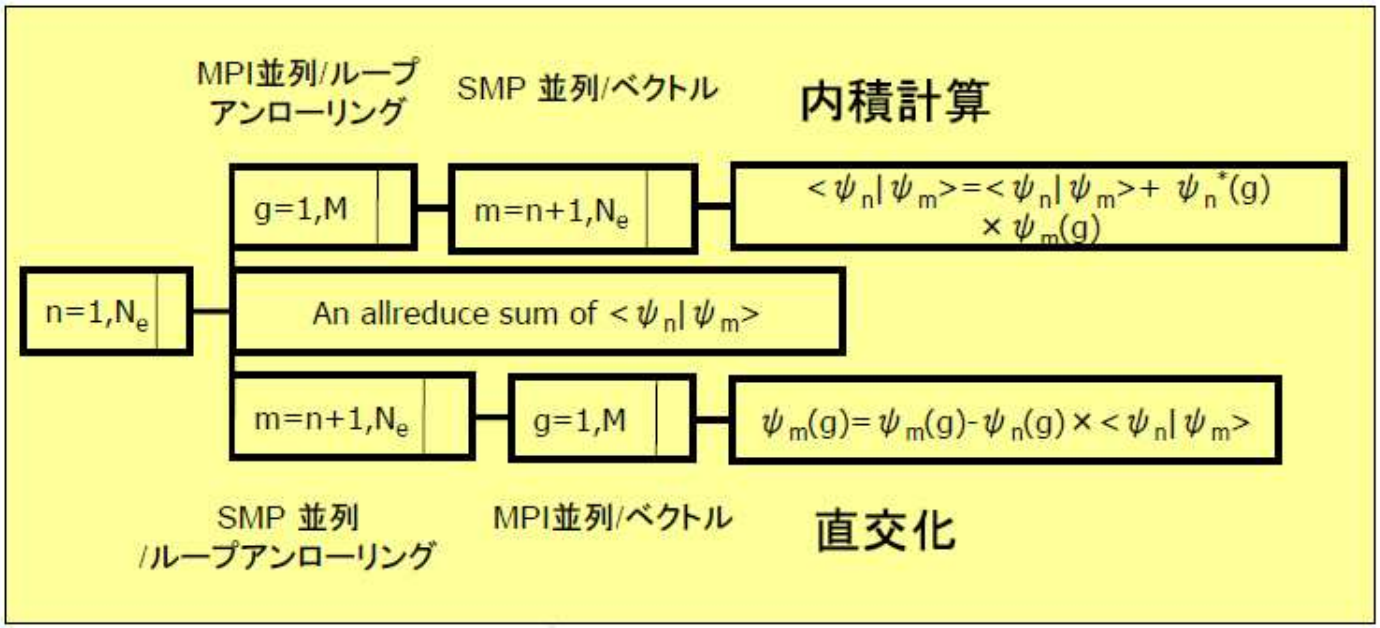
$$[V_{nl}\Psi_n](\mathbf{g}) = \sum_i \sum_{\mathbf{g}} \gamma_i(\mathbf{g}) \langle \beta_i | \Psi_n \rangle$$

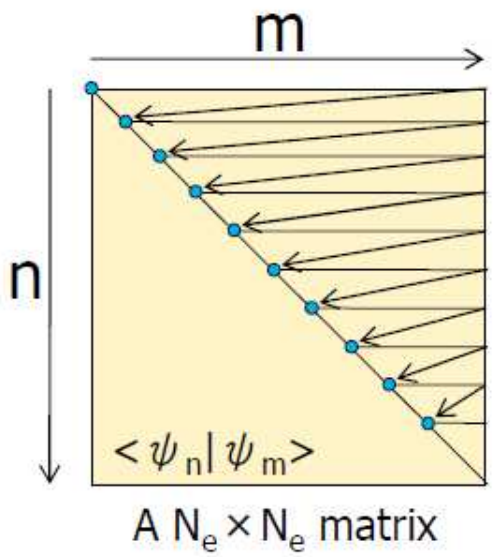


© NEC Corporation 2011

波動関数の規格直交化(GSO)

