TOWARDS AUTO-TUNING FRAMEWORK FOR NUMERICAL LIBRARIES

Takahiro Katagiri

Information Technology Center,

The University of Tokyo

First French-Japanese Workshop
- Petascale Application, Algorithms and Programming (PAAP) -
December 1st, 2007, 2:10pm – 2:40pm

OUTLINES

- Motivation
- Our Solutions
  - FIBER : An Auto-tuning Framework
  - ABCLibScript: An Auto-tuning Description Language
  - ABCLib: A Library with Auto-tuning Facility
    - ABCLib_DRSSED: An Eigenvalue Solver
    - MS-MPI Run-time Auto-tuning Project
- Related Projects
- Conclusion Remarks

To establish high productivity on numerical software

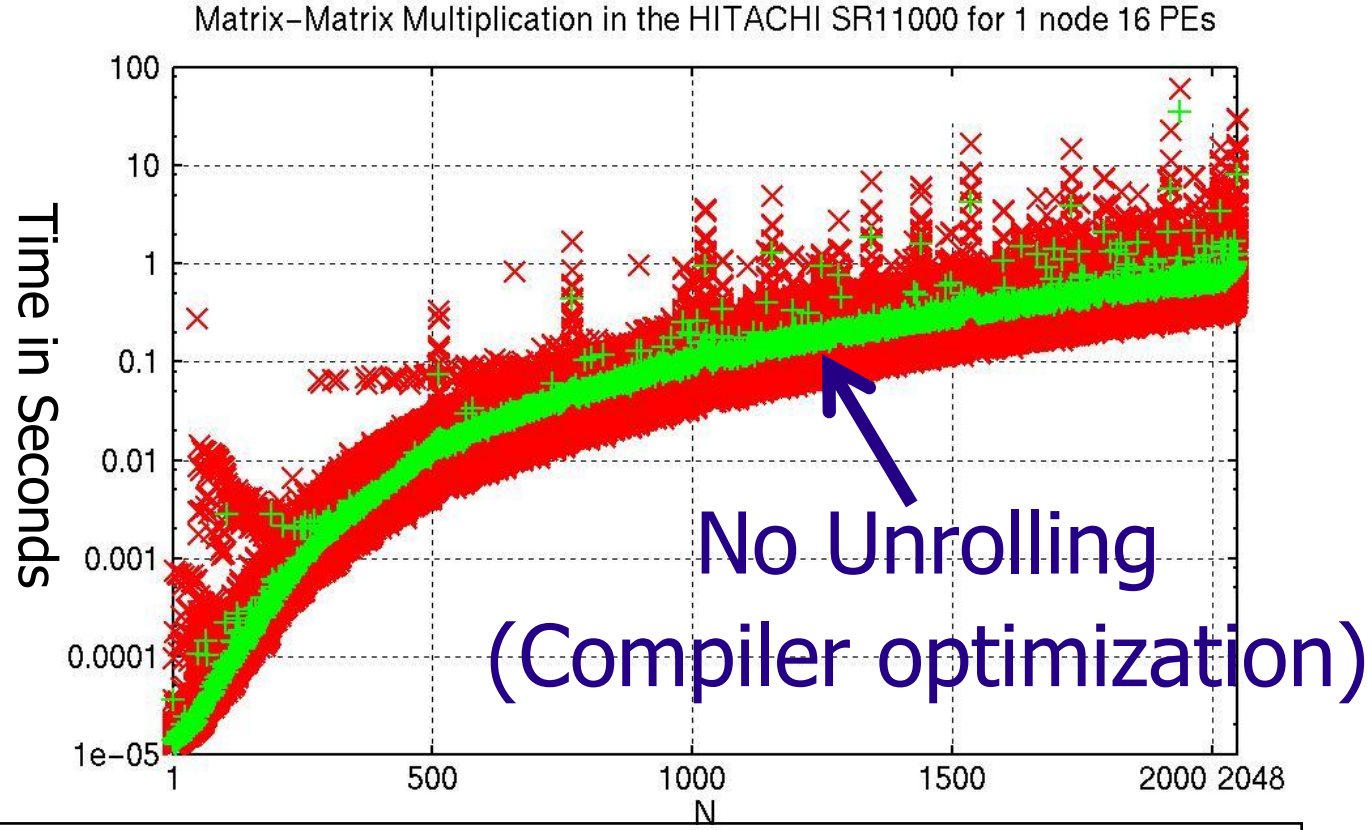
MOTIVATION

HIGH COST OF NUMERICAL SOFTWARE DEVELOPMENT

- Why so high cost?
  1. Explosion of search space for tuning parameters
     - Excessive development processes
  2. Tuning is not science, but craftspeople work…
     - Excessive personnel costs

1. Excessive development processes
   - Many algorithm parameters
     - Preconditioner, restart frequency, block algorithm length, …
   - Complex current computer architectures
     - multicore, unsymmetrical memory access,…
2. Excessive personnel costs
   - Intricate high performance implementations
     - Craftspeople only can do it.
   - Compilers do not work well on the complex current computers….

