High performance eigenvalue solver in the emerging petascale computing era

Toshiyuki Imamura
The University of Electro-Communications
1-5-1 Choufugaoka, Chofu-shi, Tokyo 182-8585
imamura@im.uec.ac.jp

Facing on the emerging petascale computing era, we need much more knowledge combining hardware and software, in order to achieve higher performance on a large scale parallel system. More than thousand cores should be used on a parallel program, and they play a role on a hierarchical complex parallel program, for example, written in MPI, OpenMP, SIMD directives etc.. We have to examine the existing algorithms whether they have higher parallelism, and whether they work effectively with more than ten thousand cores.

In this talk, focusing on a case study of large-scale eigenvalue computation on such as the Earth Simulator and a T2K supercomputer system, the author would like to present some perspectives on large-scale parallel computing towards the next generation Peta-scale computer.

High performance eigenvalue solver in the emerging petascale computing era

Toshiyuki Imamura
The University of Electro-Communications, Japan

Thanks to
M. Machida and S. Yamada (JAEA), and
Auto-tuning research group

Outline

1. Review
   - Numerical algorithm for eigenvalue problems
2. Current eigensolver performance
   - On a Large scale system
   - On a single multicore PC
3. Problem on Householder tridiagonalization
   - Approach via Narrow-band reduction
4. Performance
   - on a single socket
   - Parallel Performance on T2K
   - Perspectives
5. Conclusion

1.1, Eigenvalue analysis

- Big requirements from numerical simulation fields
  - Structure analysis, Quantum simulation, Computational Chemistry, Finance ... etc.
  - **Styles of their Eigenproblems are different…**
    - Standard eigenproblem

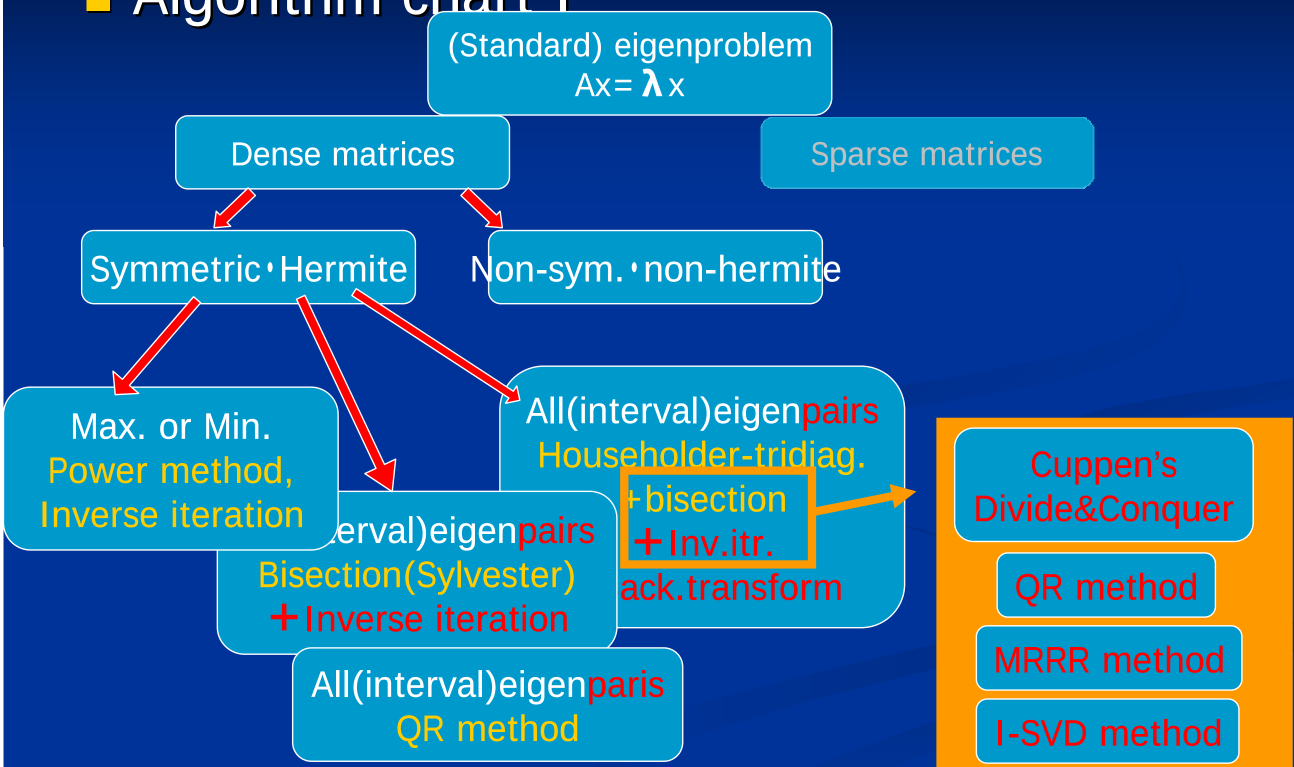
      $Ax=\lambda x$ or $AX=X\Lambda$
    - General eigenproblem

      $Ax=\lambda Bx$ or $AX=BX\Lambda$

The system dimension and the number of necessary eigenmodes is also different.