アルゴリズム





通信量の比較

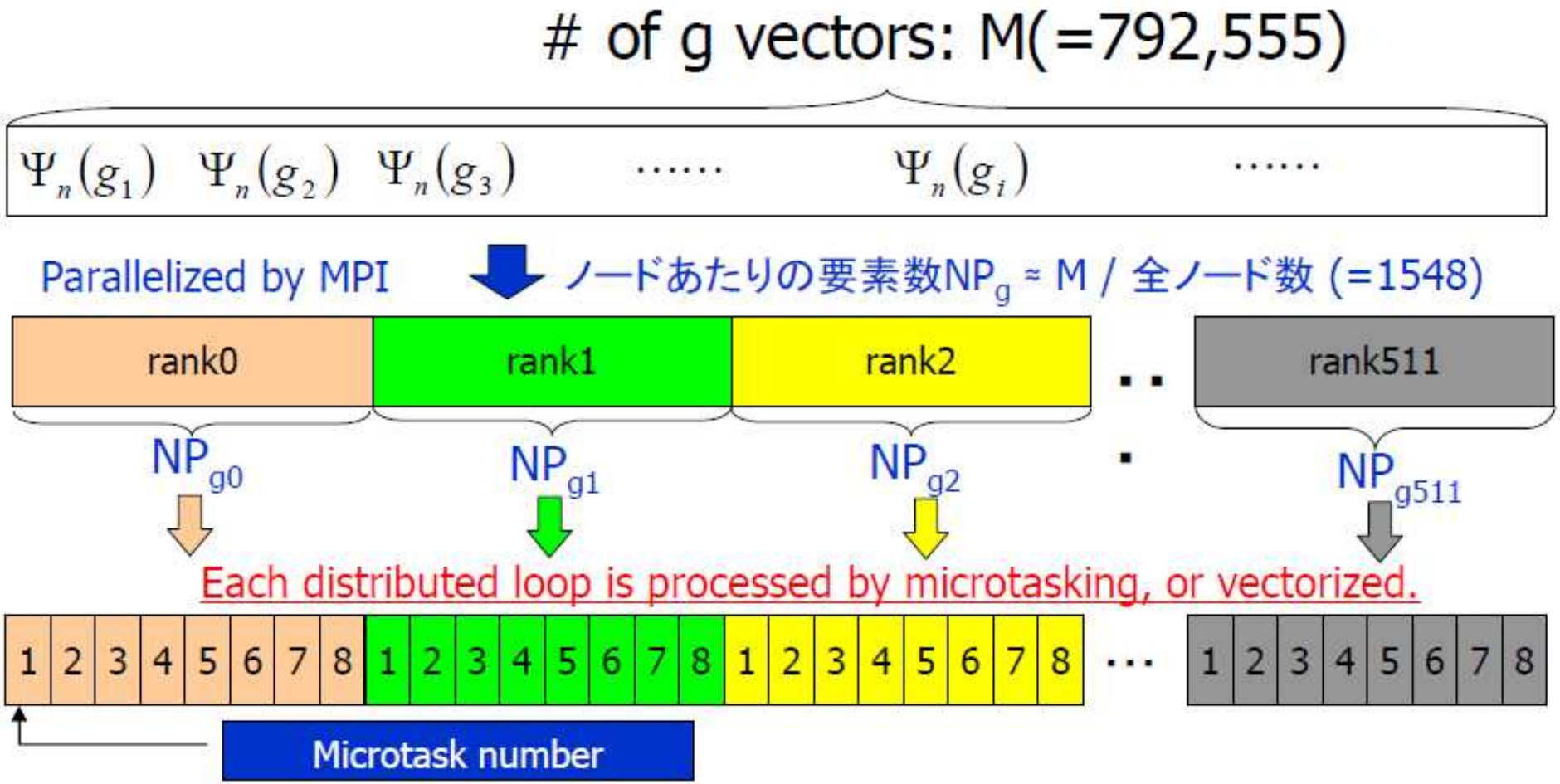
状態を分割軸とする場合	平面波成分を分割軸とする場合
$M_{mpi}^{state} = \sim N_e*M*N_{node}$ (broadcast of Ψ)	$M_{MPI}^{G} = N_e*N_e*N_{node}*log_2N_{node}/2$ (Allreduce) $+2*N_e*N_g$ (転置転送) $\approx N_e^2N_{node}log_2N_{node}/2$

$M_{mpi}^{G}/M_{mpi}^{state} < 3/20$ (assuming $M \approx 30N_e$, $N_{node}=512=2^9$)

