



Numerical Policy Interface for Automatic Tuning Library

Ken Naono
Central Research Laboratory, Hitachi, Ltd.

Acknowledgements

- Professor Toshitsugu Yuba
- Professor Kazuo Murota
- Professor Akimichi Takemura
- Professor Reiji Suda
- Professor Yusaku Yamamoto
- Professor Toshiyuki Imamura
- Professor Takahiro Katagiri
- Ministry of Education, Culture, Sports, Science and Technology (MEXT) Grant-in-Aid for Scientific Research on Priority Areas, “Cyber Infrastructure for the Information-explosion Era,”
Proposal Research A02-05, “Research on Mathematical Core for Robust Auto-Tuning Software in Information Explosion Era”, No.18049014.

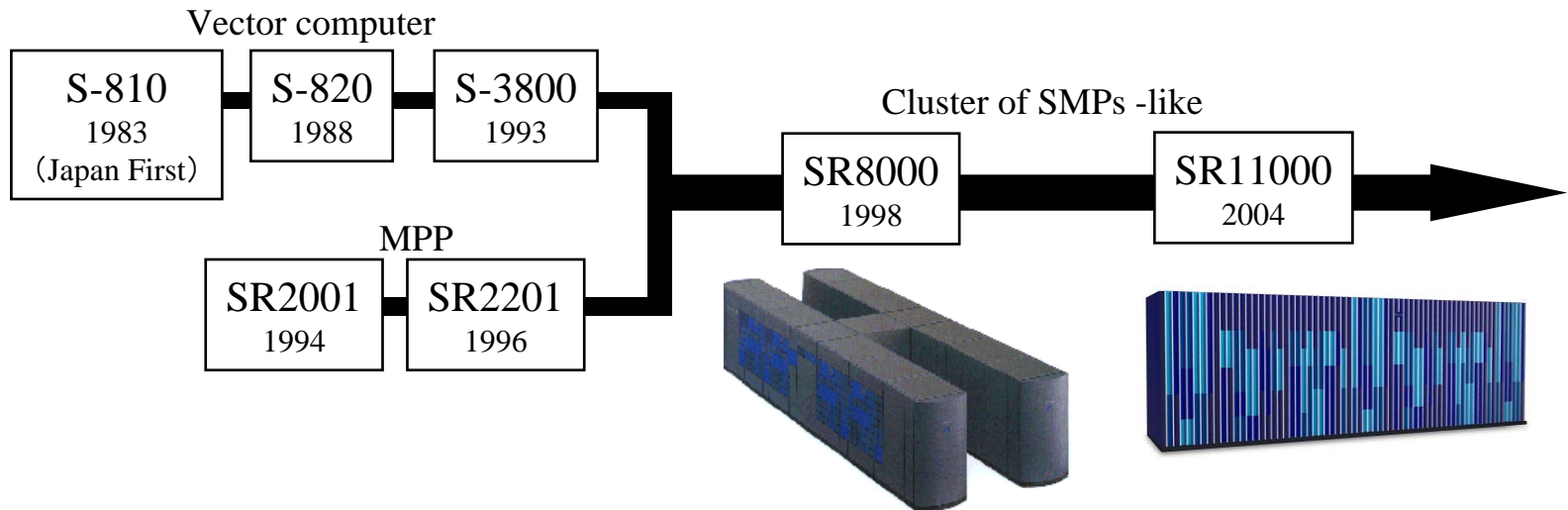
本研究の一部は文部科学省の科学研究費「情報爆発時代のロバストな自動チューニングソフトウェアに向けた数理的基盤技術の研究」(特定領域研究18049014)により行われています。

Hitachi MATRIX Series

“Matrix Libraries for Supercomputers”

(Dense/Sparse Linear Equation, Eigenvalue, FFT, and Random Number)

- MATRIX/HAP for S-810, S-820, and S-3800.
- MATRIX/MPP for SR2001, SR2201.
- MATRIX/MPP for SR8000, SR11000.



⇒ AT will be necessary.

Content

- 1 Introduction: Why is AT necessary?**
- 2 Definition of AT and its two viewpoints**
- 3 Existing AT classification**
- 4 Proposal of numerical policy interface**
- 5 Preliminary results on sparse eigensolver**
- 6 Conclusion**

1 -1 Background and motivation

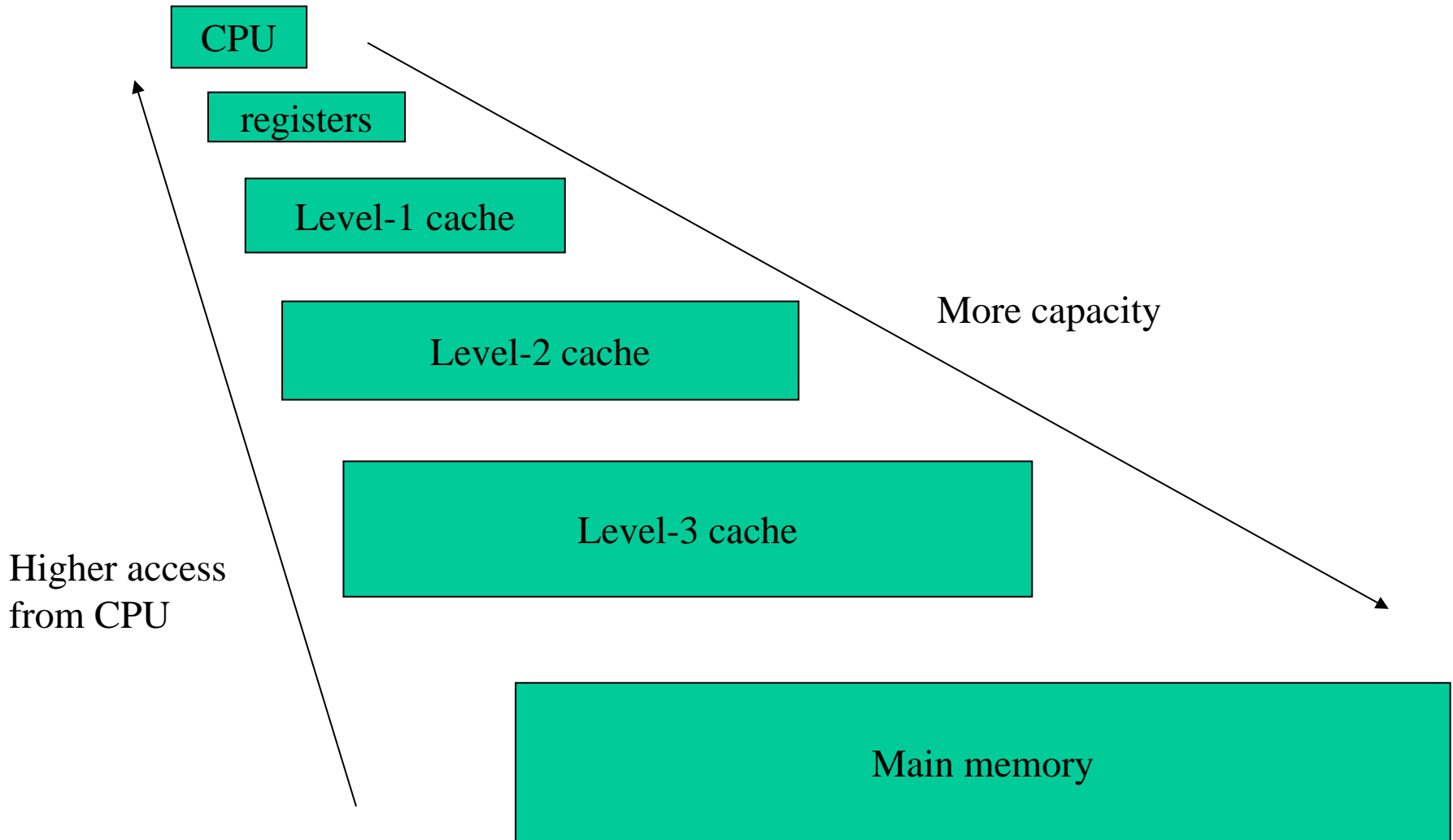
- Background
 - Various types of high performance computers.
 - The exploitations of high performance are very difficult and require enormous human power.
- Motivation of this research
 - To alleviate the cost of hand-coding on high performance computers.
 - To propose a new automatic tuning framework with performance-specific interface.

1 -2 Problems with matrix library use

1. A lot of trial-and-error computation.
 - Users cannot always control parameters related to computing accuracy and computing time.
 - How to define convergence criterion?
2. Number of matrix parameters has increased enormously.
 - HPC platforms are now massively parallelized, or hybrid parallelized
 - How to define CPU number, memory allocation ?
 - Various types of implementations
 - What kind of loop unrolling should I use?
 - Which algorithm should I use?