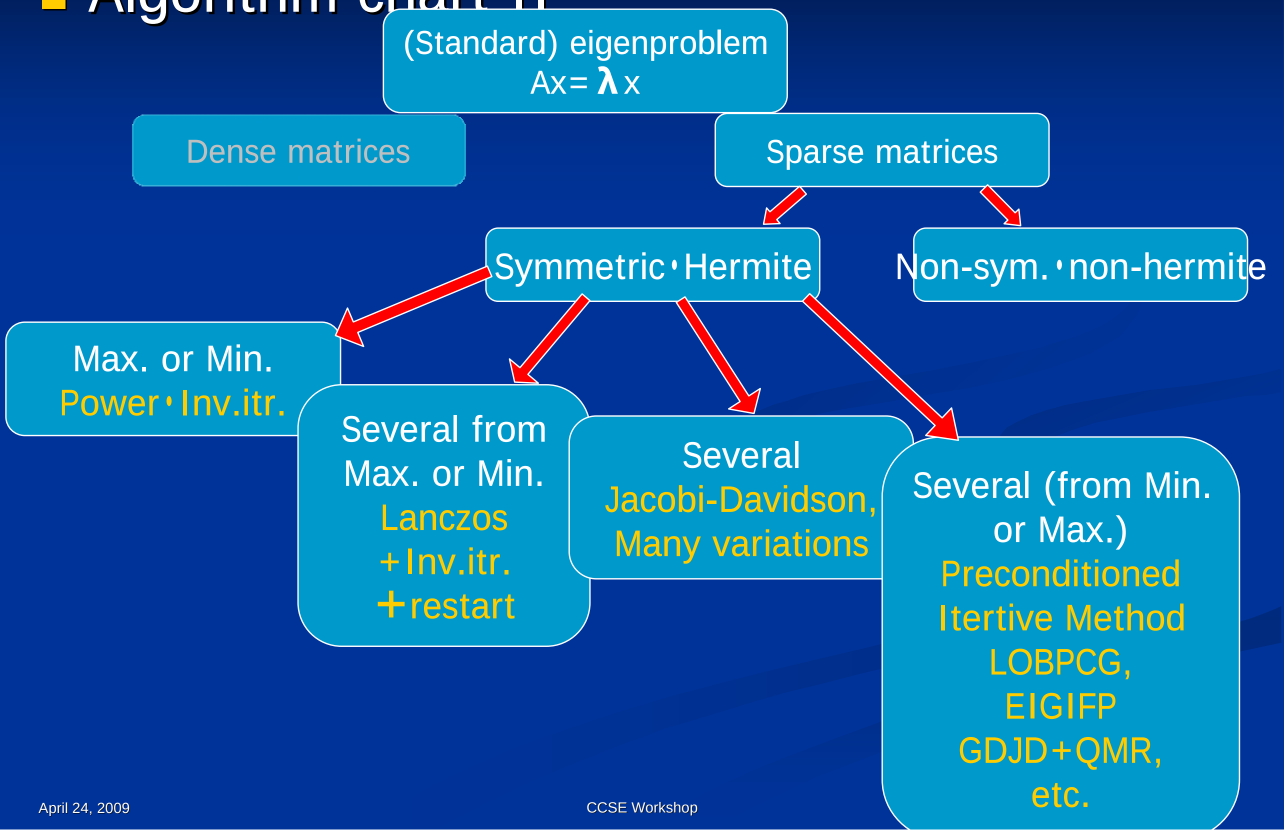
1.2, "Numerical Algorithms (1)"

- Algorithm chart I



1.2', "Numerical Algorithms (2)"

- Algorithm chart II



1.3, Iterative method

- Krylov subspace method
  - Exploring $V := \{x, Ax, A^2x, \ldots\}$ to find the min or max of $y^T Ay$, where y is chosen from the spanned space with V → to find the min/max mode of $V^T AV$.
  - Lanczos method
  - Jacobi-Davidson method
  - Conjugate Gradient method

Non-sym.	Pre-cond.	Block-ing	Stability
△	×	○	×
○	○	×	○
×	○	○	○

  - Newton method is also available.
    (sort of inverse power-iteration method)

1.4, Krylov subsp. Method

$$V^{(k+1)} := \{P^{\perp V^{(k)}} f_A(v^{(k)}), V^{(k)}\}$$ :: General update rule

- Lanczos
  - Klyrov+Gram-schmidt, Reduction to a compact tridiagonal form
- LOBPCG
  - Ritz vector⇒ {residual, approx., prior} ∵analogous to CGM

- Lanczos

$x_0 \neq 0$ (an initial guess)
$\beta_0 = 0, v_{-1} = 0, v_0 = x_0/\|x_0\|$,
do k=0, … $m-1$ or until convergence
  $w = Av_k - \beta_k v_{k-1}$
  $\alpha_k = (w, v_k)$
  $w = w - \alpha_k v_k$
  $\beta_{k+1} = \|w\|$
  $v_{k+1} = w/\beta k + 1.$
enddo