© NEC Corporation 2011
Empowered by Innovation NEC

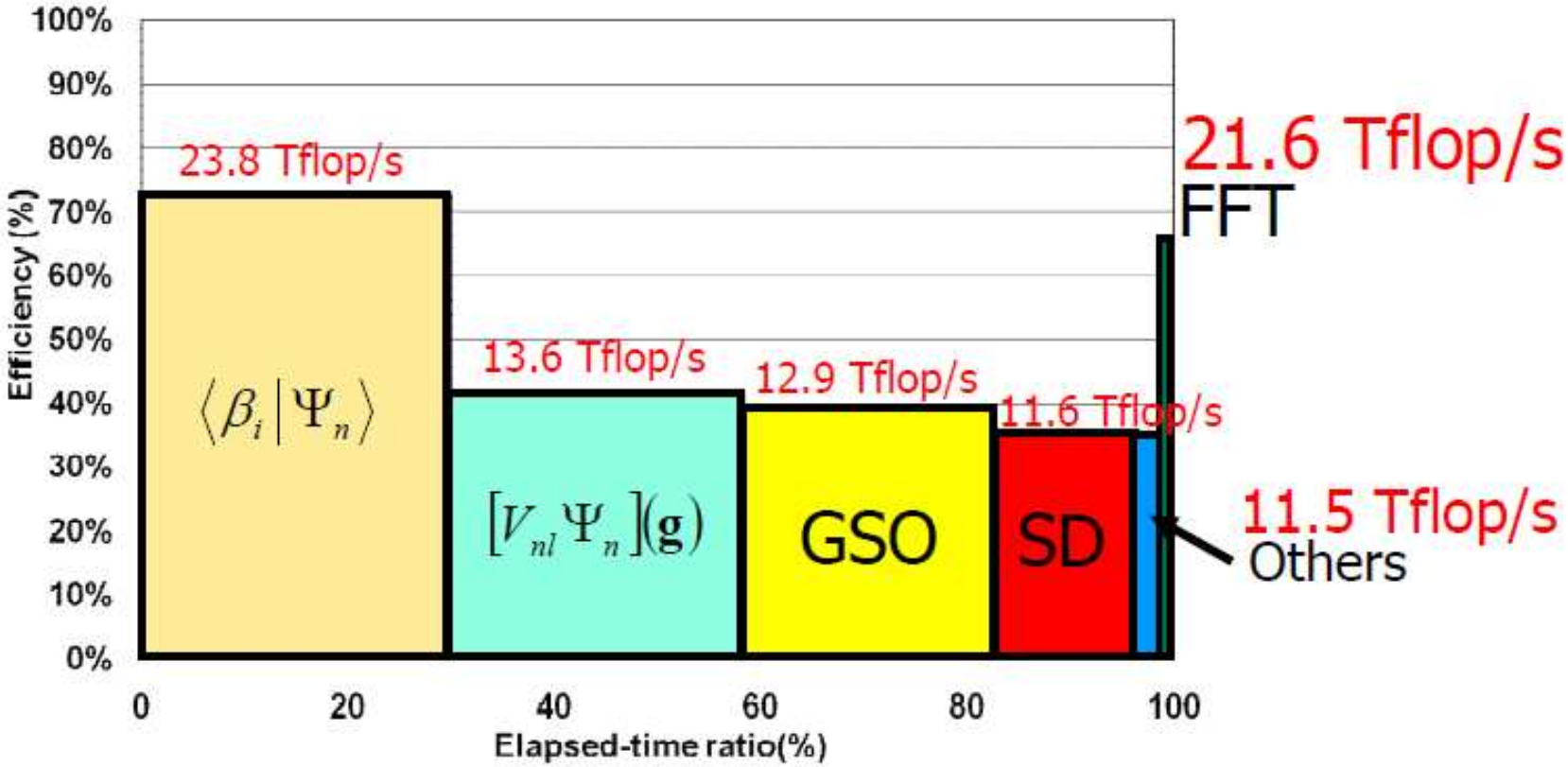
波動関数の成分(平面波)で分割

状態に関して分割された波動関数を転置転送して成分分割する



© NEC Corporation 2011

チューニングしたあとの効率



部分対角化(SD)は電子状態の収束性を加速するために用いる

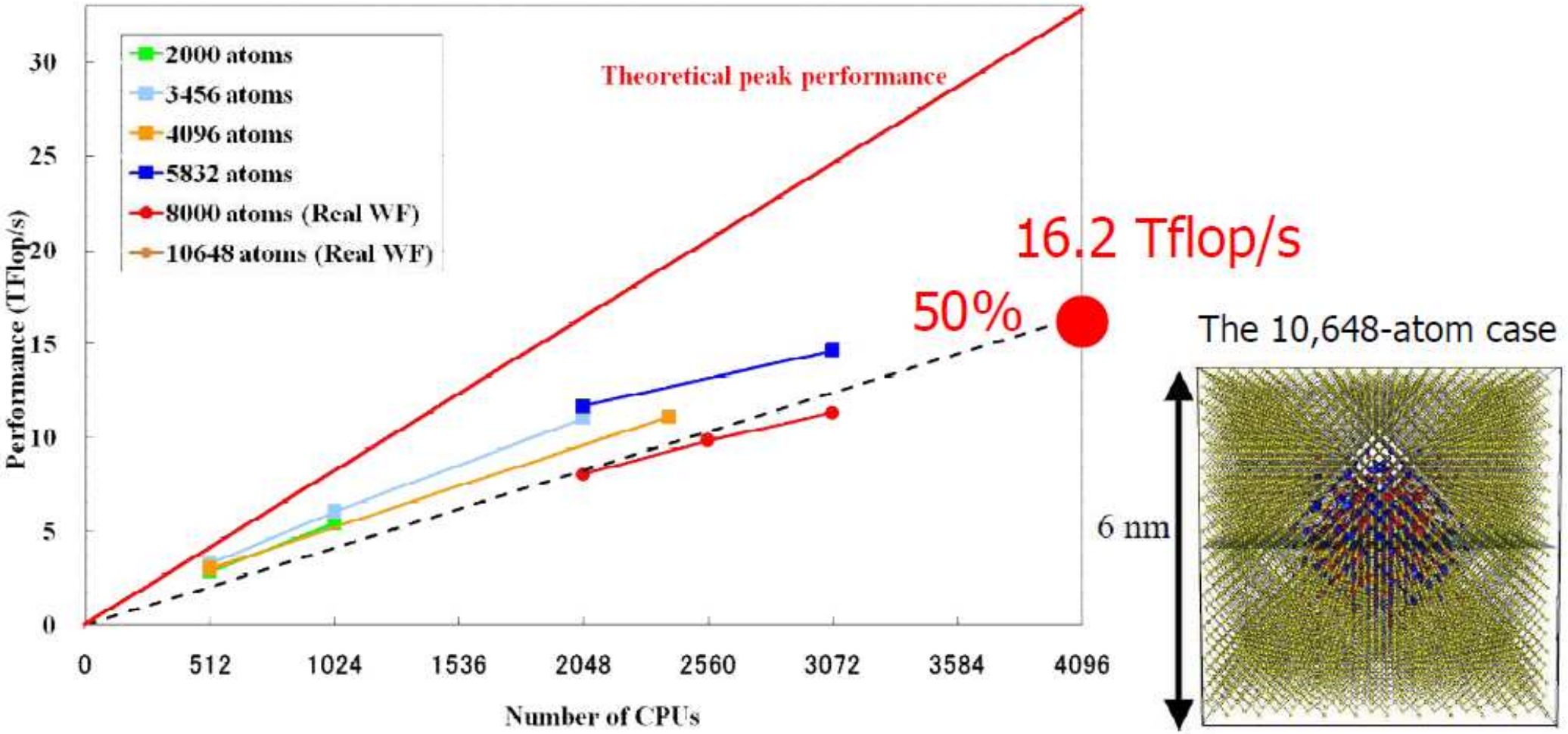
- 部分行列の計算はGSOと同様成分をMPI分割軸とする

$$\langle \Psi_m | H_{KS} | \Psi_n \rangle = \sum_{\mathbf{g}} \Psi_m^*(\mathbf{g}) [H_{KS}\Psi_n](\mathbf{g})$$

- 行列の対角化はScaLAPACKサブルーチン(PDSYEVD or PZHEEVD)を用いる

© NEC Corporation 2011

Total Performances



© NEC Corporation 2011

新地球シミュレータ(SX-9)向けの性能最適化

主な対象はO(N^3)部分

- 非局所ポテンシャルと波動関数の積を作る部分(3重ループ)
  - 射影演算子と波動関数の内積 $\langle \beta_m^I | \Psi_{kv} \rangle \equiv f_{mkv}^I$
  - 非局所ポテンシャルと波動関数の積 $V_{NL} | \Psi_{kv} \rangle = \sum_I \sum_{n,m} D_{nm}^{\sigma(I)} | \beta_n^I \rangle f_{mkv}^I$