1 -3 Platform trend (1)

Multiple layered cache system



1 -4 Loop unrolling; more hand-coding time

```
DO J = 1,100
  DO I = 1,100
    C(J) = C(J) + A(I,J)*B(I)
  ENDDO
ENDDO
```


unrolling

```
DO J = 1,100,2
  DO I = 1,100
    C(J) = C(J) + A(I,J)*B(I)
    C(J+1) = C(J+1) + A(I,J+1)*B(I)
  ENDDO
ENDDO
```

STORE 8B + LOAD 24B
/ 2 FLOP

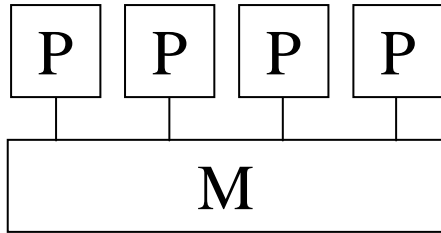


STORE 16B + LOAD 40B
/ 4 FLOP

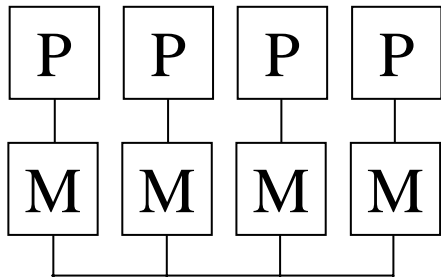
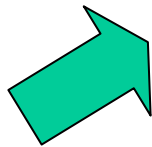
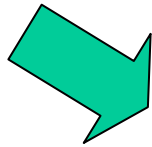
Less memory bandwidth requirement

1 -5 Platform trend (2)

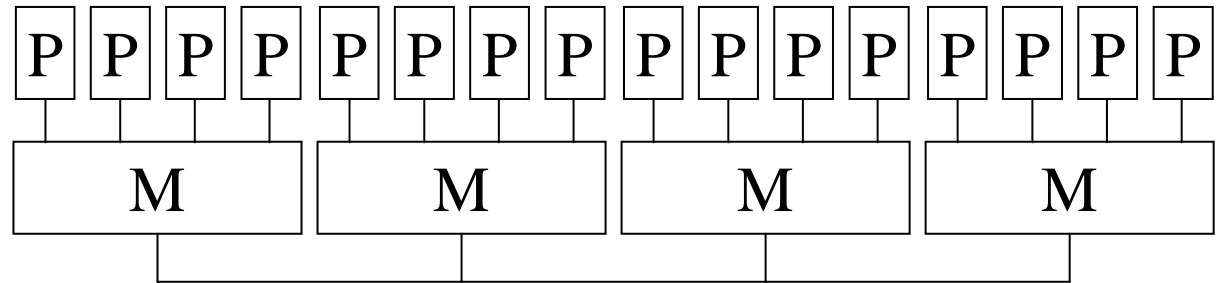
Parallel architecture development



SMP



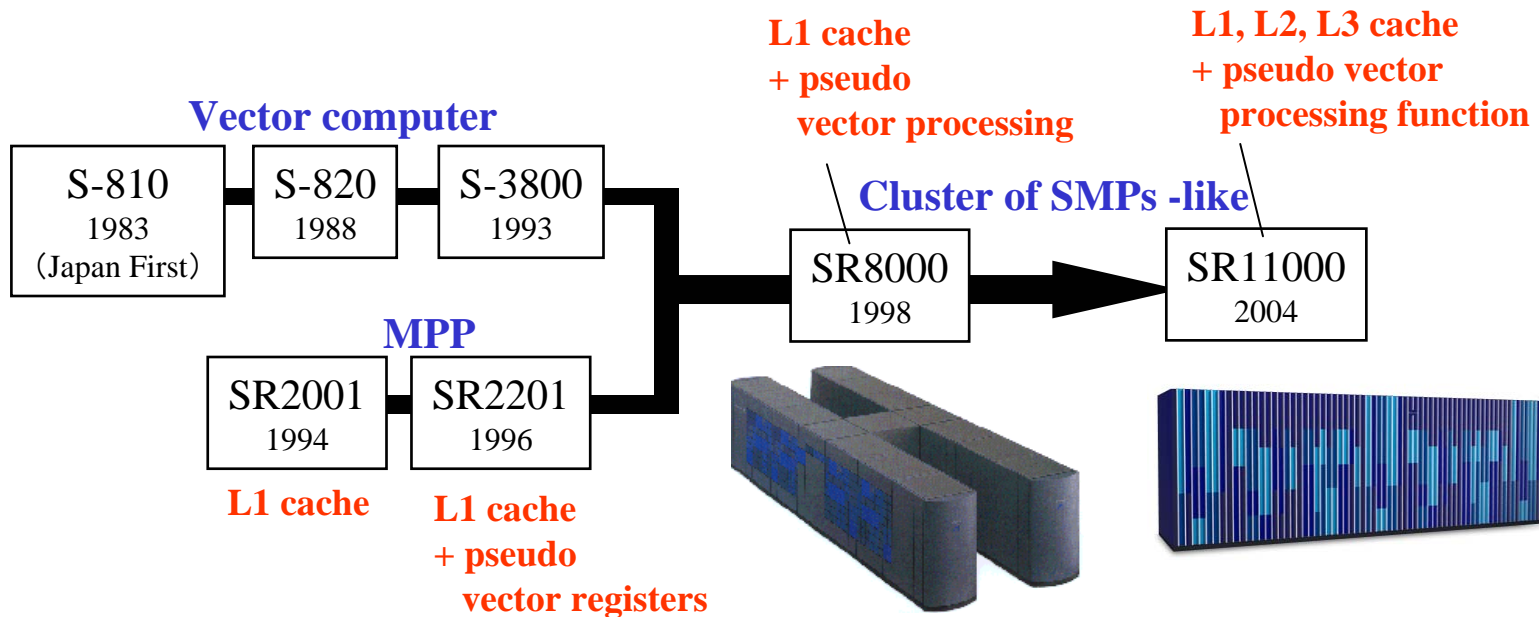
MPP



Cluster of SMPs

1 -6 Platform trend example

Supercomputing platform development history example:
The Hitachi supercomputers

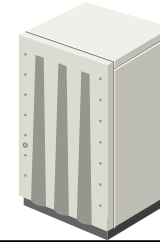


1 -7 Problems in the Grid generation

Policy

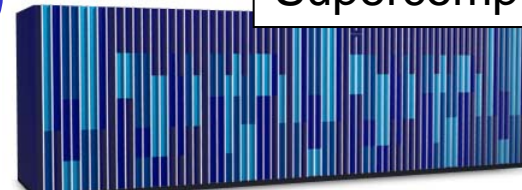
- Performance requirement :faster
- Output Data Requirement:
under 10^{-8} precision orthogonality

PC (high-end 1CPU)



Best -**Classical Gram-Schmidt**
Next -Modified Gram-Schmidt
Worst -DGKS Gram-Schmidt

Supercomputer



Best -**Modified Gram-Schmidt**
Next -DGKS Gram-Schmidt
Worst -Classical Gram-Schmidt

User



```
.....  
Call  
ClassicalGramSchmidt  
(V, N, JMAX)  
.....
```

The best algorithm on a PC may be the worst algorithm on a supercomputer.

1 -8 Purpose of today's talk

- Describe AT definition
 - Parameters and viewpoints
- Overview existing AT researches
 - Classification from the viewpoints
- Propose a new framework for AT
 - “Numerical Policy”
 - Tradeoff between accuracy and computing time
 - Tradeoff between resource and computing time
 - Number of CPUs and computing time
 - Memory allocation and computing time

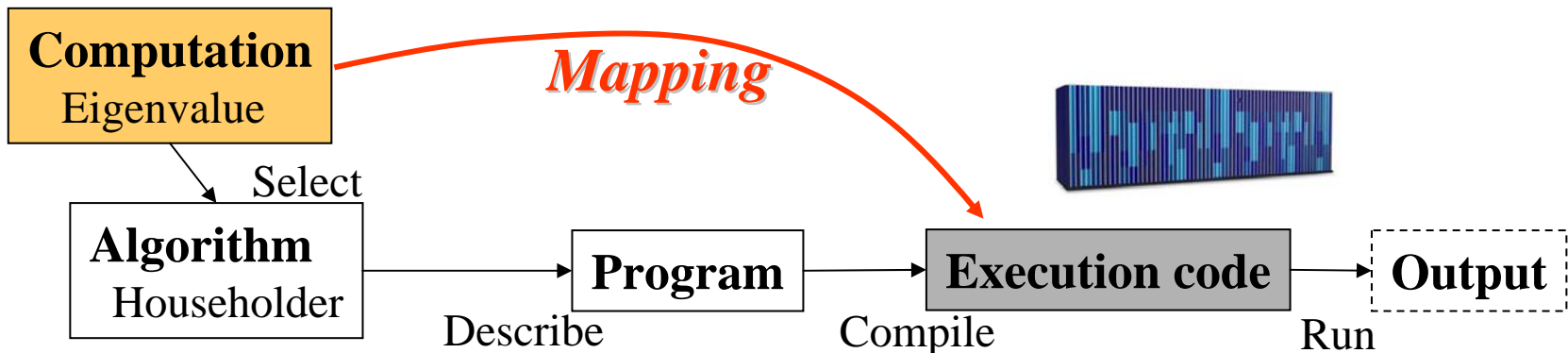
Content

- 1 Introduction: Why is AT necessary?
- 2 Definition of AT and two viewpoints**
- 3 Existing AT classification
- 4 Proposal of numerical policy interface
- 5 Preliminary results on sparse eigensolver
- 6 Conclusion

2 -1 Definition of AT

Automation of the best mapping procedure from “computation” to “execution code”, in order to exploit the highest performance of the computation.

- “Automatic Tuning” is described in PHiPAC(1997) and ATLAS(1998)
- “Optimized Mapping” by T. Yuba



2 -2 Examples

- Determine an internal parameter value under a given matrix size N .
 - Internal parameter: depth of loop unrolling, or block size
- Define determination stage of each parameter
 - Stage 1: Installation time
 - Stage 2: Before execution time (when making execution code)
 - Stage 3: Run time

2 -3 Mathematical form of AT

Def. 1: User Control Parameter (UCP)

A parameter that users want to control by themselves, e.g., matrix size or vector length.

Def. 2: Internal Critical Parameter (ICP)

A parameter that affects values of user objective functions (UOF), but is not controlled by users, e.g., depth of loop unrolling or maximum iteration number.

Def. 3: Resource Control Parameter (RCP)

A parameter that describes computation resource, e.g., number of CPUs or memory allocation.

Def. 4: User Objective Function (UOF)

A function that has values obtained after computation, and that users want to control, e.g., residual error or computation time.