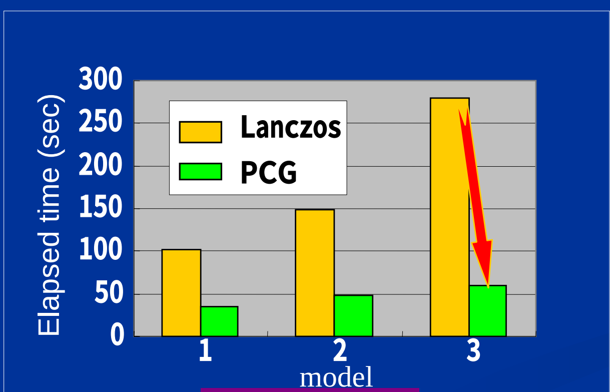
- LOBPCG

$x_0 \neq 0$ (an initial guess), $p_0 = 0$
do k=0, … until convergence
  $\mu_k = (x_k, Ax_k)/(x_k, x_k)$
  $w_k = T(Ax_k - \mu_k x_k)$
  $S_A = \{w_k, x_k, p_k\}^T A\{w_k, x_k, p_k\}$
  $S_B = \{w_k, x_k, p_k\}^T \{w_k, x_k, p_k\}$
  Solve $S_A v = \mu S_B v,\ v = (\alpha, \beta, \gamma)^T$.
  $x_{k+1} = \alpha w_k + \beta x_k + \gamma p_k$
  $p_{k+1} = \alpha w_k + \gamma p_k$
enddo

1.5, Result at the Gordon Bell Finalist Session, SC05 and 06

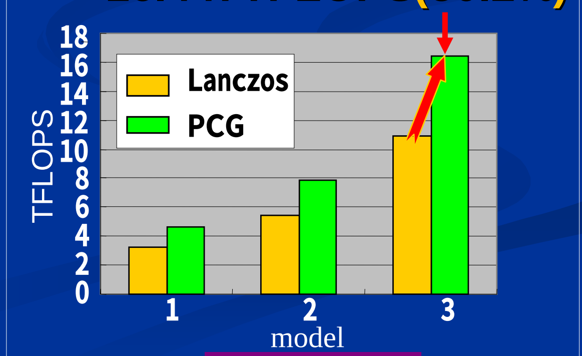
Problem & Dimension of Hamiltonian matrices

Model	No. of	No. of Fermions ↑-spin	↓-spin	Dim of H	No. of Nodes	Memory (TB) Lanczos	PCG
1	24	6	6	18,116,083,216	128	0.8	1.3
2	21	8	8	41,408,180,100	256	1.9	2.9
3	22	8	8	102,252,852,900	512	4.6	6.9



Elapsed time

16.447TFLOPS(50.2%)

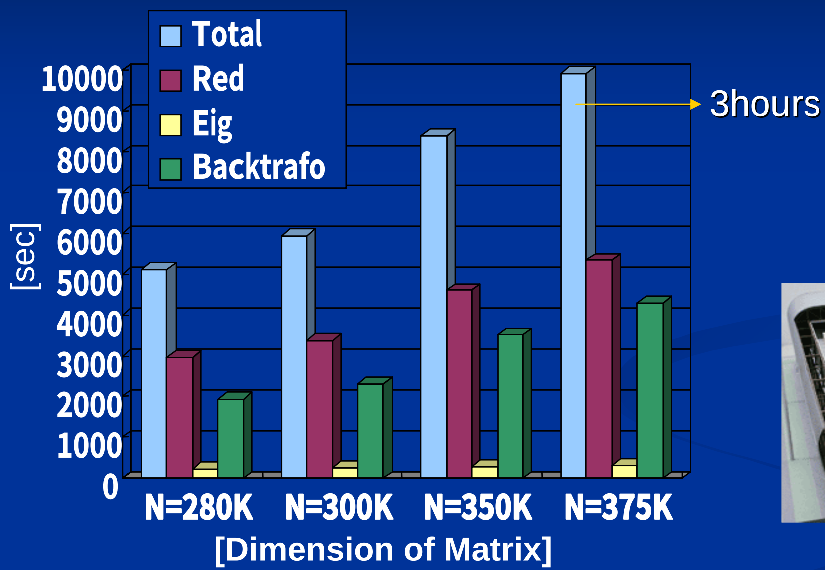


Performance

Focusing on DRSM (Dense-Real-Symmetric-Matrices) diagonalization

MULTICORE AND MULTIPROCESSOR PERFORMANCE

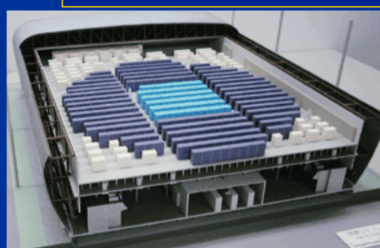
2.1 How fast does a parallel eigensolver perform on ES with a huge problem?