BLAS化

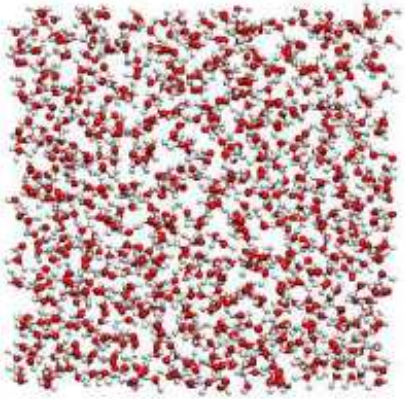
- オリジナルのループ構造
  - 内側2重ループで実装: 行列ベクトル積(Level2 BLAS)
- ループ分割を行ってデータを蓄積し、行列積の形に最適化
  - 修正後の実装: 行列積(Level3 BLAS)
  - 核心部でDGEMMを使用
- 最内ループをブロッキング
  - ブロックサイズを工夫し、メモリ使用量を削減

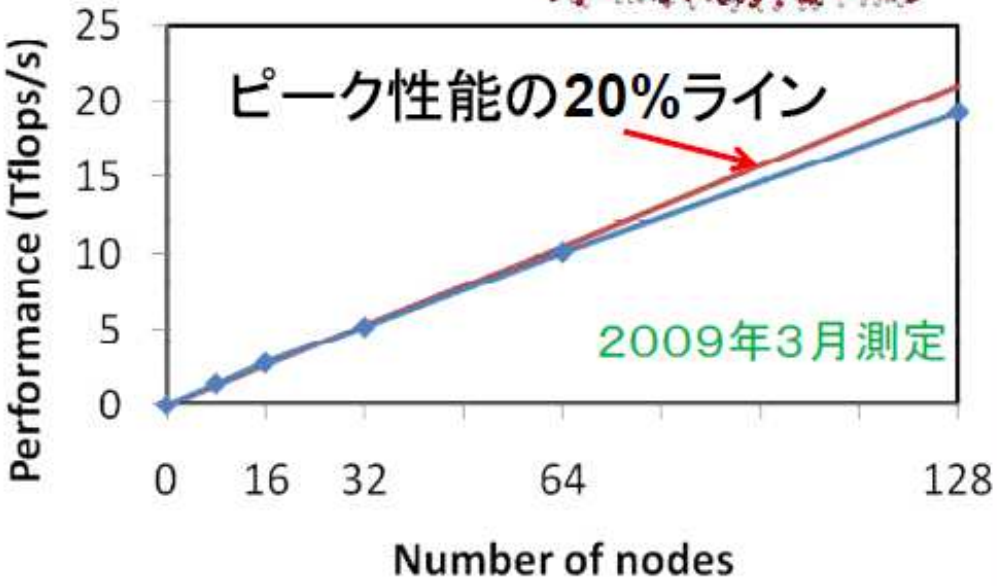
FFT(交換相関ポテンシャルを計算する部分)を並列化

© NEC Corporation 2011

新地球シミュレータにおけるPHASEの性能（性能最適化前）

水
864分子





地球シミュレータ(ES2)

計算プロセッサのピーク性能	102.4Gflops	総プロセッサ数	1280
計算ノードのピーク性能	819.2Gflops	総計算ノード数	160
計算ノードの主記憶容量	128GByte	ピーク性能	131Tflops
計算ノードのCPU数	8	主記憶容量	20TByte

●プログラムチューニング
計算コアのBLASサブルーチン置換により
2～3倍程度の性能向上が期待できる。

© NEC Corporation 2011

射影演算子と波動関数の内積

$$\left\langle \beta_m^I \middle| \Psi_{\mathbf{k}\nu} \right\rangle \equiv f_{m\mathbf{k}\nu}^I$$

オリジナルのループ構造

```
line
 100        subroutine m_ES_betar_dot_WFs_4_each_k(nfout,ik)
  :                    :
  :           call tstatc0_begin('betar_dot_WFs ',id_sname)
  :                    :
 200 +------>    do ia = 1, natm
 300 |  +--->      do i = 1, iba(ik)                          ┐
 400 |  |           nbase(i), ngabc(nbase(i))                 ├→ zfcos(i), zfsin(i)
 500 |  +---       enddo                                      ┘
 700 |  +----->    do lmt1 = 1, ilmt(it)
 800 |  |  +--->     do i = 1, iba(ik)                        ┐
 900 |  |  |           zfcos(i), zfsin(i), snl_or_snld(i)     ├→ ar(i), ai(i)
 000 |  |  +---      enddo                                    ┘
1200 |  |  +--->     do ib = 1, np_e                          ┐
1300 |  |  |  +->      do i = 2, iba(ik)                     ├→ bpr_l(ib,lmta1)
1400 |  |  |  |           ar(i), ai(i), psi_l(i,ib)          ┘
1500 |  |  |  +---      enddo
1600 |  |  +---      enddo
1700 |  +---------   enddo
1800 +------------ end do
  :
1900           call tstatc0_end(id_sname)
2000        end subroutine m_ES_betar_dot_WFs_4_each_k
```