Given UCP and RCP, find ICP to minimize
$$\text{UOF} = \text{UOF}(\text{UCP}, \text{ICP}, \text{RCP}).$$

Ex. $\text{Time} = \text{Time}(\text{N}, \text{Num_Iter}, \text{Num_CPU})$

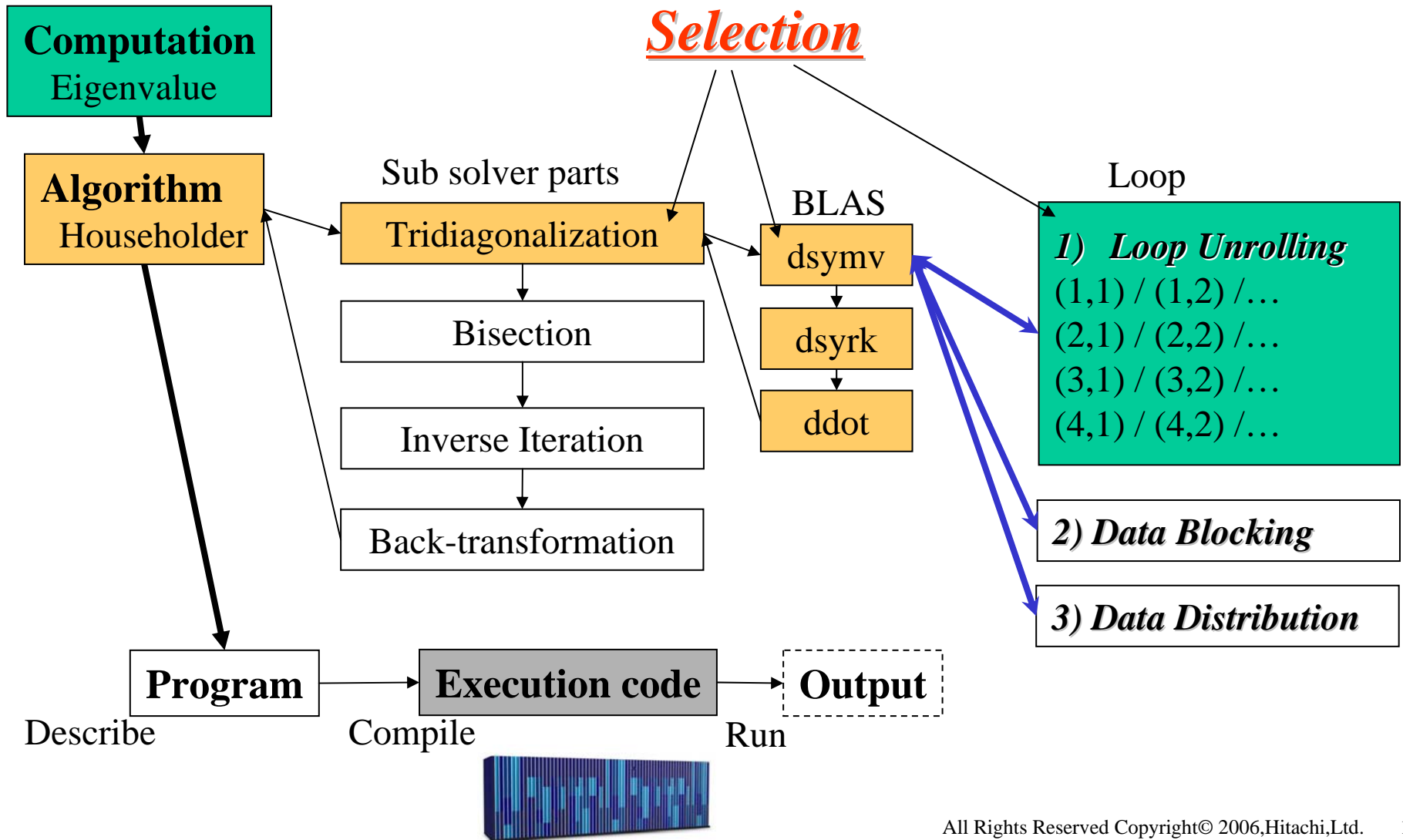
2 -4 Two viewpoints (1)

Matrix Software Hierarchy Viewpoint

Hierarchy	Examples
Solver	<i>Linear solver</i> $Ax = y$, <i>Eigensolver</i> $Av = \mu v$
Sub Solver	Tridiagonalization from A (full matrix) into T (tridiagonal matrix) Reorthogonalization from V (non-orthogonalized vectors) into V' (orthogonalized vectors)
BLAS	BLAS1 $x = x + y$, $a = (x, y)$ BLAS2 $y = Ax$, $y = Ax + A^t x$ BLAS3 $C = AB$, $A = A - yx^t - xy^t$
Loop	DO J = 1,100 DO I = 1,100 Y(J) = Y(J) + A(I,J)*X(I) ENDDO ENDDO

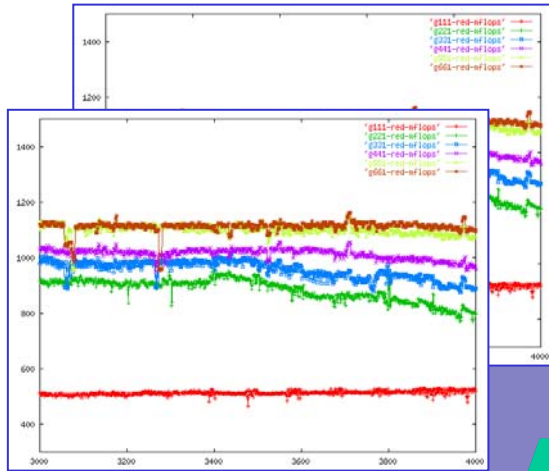
2 -5 Matrix Software Hierarchy Example

Selection of Eigenvalue Tuning



2 -6 Two Viewpoints (2)

Software Development Cycle Viewpoint



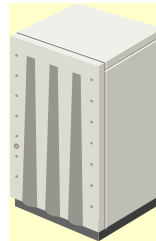
Evaluation

```
subroutine single_orthogonal(W,V,M)
do j=1, M-1
s = 0.0d0
do i=1, N
s = s + V(i, j)*w(i)
enddo
if (s .ne. 0.0d0) then
do i=1, N
w(i) = w(i) - s*V(i, j)
enddo
endif
enddo
```

```
subroutine single_orthogonal(W,V,M)
do j = 1, M-1
s = 0.0d0
do i = 1, N
s = s + V(i, j)*w(i)
enddo
hr(j) = s
Enddo
do i = 1, N
s = 0.0d0
do j = 1, M-1
s = s + V(i, j)*hr(j)
enddo
w(i) = w(i) - s
enddo
```

Programming

Compile and Run



Content

- 1 Introduction: Why is AT necessary?
- 2 Definition of AT and its two viewpoints
- 3 Existing AT classification**
- 4 Proposal of numerical policy interface
- 5 Preliminary results on sparse eigensolver
- 6 Conclusion

3 -1 Type of AT

- Library
 - ATLAS(1998), ILIB(1998)
- Script Language
 - PHiPAC(1997), ABCLibScript(2002)
- Framework
 - FIBER(2003), SANS(2003)
- Research (2004-) continues....

3 -2 PHiPAC

- Bilmes, et al.(1997)
- Loop coding techniques
 - Portable, High-Performance ANSI C Coding methodologies
- Automatic parameter search
 - Hiding instruction FPU latency
 - Loop unrolling parameter values defined from register to higher cache (L0 \rightarrow L1 \rightarrow L2 \rightarrow ...)
- Experience; DGEMM (BLAS) code on Sparc, SGI Indigo etc.

3 -3 ATLAS

- Whaley, et al.(1998)
- **A**utomatically **T**uned **L**inear **A**lgebra **S**oftware
- All BLAS functions
- Highly tuned for PC, RS/6000SP, Sun, Onyx,...
- Automatically tuned by loop unrolling, cache blocking, ..., but, empirically coded in some cases
- Wide range of performance sampling at installation
- Now used for wide area scientific computing

3 -4 ILIB

- Kuroda, et al.(1998)
- Intelligent LIBrary
- Solver function; GMRES(m), Eigensolver
- Automatic loop unrolling selection for matrix-vector multiplications
 - The depth varies 1, 2, 3, 4, and 8, for example
- Automatic orthogonalization algorithm selection
 - Run-time selection of Algorithms; MGS or CGS
- Automatic Preconditioning algorithm selection
- Experience; SR2201, SR8000

MGS = Modified Gram-Schmidt, CGS = Classical Gram-Schmidt

3 -5 ABCLibScript

- Katagiri, et al.(2002)
- Automatically **B**locking and **C**ommunication-adjustment **L**ibrary **S**cript
- Script for loop in matrix computations
- Automatic loop unrolling selection for any loop type
 - Optimization Model description; polynomial type
 - Performance sampling point description; loop indices
- Automatic source codes generation from the described model
- Experience; eigenvalue on SR8000, PC

3 -6 ABCLibscript Sample

```
!ABCLib$ install unroll (j) region start  
!ABCLib$ varied (j) from 1 to 16  
!ABCLib$ fitting polynomial 5 sampled (1-5, 8, 16)  
do j = 1, n  
  do i = 1, n  
    A(i, j) = A(i, j) * u(j) + V(i, j) * u(i)  
  enddo  
enddo
```

↓

Automatic source code generations with
Performance samplings

↓

Tuned Library Generated

3 -7 FIBER

- Katagiri, et al.(2003)
- Framework of **I**nstallation, **B**efore **E**xecution-invocation, and **R**un-time optimization
- Based on optimization framework;
 - Find tuning parameters (ex, unrolling pattern) satisfying the given conditions (ex, matrix size)
- Classifies the tuning parameters;
 - **IOP**, Installation Optimization Parameters
 - **BEOP**, Before Execution-invocation Optimization Parameters
 - **ROP**, Run-time Optimization Parameters
- Experience; eigenvalue on SR8000

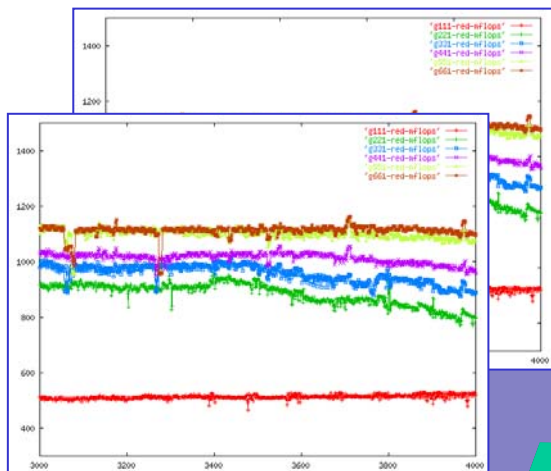
3 -8 SANS

- Dongarra, et al.(2003)
- Self-Adapting Numerical Software
- Plan-Do-See structured software architecture
 - SANS Agent
 - Method selector,
 - Simple scripting language
 - XML/CCA-based metadata
 - History database
 - System components
 - Manage of and access to the Grid
- Experience; not yet (in 2003)