Matrix dim.:
280,000~375,000

Earth Simulator:
4,096 VPU's

Memory:
2~3 T Bytes

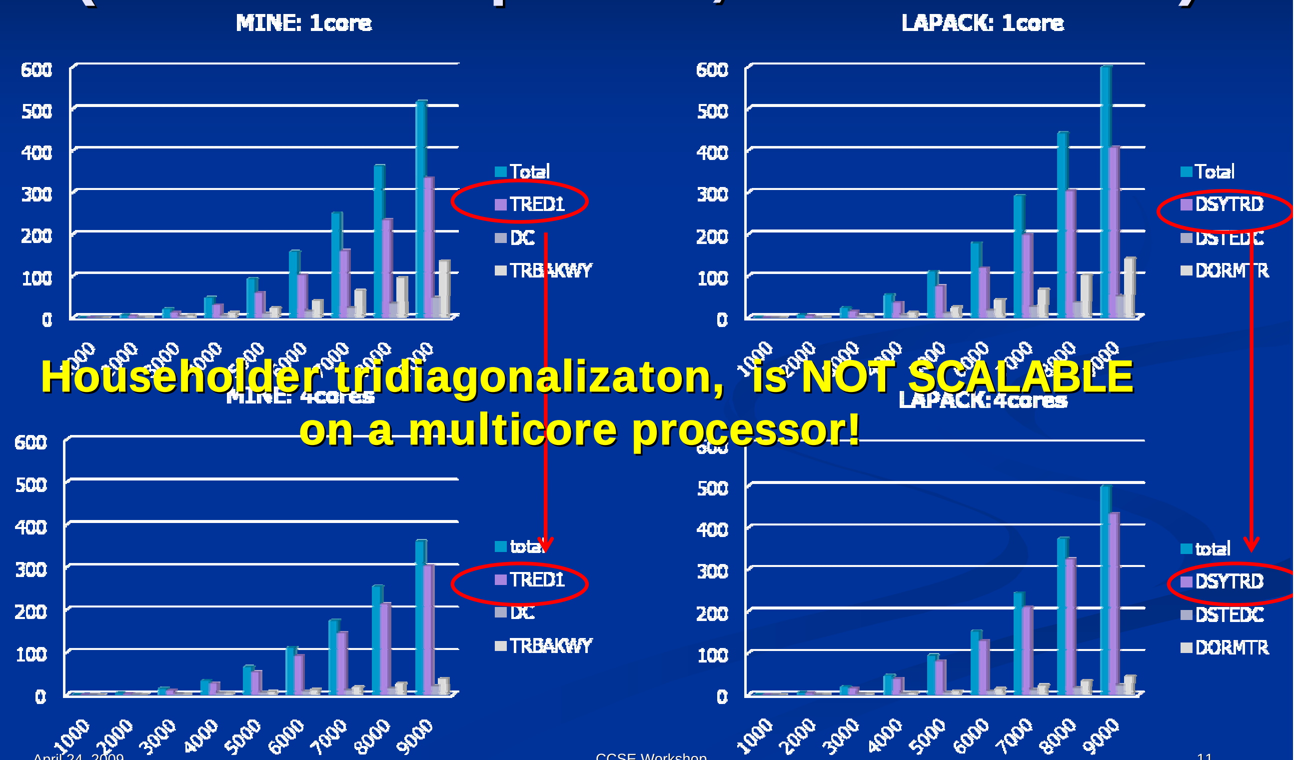


Reported at SC|06, Tampa.

We confirmed stability of the solver up to 375,000 X 375,000

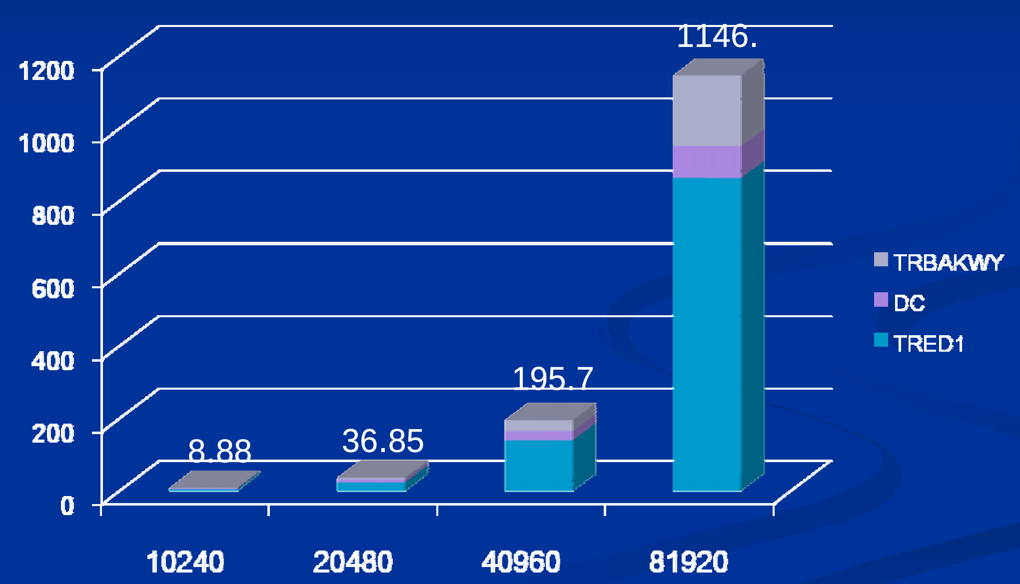
※ Accuracy is pretty excellent up to 300K X 300K (confirmed).