行列ベクトル積

修正後

```
Line
 100        subroutine m_ES_betar_dot_WFs_4_each_k(nfout,ik)
 200 +----->    do ia = 1, natm
 300 |  +--->      do lmt1 = 1, ilmt(it)
                    処理の個数をカウント(icnt)
                    リスト作成
 400 |  +---       enddo
 500 +------      enddo
 300 +----->    do ibl1=1,iba(ik),ibsize                    ループをブロッキング
 400 |            ibl2=min( ibl1+ibsize-1,iba(ik) )
 200 |  +----->    do ia = 1, natm
 300 |  |  +--->     do i = ibl1, ibl2                      ┐
 400 |  |  |           nbase(i), ngabc( nbase(i) )           ├→ zfcos(i,ia),zfsin(i,ia)
 500 |  |  +---      enddo                                  ┘
 550 |  |  +---      enddo
 750 |  |  +----->   do n = 1, icnt
 800 |  |  |  +--->    do i = ibl1, ibl2                    ┐
 900 |  |  |  |           zfcos(i,n), zfsin(i,n)            ├→  ar(i,n), ai(i,n)
     |  |  |  |           snl_or_snld(i)                    ┘
1000 |  |  |  +---     enddo
1150 |  |  +---------  enddo
1150 |  |  +----->   do n = 1, icnt                          betar_dot_WFs_core_blk
1200 |  |  |  +--->    do ib = 1, np_e
1300 |  |  |  |  +->     do i = ibl1, ibl2                  ┐
1400 |  |  |  |  |          ar(i,n), ai(i,n), psi_l(i,ib)   ├→ bpr_l(ib,n)
1500 |  |  |  |  +---     enddo                             ┘
1600 |  |  |  +---     enddo
1800 |  |  +---------  end do                                行列行列積に
     |  +------------ enddo
2000           end subroutine m_ES_betar_dot_WFs_4_each_k
```


© NEC Corporation 2011

射影演算子と波動関数の内積: 核心部

```
subroutine betar_dot_WFs_core_blk( ibsize, icnt1, tran1,  psi_l )
    integer,intent(in)  :: ibsize, icnt1
    logical, intent(in)  :: tran1
    real(kind=DP),intent(in), dimension(kg1,np_e,ista_k:iend_k,kimg) :: psi_l  ! MPI
    integer       :: j, ib, i
    integer       :: icsize
    real(kind=DP) :: alpha, beta

    if(kimg == 1) then
     icsize = ibl2-ibl1+1
       alpha= 1.d0; beta= 1.d0
       call DGEMM__('T', 'N', np_e,icnt1,icsize, alpha,psi_l(ibl1,1,ik,1),kg1, wk_ar1,ibsize, beta,bp_tmp1,LD11)
       call DGEMM__('T', 'N', np_e,icnt1,icsize, alpha,psi_l(ibl1,1,ik,1),kg1, wk_ai1,ibsize, beta,bp_tmp2,LD21)
     else if(kimg == 2) then
       icsize = ibl2-ibl1+1
       alpha= 1.d0; beta= 1.d0
       call DGEMM__('T', 'N', np_e,icnt1,icsize, alpha,psi_l(ibl1,1,ik,1),kg1, wk_ar1,ibsize, beta,bp_tmp1,LD11)
       alpha=-1.d0; beta= 1.d0
       call DGEMM__('T', 'N', np_e,icnt1,icsize, alpha,psi_l(ibl1,1,ik,2),kg1, wk_ai1,ibsize, beta,bp_tmp1,LD11)
       alpha= 1.d0; beta= 1.d0
       call DGEMM__('T', 'N', np_e,icnt1,icsize, alpha,psi_l(ibl1,1,ik,1),kg1, wk_ai1,ibsize, beta,bp_tmp2,LD21)
       alpha= 1.d0; beta= 1.d0
       call DGEMM__('T', 'N', np_e,icnt1,icsize, alpha,psi_l(ibl1,1,ik,2),kg1, wk_ar1,ibsize, beta,bp_tmp2,LD21)
     end if
  end subroutine betar_dot_WFs_core_blk
```


© NEC Corporation 2011

非局所ポテンシャルと波動関数の積

$$V_{NL}\left|\Psi_{k\nu}\right\rangle = \sum_{I}\sum_{n,m} D_{nm}^{\varsigma(I)}\left|\beta_{n}^{I}\right\rangle f_{mk\nu}^{I}$$

オリジナルのループ構造

```
Line
 100          subroutine m_ES_Vnonlocal_W(ik,iksnl,ispin,switch_of_eko_part)
  :                        :
  :             call tstatc0_begin('m_ES_Vnonlocal_W',id_sname)
  :                        :
 200:  +------>   Loop_ntyp: do it = 1, ntyp
  :
 300:  |+----->      Loop_natm : do ia = 1, natm
 400:  ||                 if(ityp(ia) /= it) cycle
  :    ||
 500:  ||+----->          do i = 1, iba(ik)
 600:  |||                   nbase(i), ngabc(nbase(i))        → zfcos(i), zfsin(i)
 700:  |||+-----           enddo
 800:  |||------>          do lmt2 = 1, ilmt(it)
 900:  ||||+---->             do lmt1 = 1,ilmt(it)
1000:  ||||+--->                do i = 1, iba(ik)
1100:  ||||||                      zfcos(i), zfsin(i), snl(i)   → ss(i),sc(i),qc(i),qs(i)
1200:  ||||||+--                enddo
1300:  ||||+---              enddo
1400:  ||||+--->             do ib = 1, np_e
1500:  ||||+--->                do i = 1, iba(ik)
1600:  ||||||                      ss(i), sc(i), fsr_l, fsi_l   → vnlph_l(i)
                                   qs(i), qc(i)
1700:  ||||||+--                end do
1800:  ||||+---              enddo
1900:  |||------          enddo
2000:  ||+-------       enddo Loop_natm
2100:  |+--------    enddo Loop_ntyp
  :
2200             call tstatc0_end(id_sname)
2300          end subroutine m_ES_Vnonlocal_W
```