3 -9 Matrix Software Hierarchy (2-4)

Hierarchy	Examples
Solver	<i>Linear solver</i> $Ax = y$, <i>Eigensolver</i> $Av = \mu v$
Sub Solver	Tridiagonalization from A (full matrix) into T (tridiagonal matrix) Reorthogonalization from V (non-orthogonalized vectors) into V' (orthogonalized vectors)
BLAS	BLAS1 $x = x + y, \alpha = (x, y)$ BLAS2 $y = Ax, y = Ax + A^t x$ BLAS3 $C = AB, A = A - yx^t - xy^t$
Loop	DO J = 1,100 DO I = 1,100 Y(J) = Y(J) + A(I,J)*X(I) ENDDO ENDDO

3 -10 Software Development Cycle (2-5)



Evaluation

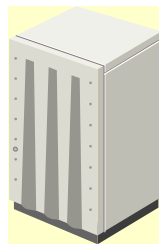
```
subroutine single_orthogonal(W,V,M)
do j=1, M-1
s = 0.0d0
do i =1, N
s = s + V(i, j)*w(i)
enddo
if (s .ne. 0.0d0) then
do i=1, N
w(i) = w(i) - s*V(i, j)
enddo
endif
enddo
```

subroutine single_orthogonal(W,V,M)

```
do j = 1, M-1
s = 0.0d0
do i = 1, N
s = s + V(i, j)*w(i)
enddo
hr(j) = s
Enddo
do i = 1, N
s = 0.0d0
do j = 1, M-1
s = s + V(i, j)*hr(j)
enddo
w(i) = w(i) - s
enddo
```

Programming

Compile and Run



3 -11 Trends of AT for Matrix Computations

AT (YEAR)	PHiPAC (1997)	ATLAS (1998)	ILIB (1998)	ABCLib Script(02)	FIBER (2002)	SANS (2003)
Type	Script	Library	Library	Script	FW	FW
Programming (Solver)	—	—	—	—	—	—
Programming (Sub solver)	—	—	△	—	△	—
Programming (BLAS)	△	◎	○	○	○ (ABC.)	◎ (ATLAS)
Programming (Loop)	◎	○	○	◎	○ (ABC.)	○ (ATLAS)
Compiling	—	—	—	—	○	—
Run	—	—	○	—	○	○
Evaluation (DB)	—	—	—	—	—	○ (XML/CCA)
Platform	Single Proc.	Single Proc.	MPPs, C. of SMPs	MPPs, C. of SMPs	MPPs, C. of SMPs	GRID

3 -12 Trend towards higher level semantics

AT (YEAR)	PHiPAC (1997)	ATLAS (1998)	ILIB (1998)	ABCLib Script(02)	FIBER (2002)	SANS (2003)
Type	Script	Library	Library	Script	FW	FW
Programming (Solver)	—	—	—	—	—	—
Programming (Sub solver)	—	—	△	—	△	—
Programming (BLAS)	△	⊙	○	○	○ (ABC.)	⊙ (ATLAS)
Programming (Loop)	⊙	○	○	⊙	○ (ABC.)	○ (ATLAS)
Compiling	—	—	—	—	○	—

1) From loop or BLAS level to sub solver level.

Step by step, the automatic tuning semantics is going higher. This is because higher semantics can make use of better performance with less information.

3 -13 Trend towards total development cycle

From programming stage to whole stages of software development cycle.

Recent automatic tuning methods, FIBER and SANS, are more conscious of software development cycle. For example, when automatic tuning for matrix computation targets sparse matrices, sparse solver performance is largely depends on the data of the matrix. Therefore, the feedback will be an important information for the later computations.

Programming (Loop)	◎	○	○	◎	○ (ABC.)	○ (ATLAS)
Compiling	—	—	—	—	○	—
Run	—	—	○	—	○	○
Evaluation (DB)	—	—	—	—	—	○ (XML/CCA)
Platform	Single Proc.	Single Proc.	MPPs, C. of SMPs	MPPs, C. of SMPs	MPPs, C. of SMPs	GRID


3 -14 Trend towards wide range of platforms

AT (YEAR)	PHiPAC (1997)	ATLAS (1998)	ILIB (1998)	ABCLib Script(02)	FIBER (2002)	SANS (2003)
Type	Script	Library	Library	Script	FW	FW
Programming (Solver)	—	—	—	—	—	—
Programming (Sub solver)	—	—	△	—	△	—
Programming	△	◎	○	○	○	◎

From single CPU to wider platform availability including Grid.

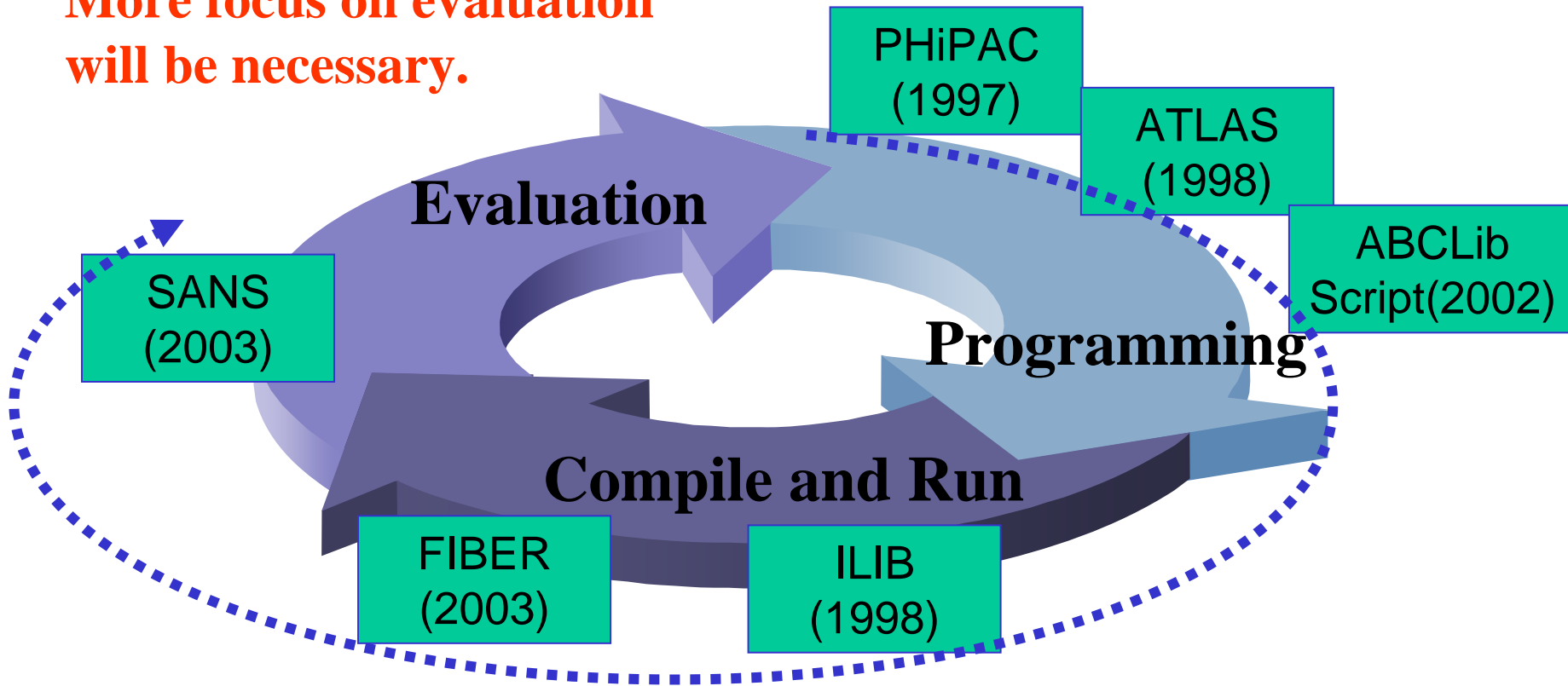
PHiPAC and ATLAS target single CPU performance, while ILIB, ABCLibScript, and FIBER aim at parallel platforms, like MPPs, cluster of SMPs. SANS targets the Grid. Newer research target automatic tuning on wider platforms

Evaluation (DB)						(XML/CCA)
Platform	Single Proc.	Single Proc.	MPPs, C. of SMPs	MPPs, C. of SMPs	MPPs, C. of SMPs	GRID



3 -15 Notes of SDC

More focus on evaluation
will be necessary.



SDC = Software Development Cycle

Content

- 1 Introduction: Why is AT necessary?
- 2 Definition of AT and its two viewpoints
- 3 Existing AT classification
- 4 Proposal of numerical policy interface**
- 5 Preliminary results on sparse eigensolver
- 6 Conclusion

4 -1 Problems of the existing methods

- Balance is quite important for matrix libraries, especially, of iterative methods.
 - Computing time and accuracy
 - Computing accuracy and computing resource
- So far, no parameter control of the tradeoff between computing time and accuracy.
 - Algorithm selection in ILIB, but, no explicit form of control the balance.

4 -2 Definitions of numerical policy

Def. 1: UCP = User Control Parameter

Def. 2: ICP = Internal Critical Parameter

Def. 3: RCP = Resource Control Parameter

Def. 4: UOF = User Objective Function

Definition of AT

Given UCP and RCP, find ICP to minimize
$$\text{UOF} = \text{UOF}(\text{UCP}, \text{ICP}, \text{RCP}).$$

Definition of Policy

Given UCP and RCP, find ICP to balance UOFs.

$$\text{UOF1} = \text{UOF1}(\text{UCP}, \text{ICP}, \text{RCP}),$$

$$\text{UOF2} = \text{UOF2}(\text{UCP}, \text{ICP}, \text{RCP}), \dots$$

4 -3 Examples of numerical policy

1) Minimize computing time (UOF1) allowing computational error (UOF2) to be large.

Time = Time(N, Iter, Ncpu)

Residual = Residual(N, Iter)

2) Minimize computing time (UOF1) allowing number of CPUs (RCP) to be ≤ 1 .