2.2 Performance on a Multicore processor with small problems (Intel Xeon: Up 1core, Bottom 4cores)



Householder tridiagonalizaton, is NOT SCALABLE on a multicore processor!

2.2' Performance on a Multicore mutli-processors (T2K 64nodes, 1024cores)



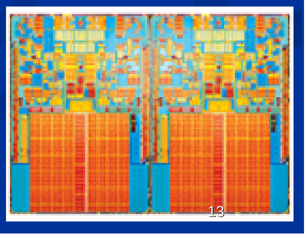
2.3 Motivation of My talk: We are now on the multicore age

- We need a high performance and scalable eigenvalue solver on from tera-scale to peta-scale computers. Furthermore, beyond them, Hexa-scale computing environment....

- What is the significant drawback?

  POOR (not rich) memory bandwidth.

  - **Performance bound**
  - Conflict on memory access with multicores
  - Cache consistency problem
  - Deep memory hierarchy



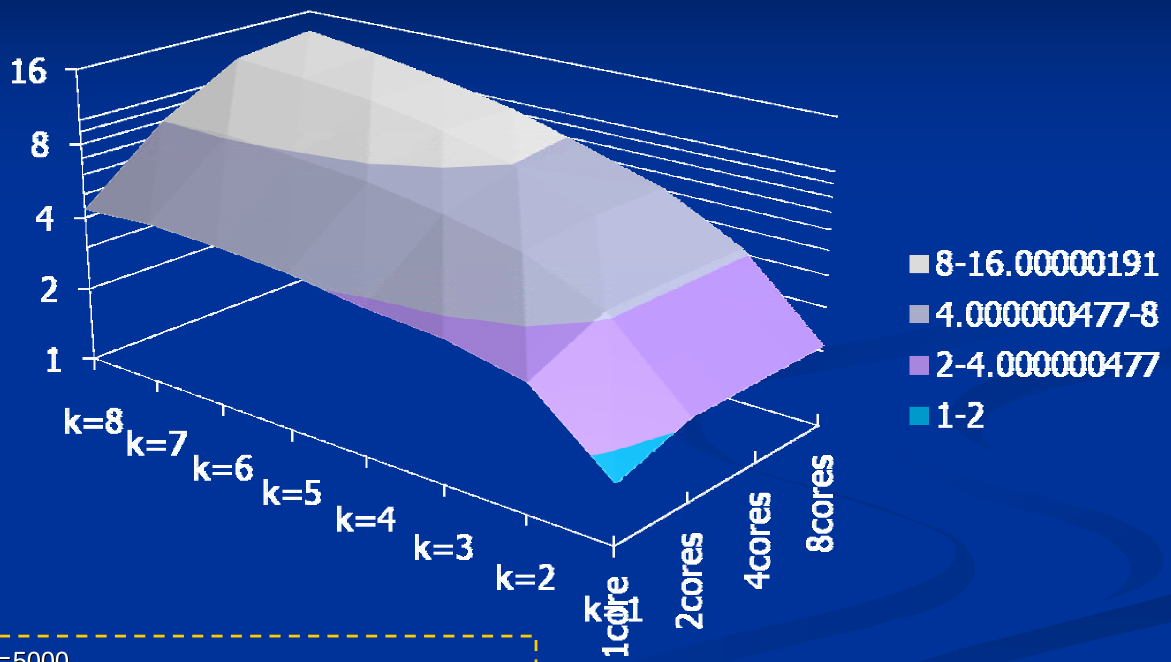
2.4 Our approach towards a peta-scale computer

- Blocking strategy
  - Single vector ops. → Multiple vectors ops.
  - Displace Level 2 BLAS → Level 3 BLAS.
  - Message aggregation → less data comms.
  - Multi-dimensional data division

    → multithreading with thousands of cores.
- However, we need algorithm change.
  - Drastically, and .....

2.5 Preliminary result

Mat*vecs **(Ave. GFLOPS among N=1...5K)**



N=5000
Intel Xeon E5345 (2.33GHz, Quad) dual socket
FB-DIMM: 533MHz

Question:

Strategy to replace "L2 BLAS" to "L2.5 or 3 BLAS", does it truly perform well?

SINGLE MULTICORE PERFORMANCE !