AN EXAMPLE: TUNING DIFFICULTY



- Unrolled coeds for matrix-matrix multiplication with nested 3 loops (i,j,k) from 1 to 4.
  - The variation is 4*4*4=64 kinds.
  - For matrix size N, it varies from 1 to 2048 stridden 1.
- Compiler: HITACHI Optimized Fortran90. Option: -Oss with automatically parallelization.
- Machine: HITACHI SR11000/J2 Model installed in Information Technology Center, The University of Tokyo. It has 16PEs per node.

- Averaged gap: 10x. Dedicated sizes: 100x.
- How should we manage it?

NEED AUTO-TUNING FACILITY

1. To reduce tuning processes:
   - **Automation of tuning can reduce the tuning process to hand-tuning.**
     - Tuning is time-consuming work even in craftsman.
       - Writing complicated codes.
       - Troublesome test-run to tune
2. To reduce personnel cost:
   - **“Automatic Tuning Recipe” makes tuning non-expert work.**
     - Software Framework
     - Auto-tuning facility
     - Computer language for non-expert developers
     - Source code generator
       - Tuning object codes and tuning control codes

FIBER, ABCLib, ABCLibScript

OUR SOLUTIONS

AUTO-TUNING FACILITY ON SOFTWARE LAYERS

Linear Equations Solvers

Eigenvalue Solvers

Sparse Direct Solvers

BLAS

...

Library Interface

Compilers

Communication Libraries (MPI)

Performance Parameters

Optimization Codes & Info.

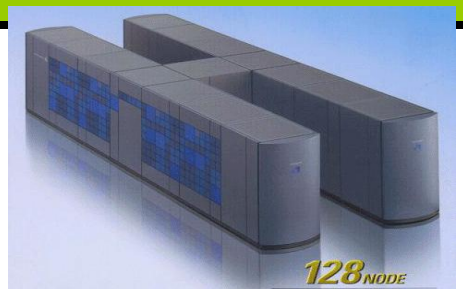
Implementation Info.

Scheduling & Computer Info.

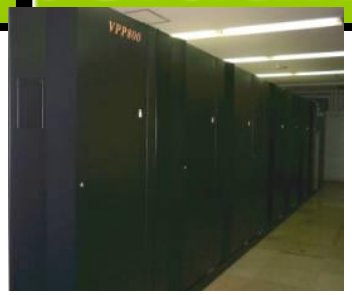
Auto-tuning Facility

- Auto-modeling Funct.
- Code generation Funct.
- Parameter Opt. Funct.