4 -4 Example of policy problem

AT of Iteration number is Iterative method;

ICP = Iteration number, UOF = Computing time,
RCP = Number of CPUs, UCP = Matrix size,
$$\text{UOF} = \text{UOF}(\text{UCP}, \text{ICP}, \text{RCP})$$

<AT problem>

A user wants to minimize UOF with $\text{UCP} = 10000$, $\text{RCP} = 4$.
Find the best ICP value.

<Policy problem>

A user wants to minimize UOF with $\text{UCP} = 10000$, $\text{RCP} = 1$, or 2 , or 4 .
Also, the user wants to balance UOF and RCP.
Find the best ICP value.

Content

- 1 Introduction: Why is AT necessary?
- 2 Definition of AT and its two viewpoints
- 3 Existing AT classification
- 4 Proposal of numerical policy interface
- 5 Preliminary results on sparse eigensolver**
- 6 Conclusion

5 -1 Target

1) Target problem

Standard eigenvalue problem of symmetric sparse matrix A with size N that is over 10,000.

$$Av = ev$$

e - eigenvalue, v - eigenvector.

2) Target solver; Lanczos method

Main calculation

- Sparse matrix vector multiplications - today's focus.
- Reorthogonalizations - in another paper.

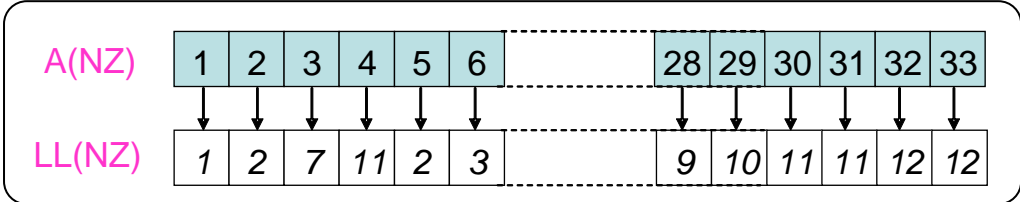
5 -2 Sparse matrix multiplications

Matrix elements are stored in A(NZ) row-wise with nonzero indices LL(NZ) and diagonal indices LC(N+1).

	1	2	3	4	5	6	7	8	9	10	11	12	
1	1	2					3				4		1
2		5	6				7	8	9				2
3			10						11		12	13	3
4				14		15						16	4
5					17	18				19	20		5
6						21					22	23	6
7							24	25					7
8								26	27				8
9									28				9
10										29	30		10
11											31	32	11
12												33	12

- LC(N+1)
- LC(1) = 1
 - LC(2) = 5
 - LC(3) = 10
 - LC(4) = 14
 - LC(5) = 17
 - LC(6) = 21
 - LC(7) = 24
 - LC(8) = 26
 - LC(9) = 28
 - LC(10) = 29
 - LC(11) = 31
 - LC(12) = 33
 - LC(13) = 34

Diagonal indices
LC(N+1)



Nonzero indices
LL(NZ)

5 -3 SMP implementation

Implementation of sparse matrix vector multiplication: $r = Ax$

```
# Program 1 (Single processing implementation)
REAL*8 A(NZ), LL(NZ), LC(N+1), R(N), X(N)
```

```
DO I = 1, N
  S = A(LC(I))*X(I)
  DO JC = LC(I)+1, LC(I+1)-1
    JJ = LL(JC)
    S = S + A(JC) * X(JJ)
    R(JJ) = R(JJ) + A(JC) * X(I)
  ENDDO
  R(I) = R(I) + S
ENDDO
```

	1	2	3	4	5	6	7	8	9	10	11	12	
1	1	2					3				4		1
2		5	6				7	8	9				2
3			10						11	12	13		3
4				14	15							16	4
5					17	18				19	20		5
6						21					22	23	6
7							24	25					7
8								26	27				8
9									28				9
10										29	30		10
11											31	32	11
12												33	12

LC(N+1)
LC(1) = 1
LC(2) = 5
LC(3) = 10
LC(4) = 14
LC(5) = 17
LC(6) = 21
LC(7) = 24
LC(8) = 26
LC(9) = 28
LC(10) = 29
LC(11) = 31
LC(12) = 33
LC(13) = 34

A(NZ)	1	2	3	4	5	6	...	28	29	30	31	32	33
LL(NZ)	1	2	7	11	2	3	...	9	10	11	11	12	12

Cannot be SMP-parallelized

5 -4 SMP implementation (2)

Program 2 (parallel processing implementation on SMP)

```
REAL*8 A(NZ), LL(NZ), LC(N+1), R(N), X(N), RR(N,16)
DO IP = 1, 16
  DO I = (IP-1)*N/16+1, IP*N/16
    XX = X(I)
    S = A(LC(I)) * XX
    DO JC = LC(I)+1, LC(I+1)-1
      JJ = LL(JC)
      S = S + A(JC) * X(JJ)
      RR(JJ, IP) = RR(JJ, IP) + A(JC) * XX
    ENDDO
    R(I) = R(I) + S
  ENDDO
ENDDO
```

```
DO I = 1, N
  S = R(I)
  DO IP = 1, 16
    S = S + RR(I, IP)
  ENDDO
  R(I) = S
ENDDO
```

Later, R is taken
as the sum of
RRs.

Working space for 16 CPUs (on
SR11000) in SMP parallelization

5 -5 Numerical policies of sparse eigensolver

Policy 1

“I want to reduce memory allocation as much as possible, even if it would result in a longer computation time.”

Then, RR should be eliminated and one CPU computation should be executed.

Policy 2

“I want to reduce computation time as much as possible, even it would result in a larger amount of memory use.”

Then, RR should be $16N$ and $16CPU$ computations should be executed.

How should the following cases be determined?

Policy 3

“I want to reduce computation time as much as possible, under the condition of a certain amount of memory use.”

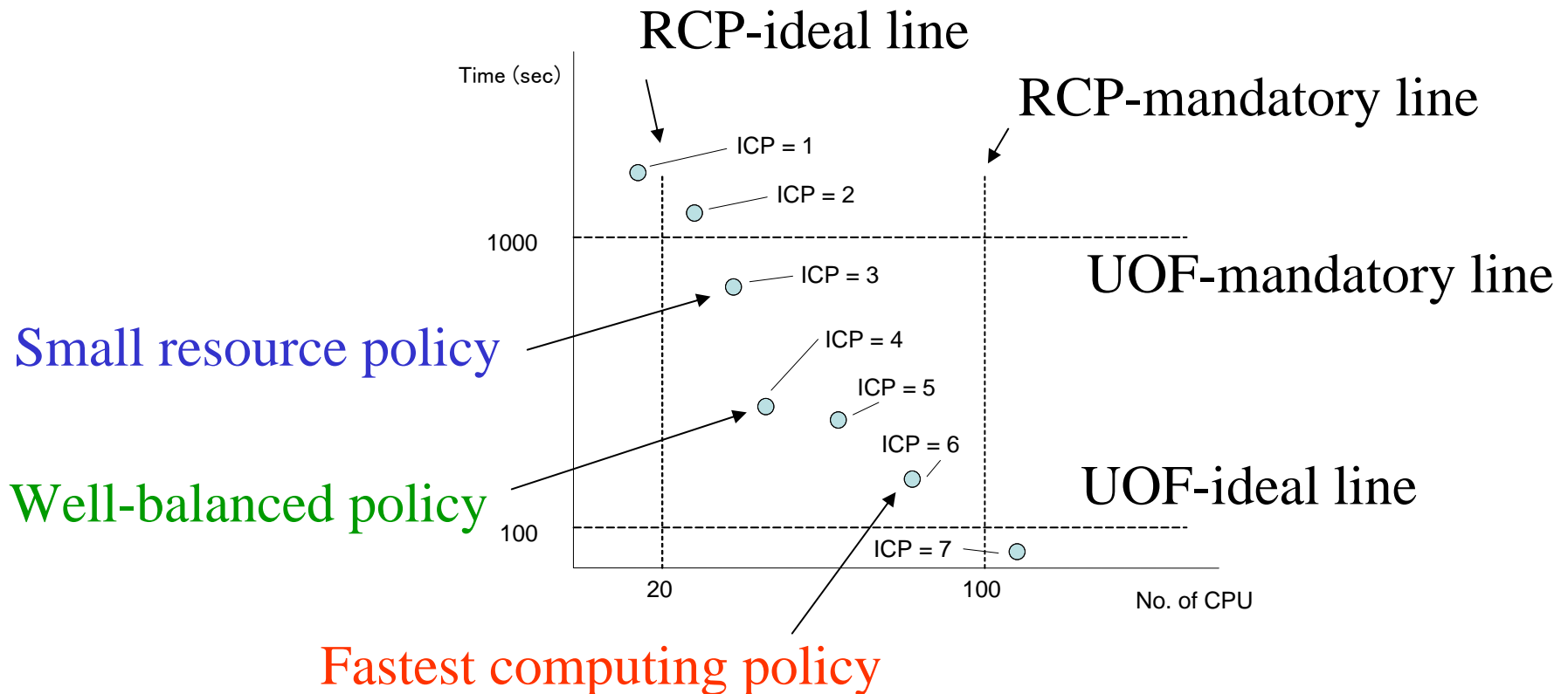
Policy 4

“I want to balance between amount of memory use and computation time, and I can use 4 CPUs. Which working space is better, $4N$ or $16N$?”

5 -6 Numerical policy interface of sparse eigensolver

An interface for numerical policy set (RCP, UOF)

Limit (mandatory) line, and the Best (ideal) line



5 -7 Numerical policy interface example

Example of script language implementation of numerical policy

Library name processed by assigned policy

No. of CPU is set equal to RCP, where the limit is 100 while the best is 20.

```
1 $ POLICY libname = SOLVER-A
2 $ POLICY RCP = CPU, CPU_limit = 100, CPU_best = 20
3 $ POLICY UOF = time, time_limit = 1000sec, time_best = 100 sec
4 $ POLICY selection = min; time/time_best + CPU/CPU_best
5 $ POLICY tune ICP = No_Iter
6   CALL SOLVER-A (N, No_Iter, No_CPU)
```

Policy description

Computing time is set to UOF, where the limit is 1000 sec while the best is 100 sec.