3.1, Householder narrow-band-reduction

for $j = N, \ldots, 1$ **step** $-M$

$U \leftarrow \emptyset, V \leftarrow \emptyset, W \leftarrow A_{(*,j-M+1:j)}$

for $k = 0, \ldots, M-1$ **step** K

(1) Householder block reflector: $(C, U^{(k)}) := H(W_{(*,j-k)})$

(2) Matrix-Vectors multiplication (BLAS2.5)

$V^{(k-\frac{2}{3})} \leftarrow A_{(1:j-k-1,1:j-k-1)}U^{(k)}$

(3) $V^{(k-\frac{1}{3})} \leftarrow V^{(k-\frac{2}{3})} - (UV^T + VU^T)U^{(k)}$

(4) $V^{(k)} \leftarrow V^{(k-\frac{1}{3})}C^T - U^{(k)}S, S = \frac{1}{2}CU^{(k)^T}V^{(k-\frac{1}{3})}C^T$

$U \leftarrow [U, U^{(k)}], V \leftarrow [V, V^{(k)}].$

(5) $W_{(*,j-k:j)} \leftarrow W_{(*,j-k:j)} - (U^{(k)}V^{(k)^T} + V^{(k)}U^{(k)^T})_{(*,j-k:j)}$

endfor

$A_{(*,j-M+1:j)} \leftarrow W$

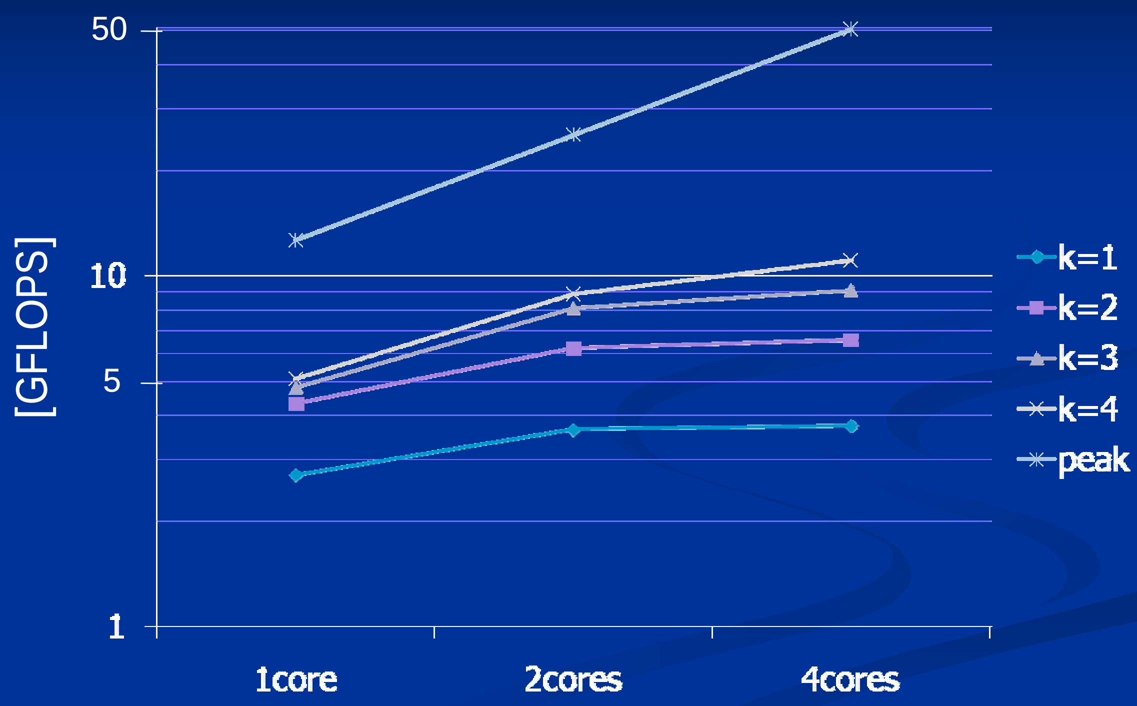
(6) $2M$ rank-update (BLAS3)

$A_{(1:j-M,1:j-M)} \leftarrow A_{(1:j-M,1:j-M)} - (UV^T + VU^T)_{(1:j-M,1:j-M)}$

endfor

3.2, Xeon: scalability check

Total performance; GFLOPS rate in Narrow-band reduction routine



3.3, Short Summary single multicore processor.

- The strategy to replace L2 BLAS to L2.5 or 3
  - It actually guarantees higher performance.
  - Since it reduces the required B/F, upper bound of performance increases naturally.
  - The replacement strategy can **take advantage of the power of multicore.**
  - However, enough wider bandwidth is necessary to assure the performance on a many-core processor...