行列ベクトル積

修正後

```
Line
 100          subroutine m_ES_Vnonlocal_W(ik,iksnl,ispin,switch_of_eko_part)
 200  +------>  do ia = 1, natm
      +------>     do lmt2 = 1, ilmt(it)
                      処理の個数をカウント(icnt)
                      リスト作成
      +-------     enddo
      +--------  enddo

 300  +------>  do ibl1=1,iba(ik),ibsize                    ループをブロッキング
 400               ibl2=min( ibl1+ibsize-1,iba(ik) )

 500  +------>     do ia = 1, natm
 600  |+----->        do i = ibl1, ibl2
 700  |||               nbase(i), ngabc( nbase(i) )        => zfcos(i,ia), zfsin(i,ia)
 800  |||+-----        enddo
 900  ||+--------    enddo
1000  ||+------>     do ic = 1, icnt
1100  ||+------>        do lmt1 = 1,ilmt(it)
1200  ||+----->           do i =  ibl1, ibl2
                          zfcos(i,ia), zfsin(i,ia)        => ss(i,ic), sc(i,ic)
                          snl(i)                             qs(i,ic), qc(i,ic)
1300  ||+-----           enddo
1400  ||+--------     enddo
1500  ||+--------    enddo
1600  ||+------>     do ic = 1, icnt_with                   行列行列積に
1700  ||+------>        do ib = 1, np_e
1800  ||+----->           do i = ibl1, ibl2
                             ss(i,ic),      sc(i,ic)      => vnlph_l(i,ib)
                             qs(i,ic),      qc(i,ic)
                             fsr_l(ib,ic), fsi_l(ib,ic)
1900  ||+-----           enddo
2000  ||+--------     enddo
2100  |+---------    enddo                                  add_vnlph_l_(with/without)_eko_blk

2200  |+-----          end do
2300          end subroutine m_ES_Vnonlocal_W
```


© NEC Corporation 2011

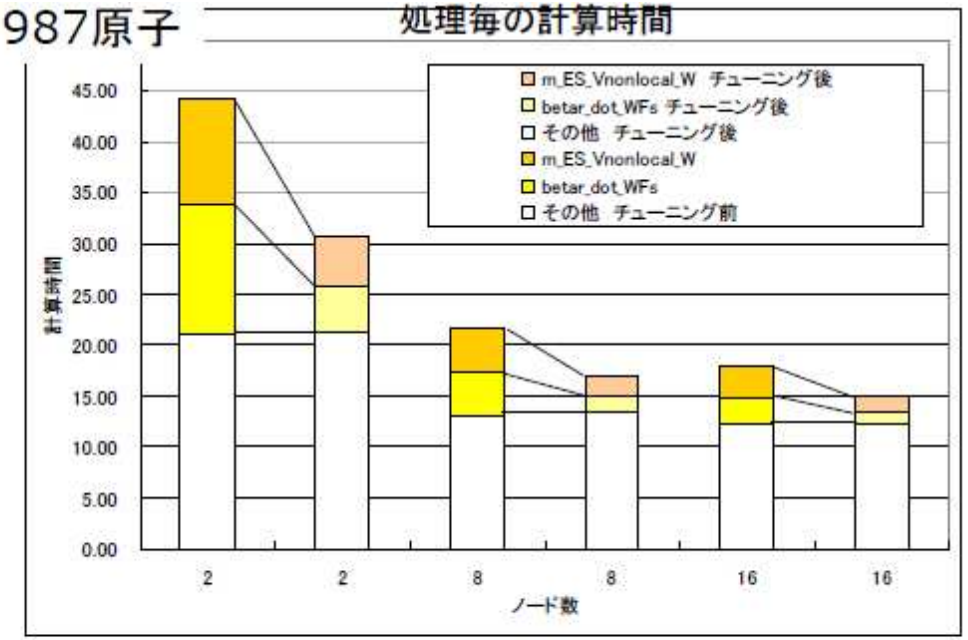
非局所ポテンシャルと波動関数の積：核心部

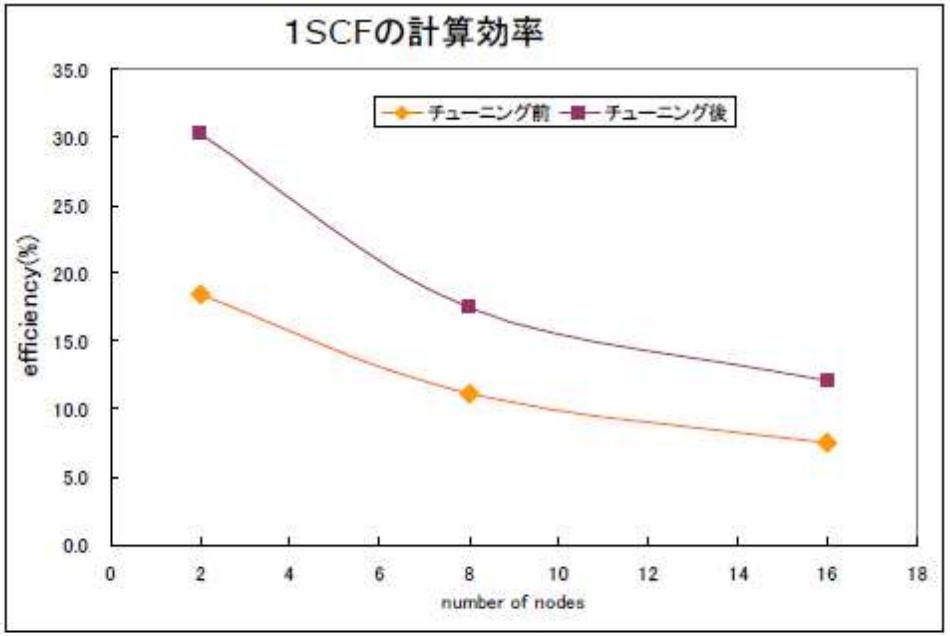
```
subroutine add_vnlph_l_without_eko_blk(ibsize,ibl1,ibl2,icnt_without,vnlph)
   integer, intent(in) :: ibsize, ibl1, ibl2, icnt_without
   real(kind=DP), intent(inout), dimension(kg1,np_e,kimg) :: vnlph
   integer      :: ic, ib, i
   integer      :: icsize
   real(kind=DP) :: alpha, beta

   if(kimg == 1) then
      ......
   else if(kimg == 2) then
     icsize=ibl2-ibl1+1
     alpha= 1.d0;  beta= 1.d0
     call DGEMM__('N','T', icsize,np_e,icnt, alpha,wk_sc,ibsize, fsr_tmp,np_e, beta,vnlph(ibl1,1,1),kg1)
     alpha=-1.d0;  beta= 1.d0
     call DGEMM__('N','T', icsize,np_e,icnt, alpha,wk_ss,ibsize, fsi_tmp,np_e, beta,vnlph(ibl1,1,1),kg1)
     alpha= 1.d0;  beta= 1.d0
     call DGEMM__('N','T', icsize,np_e,icnt, alpha,wk_sc,ibsize, fsi_tmp,np_e, beta,vnlph(ibl1,1,2),kg1)
     alpha= 1.d0;  beta= 1.d0
     call DGEMM__('N','T', icsize,np_e,icnt, alpha,wk_ss,ibsize, fsr_tmp,np_e, beta,vnlph(ibl1,1,2),kg1)
     end if
   end if
 end subroutine add_vnlph_l_without_eko_blk
```