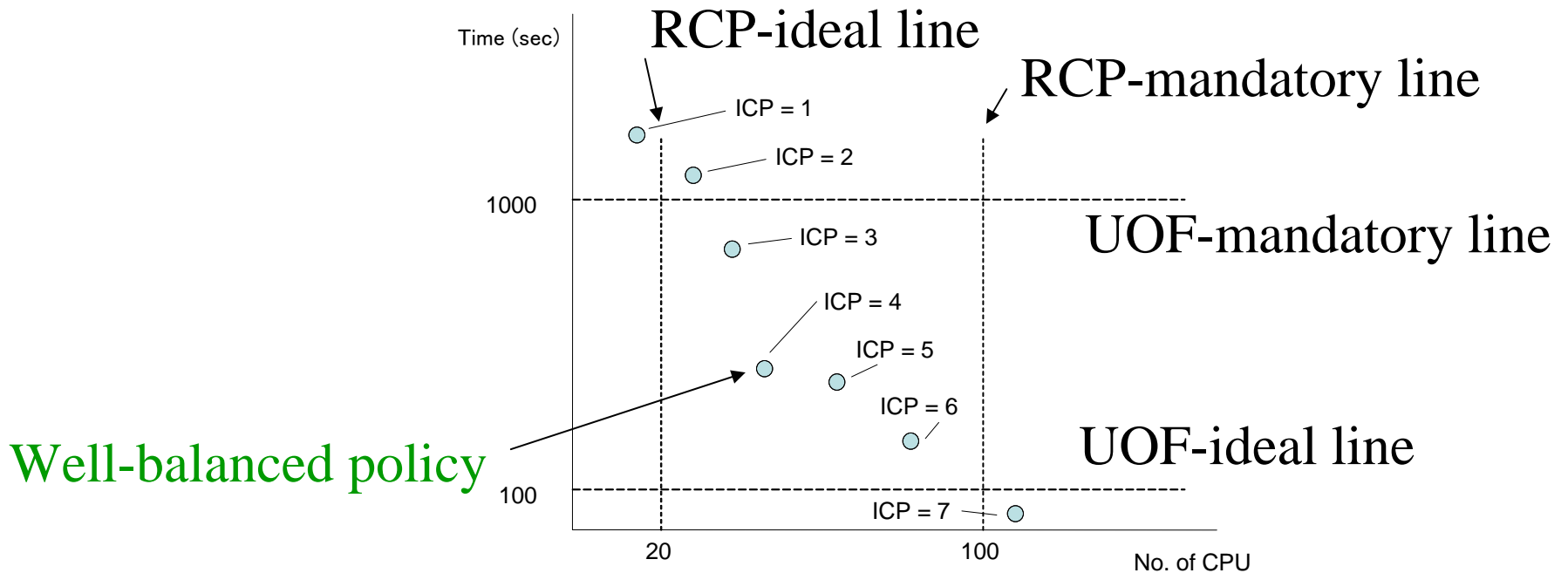
Target parameter to be tuned

5 -8 When to define the ICP?

When to solve the optimization problem (i.e., select the best ICP)?

Installation: Mainly used ICPs are executed. The results are stored.

Before-execution: Mandatory and ideal lines are defined. The best ICP is selected according to a given policy.



5 -9 Data and computing platform

MatrixMarket sparse matrices

Case	Name	Matrix size (N)	Non-zero no. (NZ)	Sparsity (NZ/(N*N))
1	bcsstk39.rsa	46,772	1,068,033	0.0488%
2	ct20stif.rsa	52,329	1,375,396	0.0502%
3	Pwtk.rsa	217,918	5,926,171	0.0125%

Computing platform

Name	Super technical server Hitachi SR11000
Hardware	1 node (16 CPU), CPU: Power5 1.9 GHz, main memory: 128 GB L1: 32 KB, L2: 1.875 MB, L3: 288 MB,
Compiler	Hitachi FORTRAN for SR11000
Compile option	f90 -64 -Os -noscope -parallel (same for 1CPU, 4CPUs, and 16CPUs). At the execution for 16CPUs, %>a.out -F'PRUNST(THREADNUM(16))'

5 -10 Policies and ICP for sparse eigensolver

1) Fastest computing policy

$$\mathbf{UOF1 = time}$$

2) Small memory policy

$$\mathbf{UOF2 = memory}$$

3) Well-balanced policy

$$\mathbf{UOF3 = memory/memory_best + time/time_bset}$$

ICP, working space for vectors RR (in program 2)

ICP = 1, No space for RR

ICP = 2, four-vector space (4N) for RR

ICP = 3, sixteen-vector space (16N) for RR

5 -11 Model to calculate limit and best lines

Nonzeros of matrix A

Best line of memory use:

$$\text{Memory_best} = 8 * \text{NZ} + 8 * \text{NV} * \text{N}$$

NV eigenvectors

Limit line of memory use is set 3 times greater than best line.

$$\text{Memory_limit} = 3 * \text{memory_best}$$

5 -12 Model to calculate limit and best lines

Best line of time is set under the following assumption.

Calculation executed on SR11000 (16 CPUs) with 25% of the peak performance, and the Lanczos iteration number is 100.

$$\text{Time_best} = 100 * (5 * 2 * \text{NZ} + 3 * 2 * \text{KMAX} * \text{N}) / (1900 \text{Mflop/s} * 16)$$

5 sparse mat-vec multiplications

3 reorthogonalizations

16 CPUs

25% of peak performance of SR11000 (1 CPU)

5 -13 Model to calculate limit and best lines

Limit line of time is set under the following assumption.

Calculation executed on SR11000 (1 CPU, even with 16 CPUs) with 25% of peak performance, and the Lanczos iteration number is 200.

$$\text{Time_limit} = 200 * (5 * 2 * \text{NZ} + 3 * 2 * \text{KMAX} * \text{N}) / (1900 \text{Mflop/s} * 1)$$

5 sparse mat-vec multiplications

3 reorthogonalizations

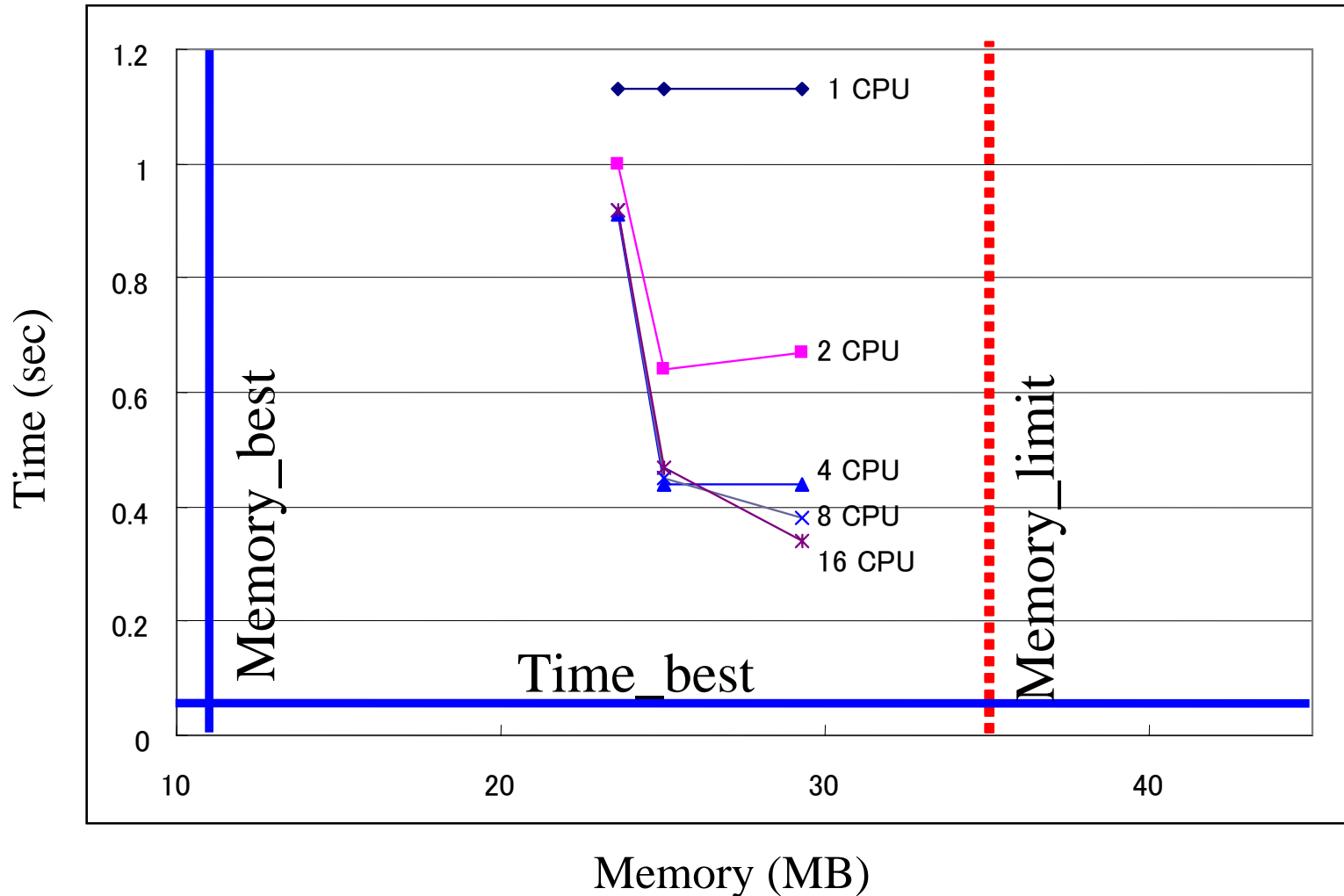
Iteration number

25% of peak performance of SR11000 (1 CPU)