Parallel Performance Narrow-band reduction (k=1)

k=1, equivalent to tridiagonalization

```
N=      10000 NM=       5008
 NUM.OF.PROCESS=          4 (          2          2 )
 NUM.OF.THREADS=          4
 calc (u,beta)       1.14345932006836
 mat-vec (Au)          73.0448567867279       9.12681187962571
2update (A-uv-vu)    7.71526074409485        86.4088316363026
 calc v             1.03407597541809
 v=v-(UV+VU)u          3.97916340827942
 UV post reduction   1.04587674140930
 COMM_STAT
    BCAST  ::   0.902075290679932
    REDUCE ::     3.12653517723083
    REDIST ::   0.000000000000000E+000
    GATHER ::    0.327677249908447
 TRD-BLK       10000     88.7165729999542         15.0291347856056      GFLOP
```

T2K supercom at U.Tokyo, single node (Theoretical Peak 147GFLOPS)

Parallel Performance
Narrow-band reduction (k=2)

k=2, bandwidth=5

```
N=        10000 NM=        5008
NUM.OF.PROCESS=           4 (          2          2 )
NUM.OF.THREADS=           4
calc (u,beta)        1.04480671882629
mat-vec (Au)          38.8238267898560       17.1715856418578
2update (A-uv-vu)   7.95715451240540        83.7820436473010
calc v             1.01791191101074
v=v-(UV+VU)u        1.80215930938721
UV post reduction  0.638929605484009
COMM_STAT
   BCAST   ::    1.08294034004211
   REDUCE ::     2.36893534660339
   REDIST ::   0.000000000000000E+000
   GATHER ::    0.453493356704712
TRD-BLK        10000     51.9860939979553         25.6478844782179      GFLOP
```

Communication cost decreases slightly

T2K supercom at U.Tokyo, single node (Theoretical Peak 147GFLOPS)

Parallel Performance
Narrow-band reduction (k=4)

k=4, bandwidth=9

```
N=        10000 NM=        5008
NUM.OF.PROCESS=           4 (          2          2 )
NUM.OF.THREADS=           4
calc (u,beta)        1.39939284324646
mat-vec (Au)          22.5483167171478       29.5661390173605
2update (A-uv-vu)   7.93256497383118        84.0417530604465
calc v             0.835858106613159
v=v-(UV+VU)u        1.24786186218262
UV post reduction  0.303748607635498
COMM_STAT
   BCAST   ::    0.902456998825073
   REDUCE ::     2.19763445854187
   REDIST ::   0.000000000000000E+000
   GATHER ::    0.429951906204224
TRD-BLK        10000     34.9330039024353         38.1682988688065      GFLOP
```

Communication cost decreases slightly

T2K supercom at U.Tokyo, single node (Theoretical Peak 147GFLOPS)

Parallel Performance Narrow-band reduction (k=8)

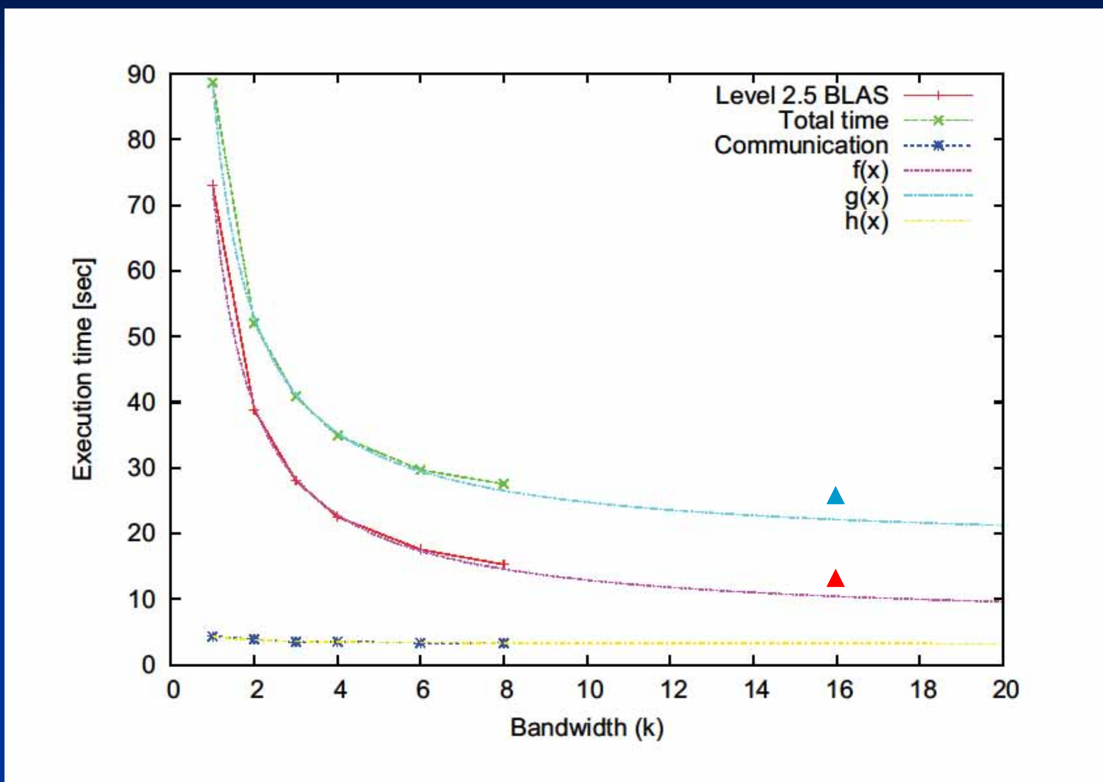
k=8, bandwidth=17

```
N=        10000 NM=        5008
 NUM.OF.PROCESS=          4 (          2          2 )
 NUM.OF.THREADS=          4
 calc (u,beta)      1.54309058189392
 mat-vec (Au)       15.2870652675629       43.6098528395274
 2update (A-uv-vu)   7.94437193870544       83.9168497913130
 calc v             0.902369499206543
 v=v-(UV+VU)u       1.03358817100525
 UV post reduction  0.152922153472900
 COMM_STAT
    BCAST   ::   0.905138254165649
    REDUCE ::     2.03297543525696
    REDIST ::   0.000000000000000E+000
    GATHER ::   0.421426534652710
 TRD-BLK        10000     27.5201659202576        48.4493202983151       GFLOF
```