© NEC Corporation 2011

性能最適化の効果(1/2)

Si 987原子





Si 1000原子

		1nodeの結果				4nodeの結果			
		MFLOPS	efficiency	V.OP RATIO	AVER. V. LEN	MFLOPS	efficiency	V.OP RATIO	AVER. V. LEN
betar_dot	dgemm_	61,007.1	59.6	99.37	197.5	49,544.8	48.4	99.28	166
	〃—micro1	60,964.5	59.5	99.37	197.5	49,482.2	48.3	99.28	166
	〃—micro2	61,005.3	59.6	99.37	197.5	49,560.2	48.4	99.28	166
	〃—micro3	61,005.4	59.6	99.37	197.5	49,551.3	48.4	99.28	166
	〃—micro4	61,027.6	59.6	99.37	197.5	49,558.9	48.4	99.28	166
	〃—micro5	60,999.1	59.6	99.37	197.5	49,563.8	48.4	99.28	166
	〃—micro6	61,007.7	59.6	99.37	197.5	49,556.2	48.4	99.28	166
	〃—micro7	61,021.2	59.6	99.37	197.5	49,548.6	48.4	99.28	166
	〃—micro8	61,025.8	59.6	99.37	197.5	49,537.0	48.4	99.28	166
Vnonlocal	dgemm_	75,361.4	73.6	99.41	249.6	75,051.1	73.3	99.39	249.5
	〃—micro1	75,319.8	73.6	99.41	249.6	74,922.4	73.2	99.39	249.5
	〃—micro2	75,386.2	73.6	99.41	249.6	75,085.0	73.3	99.39	249.5
	〃—micro3	75,353.2	73.6	99.41	249.6	75,103.4	73.3	99.39	249.5
	〃—micro4	75,369.1	73.6	99.41	249.6	75,027.8	73.3	99.39	249.5
	〃—micro5	75,359.2	73.6	99.41	249.6	75,062.5	73.3	99.39	249.5
	〃—micro6	75,341.3	73.6	99.41	249.6	75,065.4	73.3	99.39	249.5
	〃—micro7	75,388.4	73.6	99.41	249.6	75,086.1	73.3	99.39	249.5
	〃—micro8	75,373.8	73.6	99.41	249.6	75,056.9	73.3	99.39	249.5

dgemm部分は、ピーク性能の48.4%～78.0%の効率

Si 2744原子

		4nodeの結果			
		MFLOPS	efficiency	V.OP RATIO	AVER. V. LEN
betar_dot	dgemm_	72,384.7	70.7	99.52	249.4
	〃—micro1	72,214.5	70.5	99.52	249.4
	〃—micro2	72,409.5	70.7	99.52	249.4
	〃—micro3	72,411.5	70.7	99.52	249.4
	〃—micro4	72,399.8	70.7	99.52	249.4
	〃—micro5	72,398.1	70.7	99.52	249.4
	〃—micro6	72,417.9	70.7	99.52	249.4
	〃—micro7	72,411.4	70.7	99.52	249.4
	〃—micro8	72,415.4	70.7	99.52	249.4
Vnonlocal	dgemm_	79,872.2	78.0	99.42	249.9
	〃—micro1	79,676.0	77.8	99.42	249.9
	〃—micro2	79,880.2	78.0	99.42	249.9
	〃—micro3	79,891.6	78.0	99.42	249.9
	〃—micro4	79,875.2	78.0	99.42	249.9
	〃—micro5	79,909.2	78.0	99.42	249.9
	〃—micro6	79,898.2	78.0	99.42	249.9
	〃—micro7	79,913.0	78.0	99.42	249.9
	〃—micro8	79,934.6	78.1	99.42	249.9

© NEC Corporation 2011

性能最適化の効果(2/2)

Si 4,096原子

演算効率、ベクトル演算率

	8ノード64プロセッサ	32ノード256プロセッサ
Real Time (sec)	1919.19	879.56
User Time (sec)	11934.91	3602.78
System Time (sec)	20.26	15.18
Vector Time (sec)	10637.37	2808.98
Instruction Count	5.9246E+12	1.6365E+12
Vector Instruction Count	2.7200E+12	7.0450E+11
Vector Element Count	6.7654E+14	1.7505E+14
FLOP Count	4.2828E+14	1.1035E+14
MOPS	56953.95	48847.47
MFLOPS	35884.51	30628.76
Performance(%)	35.04	29.91
Average Vector Length	248.73	248.48
Vector Operation Ratio (%)	99.53	99.47
Memory size used (MB)	55744.00	17536.00
Global Memory size used (MB)	128.00	128.00
MIPS	496.41	454.24
Instruction Cache miss (sec)	6.91	6.64
Operand Cache miss (sec)	139.58	129.12
Bank Conflict Time		
CPU Port Conf.	954.83	358.43
Memory Net. Conf.	4607.40	1219.75