even with 16 CPUs

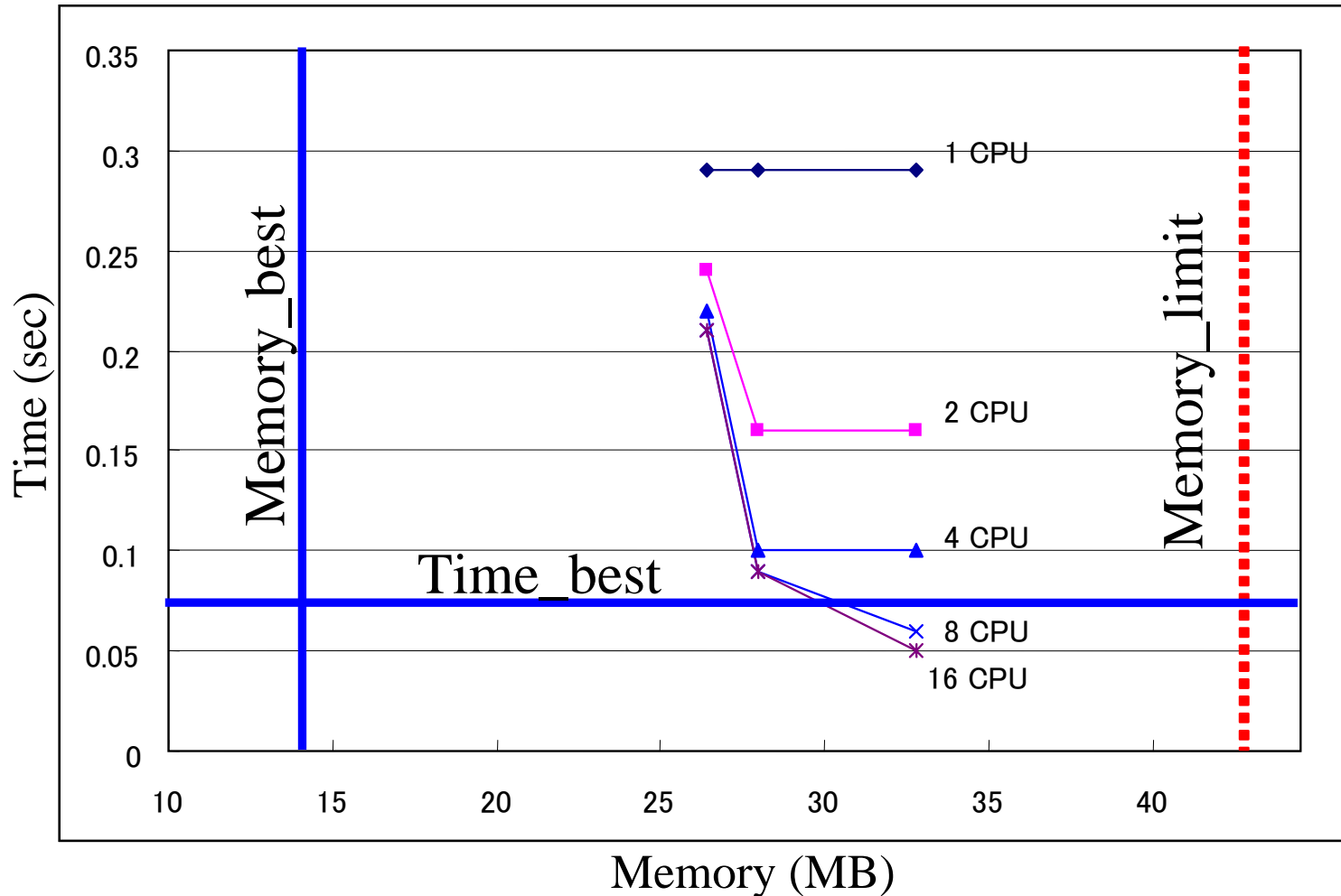
5 -14 Result for case 1 (bcsstk39.rsa)

Time_limit (over 2 sec)



5 -15 Result for case 2 (ct20stif.rsa)

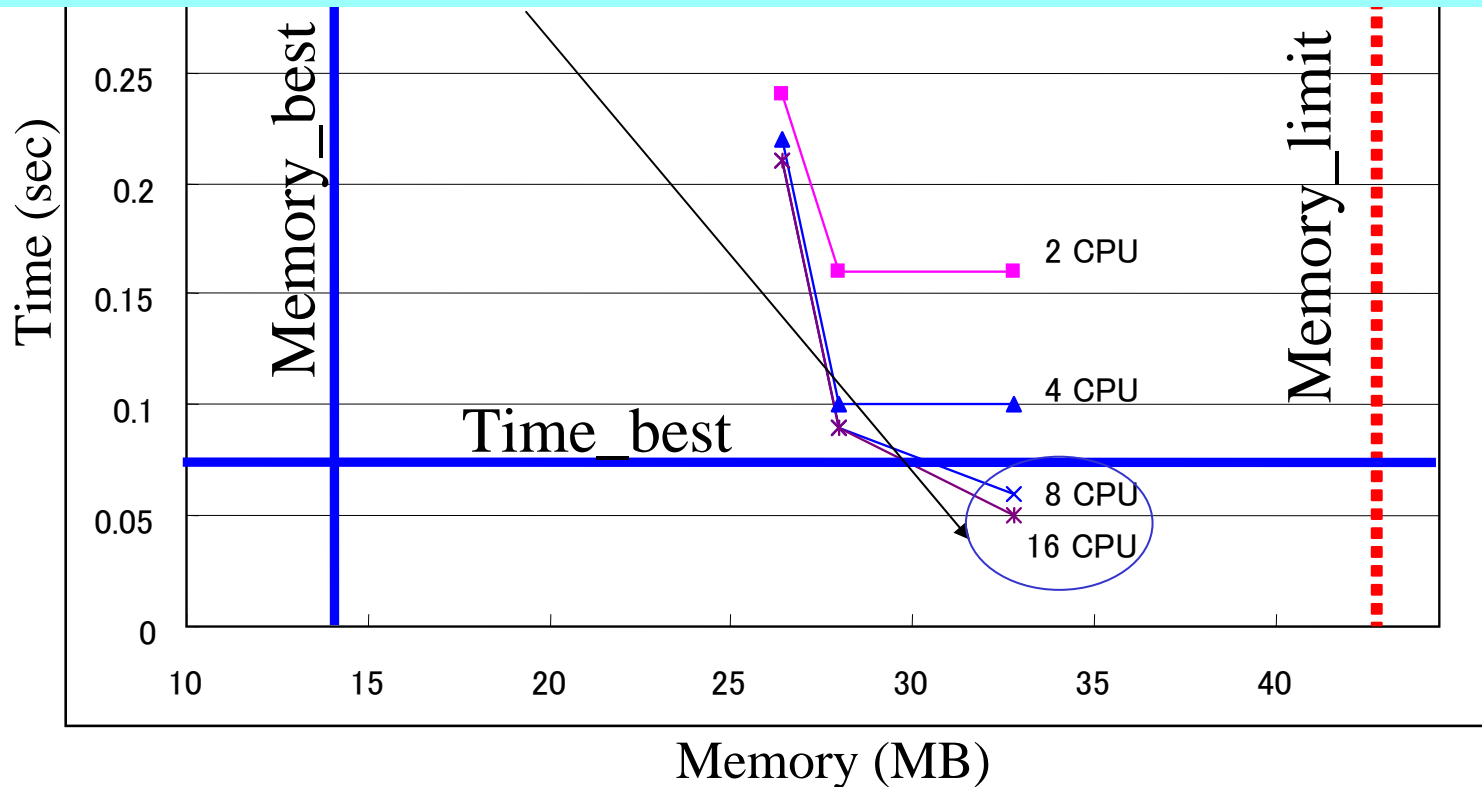
Time_limit (over 2 sec)



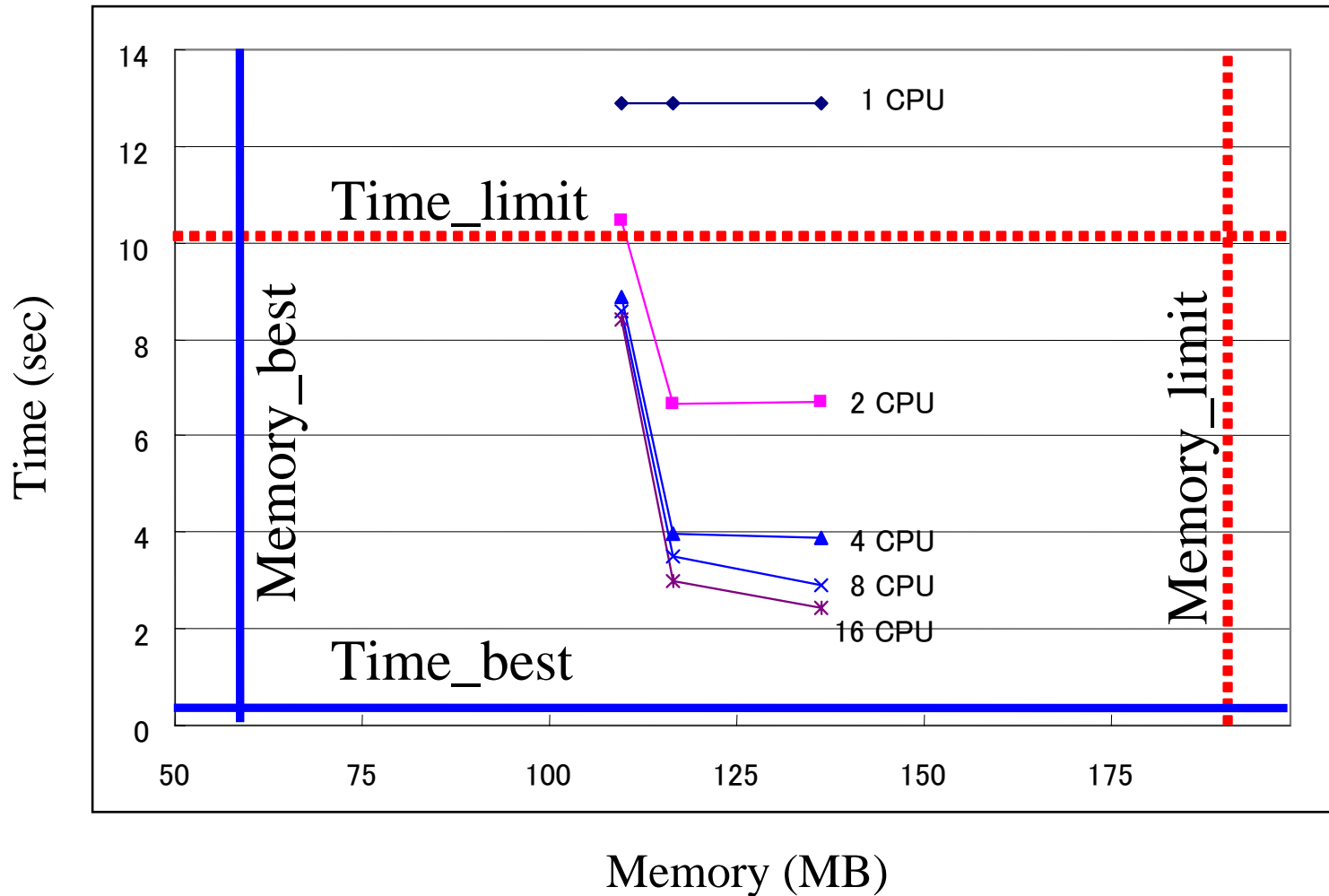
5 -15 Result for case 2 (ct20stif.rsa)