Communication cost decreases slightly

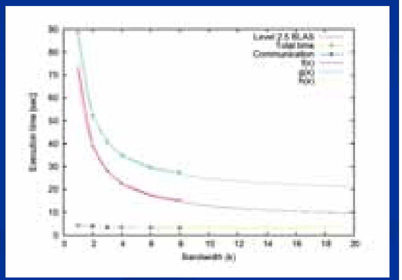
T2K supercom at U.Tokyo, single node (Theoretical Peak 147GFLOPS)

3.4, Parallel Performance



3.5, Discussion

- Performance Model (4procsx4threads, N=10K)
  - We assume $T_{\mathrm{NBR}} = T_{\mathrm{MV}} + T_{\mathrm{r2k}} + T_{\mathrm{others}}$
    - $T_{\mathrm{MV}}(k) = T_1 + T_2/k$
    - $T_{\mathrm{r2k}}(k) = T_3$
    - $T_{\mathrm{others}}(k) = T_4 + T_5/k$



  Therefore,

  $$T_{\mathrm{NBR}} = (T_1 + T_3 + T_4) + (T_2 + T_5)/k.$$

  - By fitting this function with the observation,

  $$T_{\mathrm{NBR}} = 17.5 + 70/k.$$

  - Further detail assumption, $T_{\mathrm{MV}} > T_{\mathrm{2rk}}$

  $T_{\mathrm{NBR}} > 19.4 \rightarrow$ 68.6GFLOPS (46.7% of peak)

3.6, From the Latest large experiment, tri-diagonalization (k=1)

```
NUM.OF.PROCESS=        4096 (          64          64 )
 calc (u,beta)       465.143042325973511
 mat-vec (Au)          6092.16216659545898        1835.93449432374246
   COMM1/2             781.987825393676758        730.568418025970459
 2update (A-uv-vu)   614.600288152694702          18198.5119146053039
 calc v           187.255089759826660
 v=v-(UV+VU)u        428.345677375793457
 UV post reduction   1.29367899894714355
 COMM_STAT
    BCAST  ::    535.078449249267578
    REDUCE ::     1960.95371365547180
    REDIST ::   0.000000000000000000E+000
    GATHER ::     62.2501411437988281
 TRD-BLK      256000     7799.53520894050598          2868.07107527270637
 PE partition =           64          64
 Split ROW/COLUMN DONE
 D&C    259.870339870452881       ERRCODE=           0
```

T2K supercom at U.Tokyo, 256nodes (Theoretical Peak 37.6TFLOPS)

3.7, Perspectives

- Householder tridiagonalization for a 25.6K dimensional matrix:
  - 7,799 [sec] on T2K 256 nodes (4096cores)
    6,092 [sec] is Level2 (P)BLAS
  - If k=2, 30-40% improvement is expected.
    2K-2.4K[sec] could be saved.→4~TFLOPS
  - If k=3~5, 40-60% improvement is expected.
    2.4K-3.6K[sec] could be saved.→5.5~TFLOPS
  - → **The approach will be acceptable (we can expect the performance beyond 15% of the peak).**
  - → **If 10PFLOPS machine is available, sustained performance will reach PFLOPS order.**
  - Untouched issue in this study is the cost of eigenpair computation for band matrices...

Question:

How faster does Narrow-Band reduction perform IT on a large scale system?

We need a bright perspective towards a Peta-scale machine.

CAN WE TRUST ON IT ?

YES!

4. Conclusion

1. Iterative method for Sparse matrices
   - Outstanding LOBPCG performance in SC|06
   - Beyond 100billion DOF
   - 24TFLOPS on Earth Simulator

2. DRSM issues
   - Algorithm should be replaced into a block version.
     - In our case Narrow-band reduction is **an inevitable approach** to diagonalize a dense matrix.
   - Level3 BLAS, matrix-matrix product, provides us relatively higher performance.
     - It takes account of **the power of multiple cores**.

   - We can expect peta-FLOPS performance on a ten-peta scale computer system!!

Any question ?

THANK YOU FOR YOUR PATIENT!