←→ チューニング前は、約20%(水864分子)

並列化率と最大利用可能ノード数

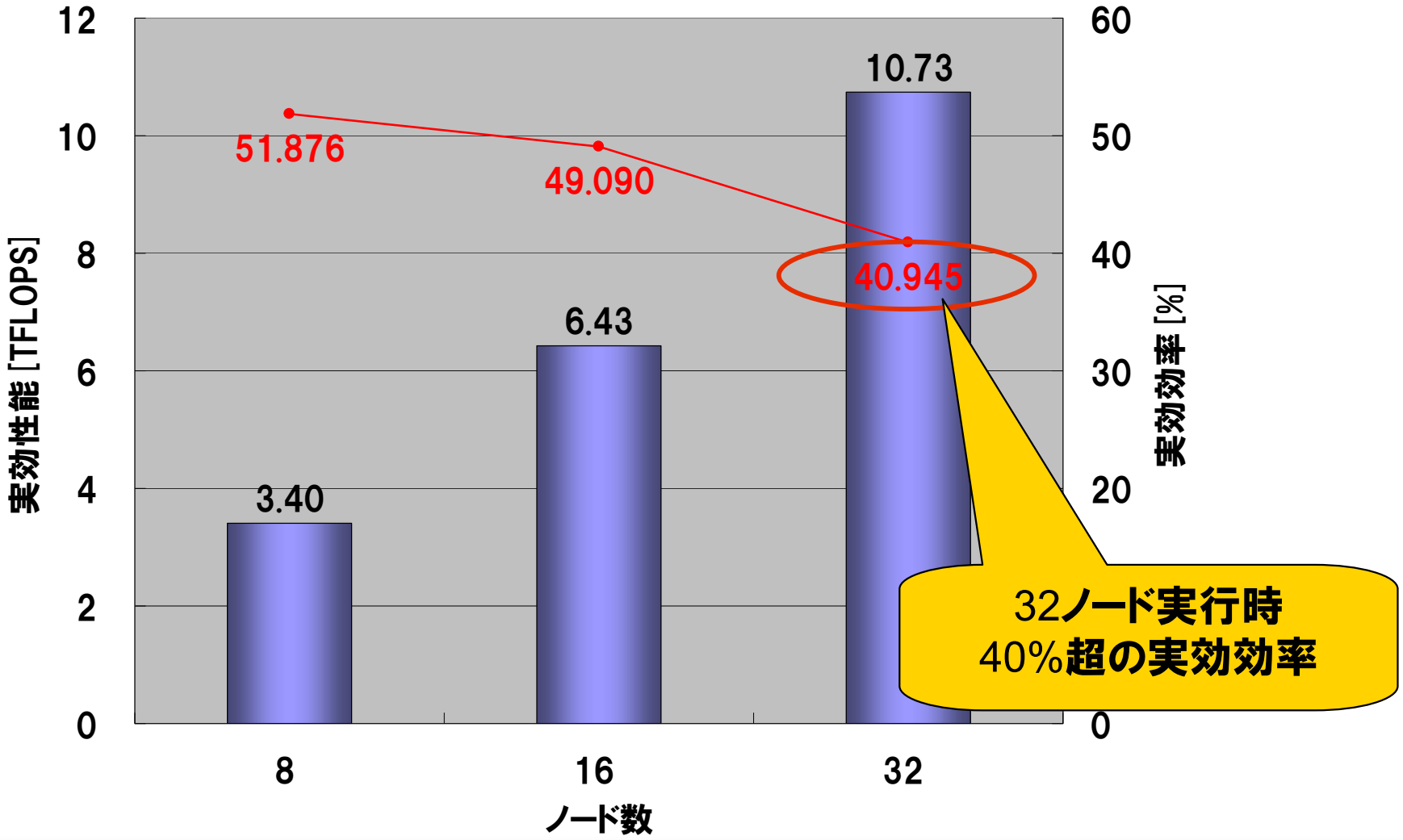
プロセッサ数n	64	実行時間Tn(秒)	1919.332
プロセッサ数m	256	実行時間Tm(秒)	879.853
並列化率α	99.402301%		
最大利用プロセッサ数	168.31		
最大利用ノード数	21.0		

© NEC Corporation 2011

最適化の効果（2011年の測定結果）

Si 4096原子における主要処理部分（SCFループ）の性能

- **実効効率は40%を超える**



© NEC Corporation 2011

おわりに

- PHASEに適用された性能最適化について、最適化内容の概要と効果を見ていただきました
  - これらの最適化の多くは、地球シミュレータ以外のプラットフォームでも有効です
- その後も、様々な最適化を適用し性能向上に努めています
  - 各種ソルバについてもDGEMM化を適用
  - MPI並列化の促進
  - 並列ライブラリ(ScaLAPACK)の利用促進
  - 各種プラットフォーム(京、PCクラスタ、地球シミュレータ)向けの性能最適化
- 今後も最適化を促進し、効率の良い運用が可能となるよう努めてまいります

© NEC Corporation 2011

NECのHPCアプリケーション高度化サービス

アイディアを成果に結びつけるお手伝い

- 詳しくは弊社Webページ
  http://www.nec.co.jp/solution/hpc/app_service/index.html
  をご参照ください

機能強化サービス

- アプリケーションの「機能」を新規開発または強化するサービスメニュー群です

性能強化サービス

- アプリケーションの本質的な機能はそのままに、「性能」を強化することにフォーカスしたサービスメニュー群です

実行支援サービス

- アプリケーションの「実行」に関する様々なお悩みを解決するサービスメニュー群です

© NEC Corporation 2021

NECグループビジョン2017

人と地球にやさしい情報社会を
イノベーションで実現する
グローバルリーディングカンパニー

Empowered by Innovation