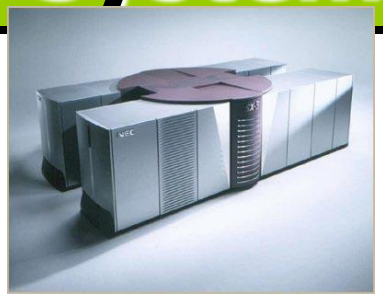
Operating Systems



HITACHI SR



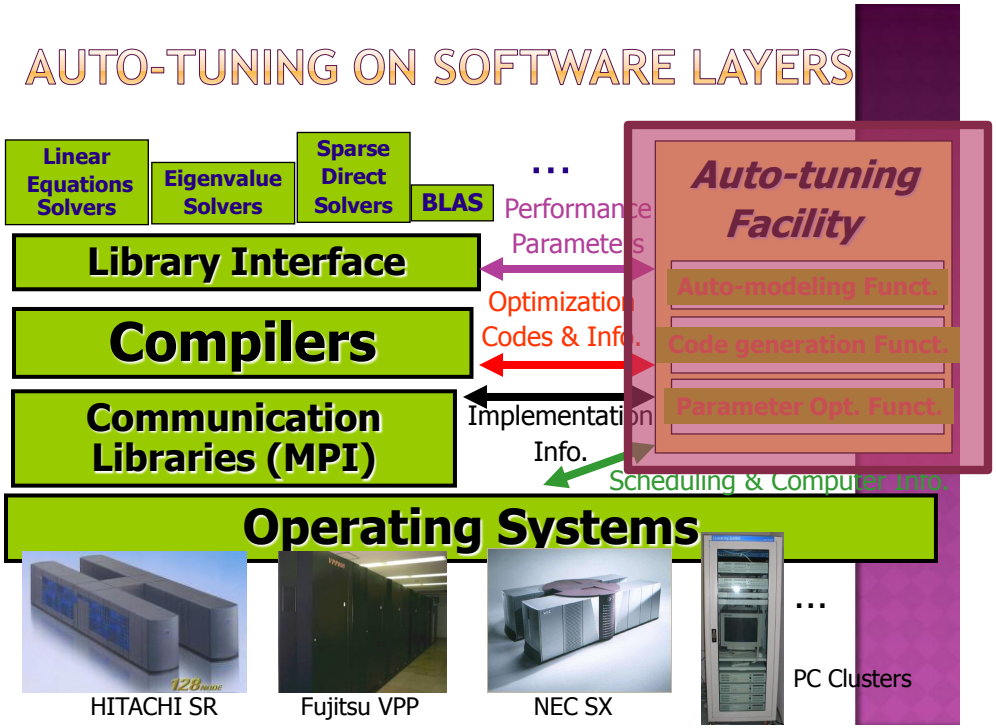
Fujitsu VPP



NEC SX

...

PC Clusters



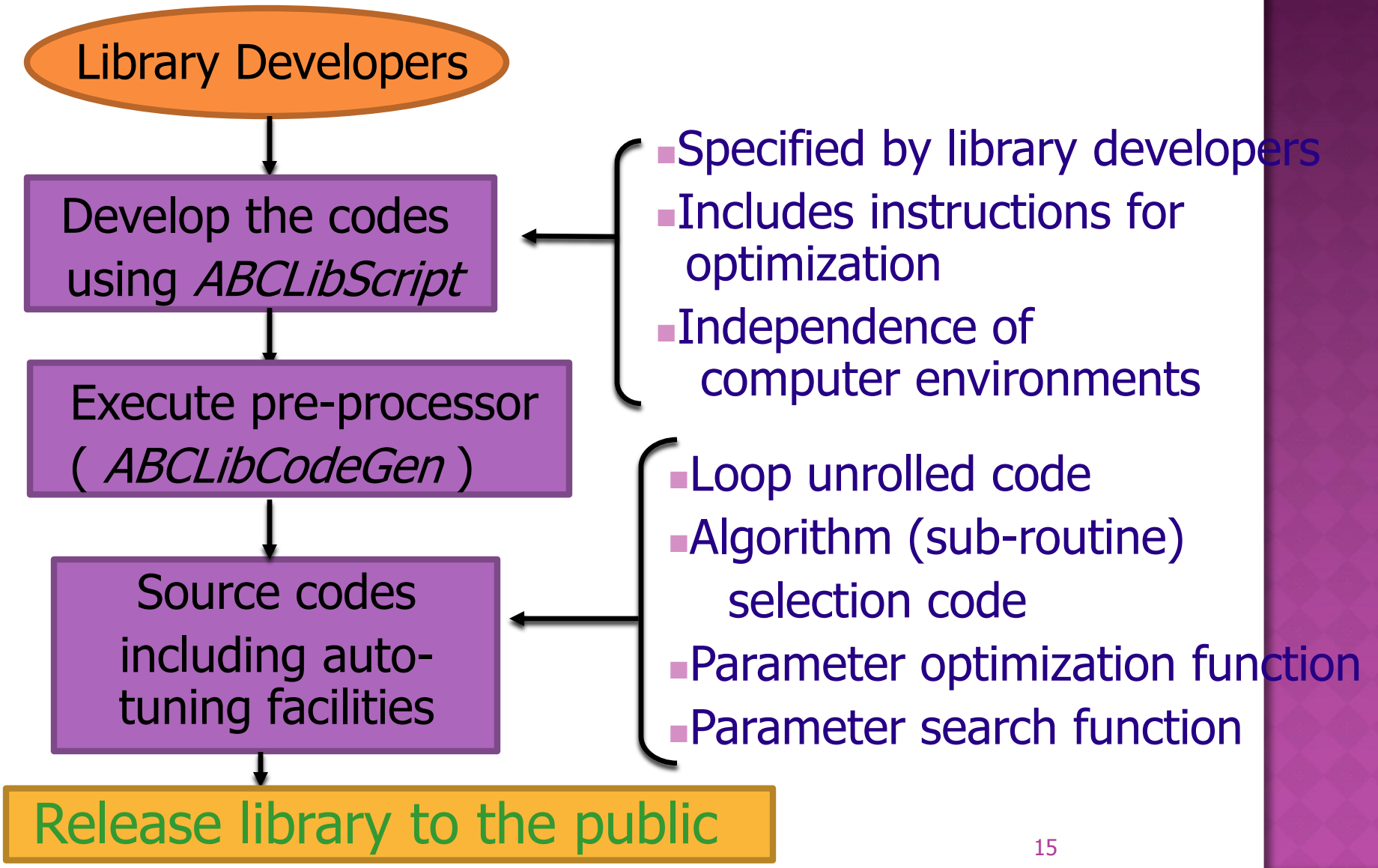
FIBER:

AN AUTO-TUNING FRAMEWORK

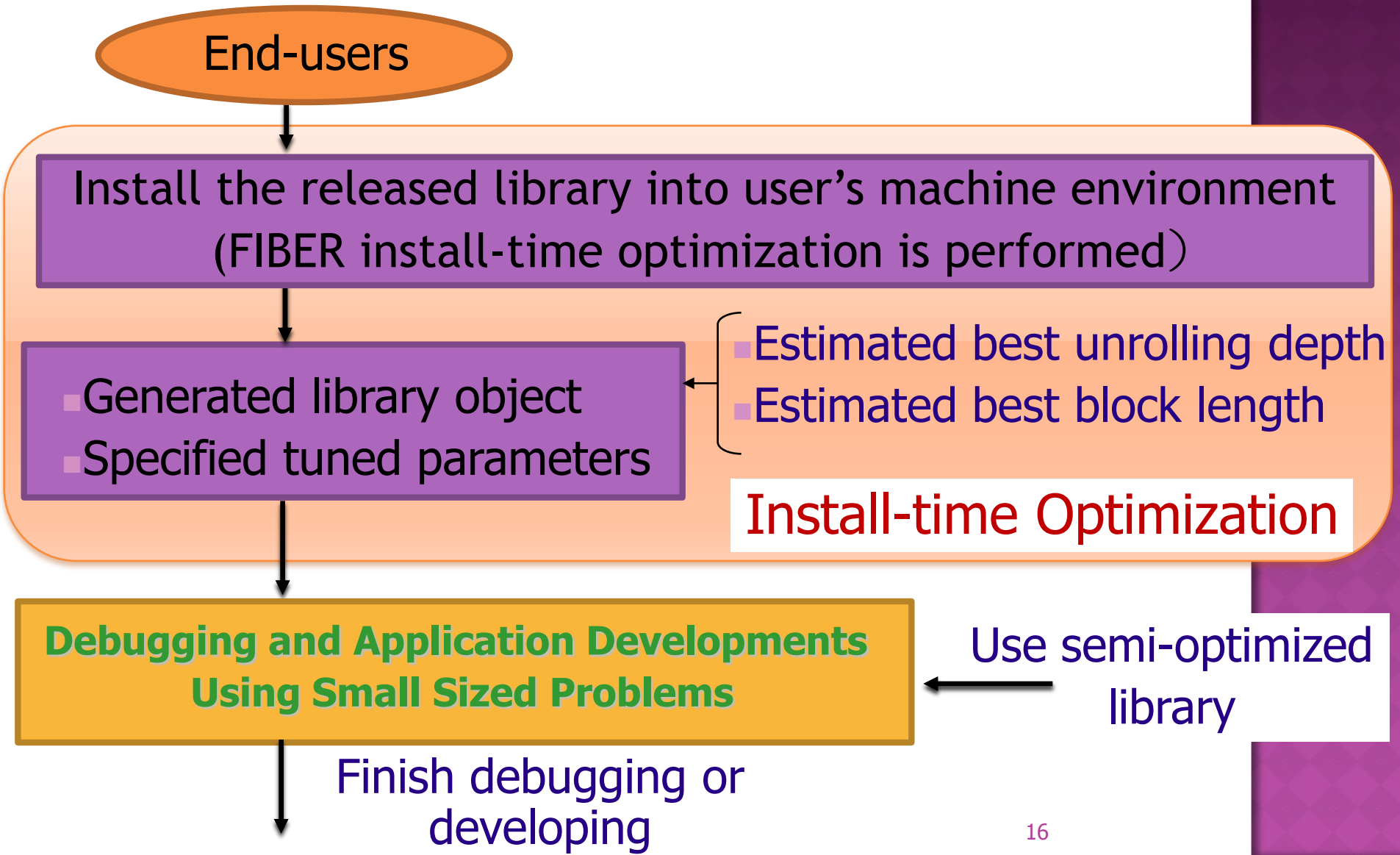
OVERVIEW OF FIBER

- FIBER (Framework for Install-time, Before Execute-time and Run-time auto-tuning) Paradigm
  - FIBER paradigm is a methodology for auto-tuning software to generalize application and obtain high accuracy for estimated parameters.
- How Auto-tuning is performed:
  - (a) Parameters that affect performance are extracted
  - (b) The parameters are automatically optimized
- (a) Parameter extraction:
  - by users utilizing a dedicated language (***ABCLibScript***)
- (b) Parameter optimization:
  - three kinds of optimization layers
  - using statistical methods