Time_limit (over 2 sec)

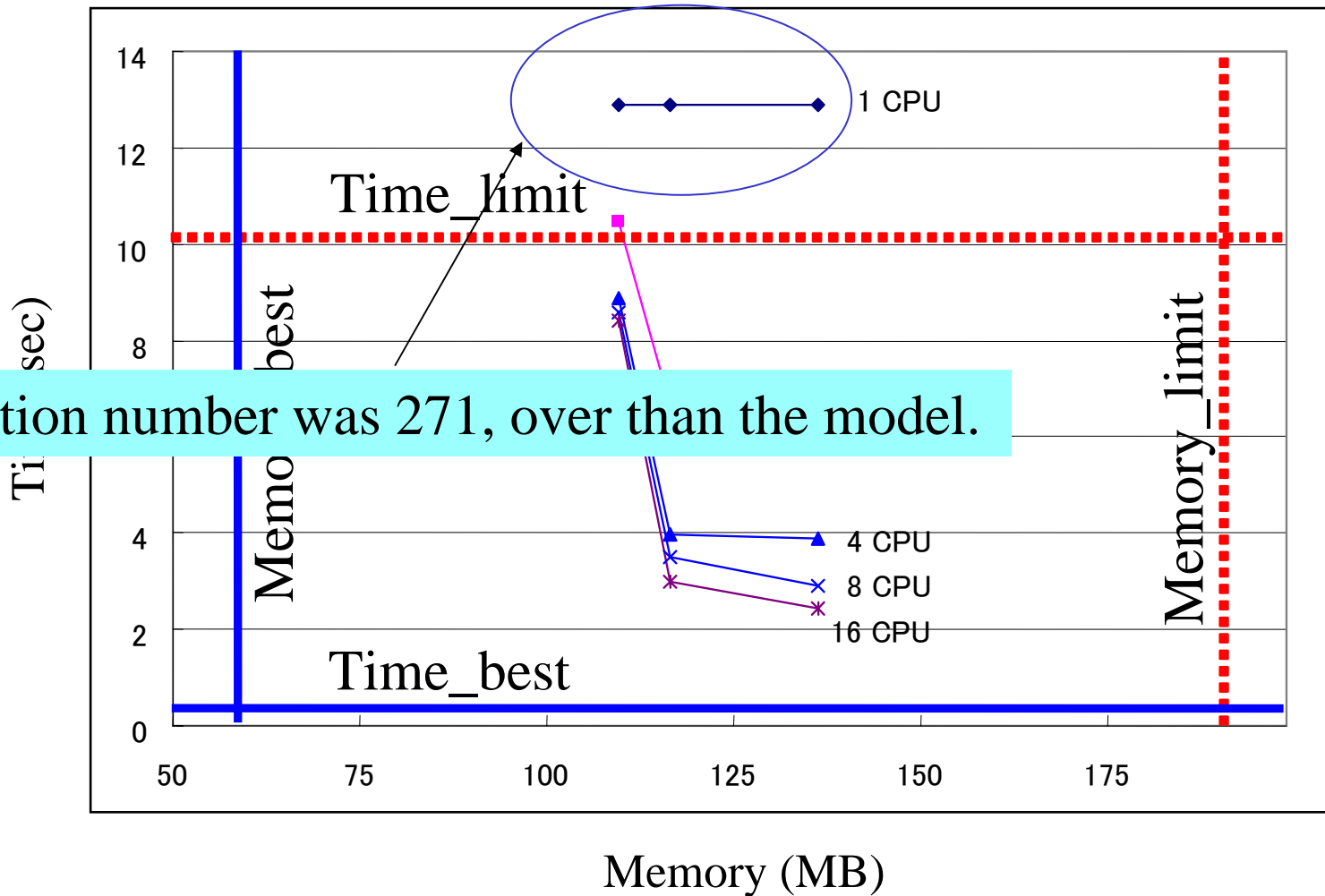
Actual iteration number was only 31, while 137 for case 1 and 271 for case 3.



5 -16 Result for case 3 (Pwtk.rsa)



5 -16 Result for case 3 (Pwtk.rsa)



5 -17 Preliminary result for optimized ICP

Case	No. of CPUs	Numerical policies		
		Fastest computing	Small memory	Well-balanced (time/memory)
1	1	ICP = 1	ICP=1	ICP = 1 (0%/0%)
	2	ICP = 2	ICP=1	ICP = 2 (0%/6.1%)
	4	ICP = 2	ICP=1	ICP = 2 (0%/6.1%)
	8	ICP = 3	ICP=1	ICP = 3 (0%/24.2%)
	16	ICP = 3	ICP=1	ICP = 3 (0%/24.2%)
2	1	ICP = 1	ICP=1	ICP = 1 (0%/0%)
	2	ICP = 2	ICP=1	ICP = 1 (0%/24.2%)
	4	ICP = 2	ICP=1	ICP = 2 (0%/24.2%)
	8	ICP = 3	ICP=1	ICP = 2 (50.0%/6.1%)
	16	ICP = 3	ICP=1	ICP = 2 (50.0%/6.1%)
3	1	NG	NG	NG
	2	ICP = 2	ICP=2	ICP = 2 (0%/0%)
	4	ICP = 3	ICP=1	ICP = 2 (1.8%/6.1%)
	8	ICP = 3	ICP=1	ICP = 2 (20.0%/6.1%)
	16	ICP = 3	ICP=1	ICP = 2 (22.1%/6.1%)

5 -17 Preliminary result for optimized ICP

Case	No. of CPUs	Numerical policies		
		Fastest computing	Small memory	Well-balanced (time/memory)
1	1	ICP = 1	ICP=1	ICP = 1 (0%/0%)
	2	ICP = 2	ICP=1	ICP = 2 (0%/6.1%)
2	2	ICP = 2	ICP=1	ICP = 1 (0%/24.2%)
	4	ICP = 2	ICP=1	ICP = 2 (0%/24.2%)
	8	ICP = 3	ICP=1	ICP = 2 (50.0%/6.1%)
	16	ICP = 3	ICP=1	ICP = 2 (50.0%/6.1%)
	16	ICP = 3	ICP=1	ICP = 2 (50.0%/6.1%)
3	1	NG	NG	NG
	2	ICP = 2	ICP=2	ICP = 2 (0%/0%)
	4	ICP = 3	ICP=1	ICP = 2 (1.8%/6.1%)
	8	ICP = 3	ICP=1	ICP = 2 (20.0%/6.1%)
	16	ICP = 3	ICP=1	ICP = 2 (22.1%/6.1%)

In well-balanced policy;

- Computing time: 50% longer than fastest computing policy.
- Amount of memory: 6.1% larger than small memory policy.

5 -17 Preliminary result for optimized ICP

Case	No. of CPUs	Numerical policies		
		Fastest computing	Small memory	Well-balanced (time/memory)
1	1	ICP = 1	ICP=1	ICP = 1 (0%/0%)
	2	ICP = 2	ICP=1	ICP = 2 (0%/6.1%)

In well-balanced policy:

- Computing time: 1.8 - 22.1% longer than fastest computing policy.
- Amount of memory: 6.1% larger than small memory policy

	2	ICP = 2	ICP=1	ICP = 1 (0%/24.2%)
	4	ICP = 2	ICP=1	ICP = 2 (0%/24.2%)
	8	ICP = 3	ICP=1	ICP = 2 (50.0%/6.1%)
	16	ICP = 3	ICP=1	ICP = 2 (50.0%/6.1%)
3	1	NG	NG	NG
	2	ICP = 2	ICP=2	ICP = 2 (0%/0%)
	4	ICP = 3	ICP=1	ICP = 2 (1.8%/6.1%)
	8	ICP = 3	ICP=1	ICP = 2 (20.0%/6.1%)
	16	ICP = 3	ICP=1	ICP = 2 (22.1%/6.1%)

Content

- 1* Introduction: Why is AT necessary?
- 2* Definition of AT and its two viewpoints
- 3* Existing AT classification
- 4* Proposal of numerical policy interface
- 5* Preliminary results on sparse eigensolver
- 6* Conclusion**

6 -1 Summary

Matrix library with *numerical policy interface*, which balances between memory allocation and computing time, is proposed.

The library with the policy selects the ICP by the following methods.

- 1) The interface has the input of the limit (mandatory) line and the best line for memory allocation and computing time.
- 2) The function defines the ICP by the optimization modeled by the best line within the scope of both limit lines, based on the repository data.

- The method is applied to selection of working space of a sparse eigensolver for parallelization on SMP.
- Then, the method can select a well-balanced working space for solving matrices obtained from MatrixMarket.

6 -2 Future work

(1) Implementations of optimization problem

- More precise prediction model of best and limit lines
- More and more tests and preparation of data repository

(2) Extensions

- Extension of policy using other parameters for Lanczos
 - Working space for reorthogonalizations
 - Number of CPUs
- Extension of library
 - Arnoldi
 - GMRES
 - BiCGStab
- Extension of platforms
 - MPPs
 - Cluster of SMPs

HITACHI

Inspire the Next t