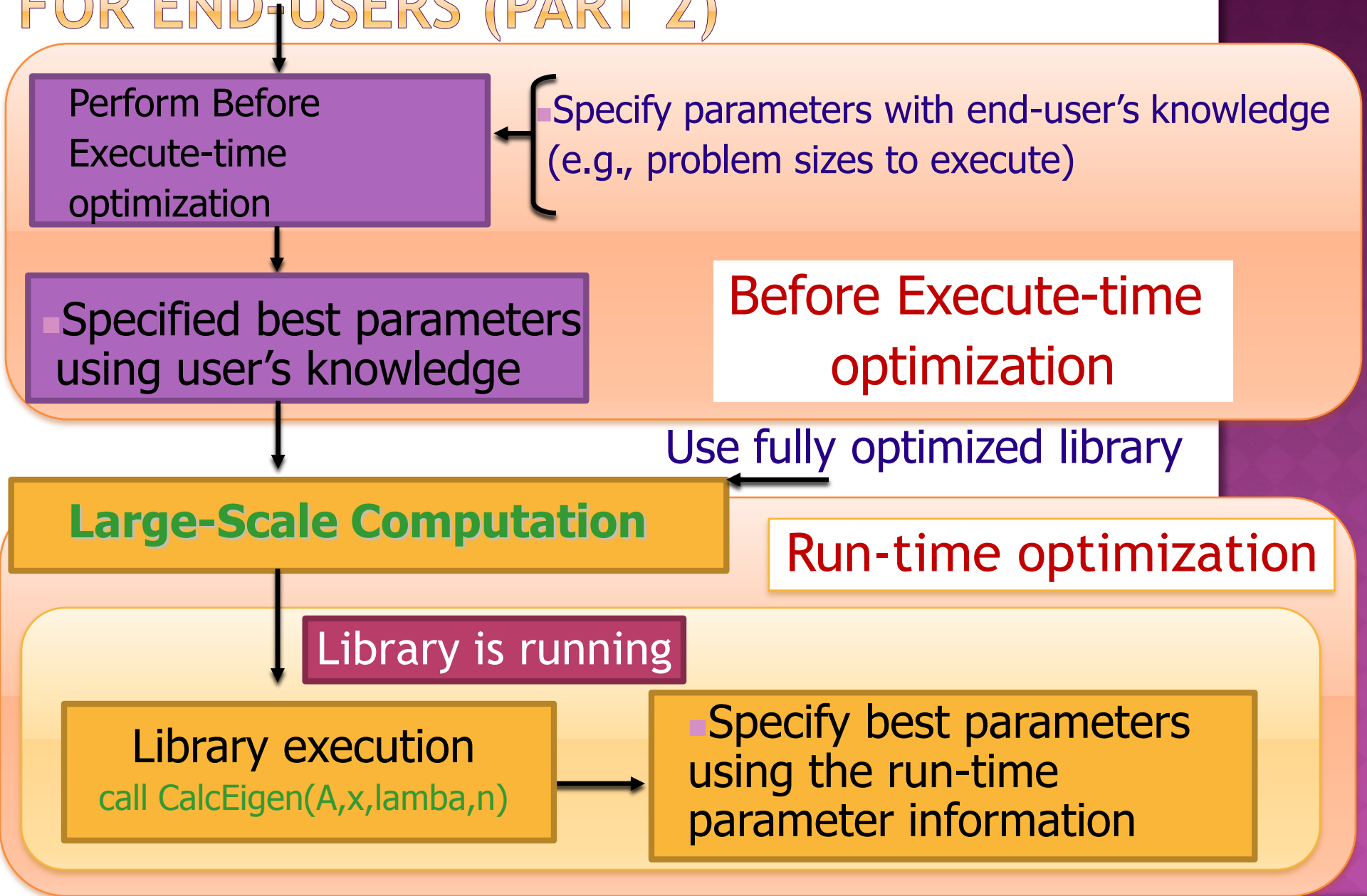
A SCENARIO OF FIBER FOR LIBRARY DEVELOPERS

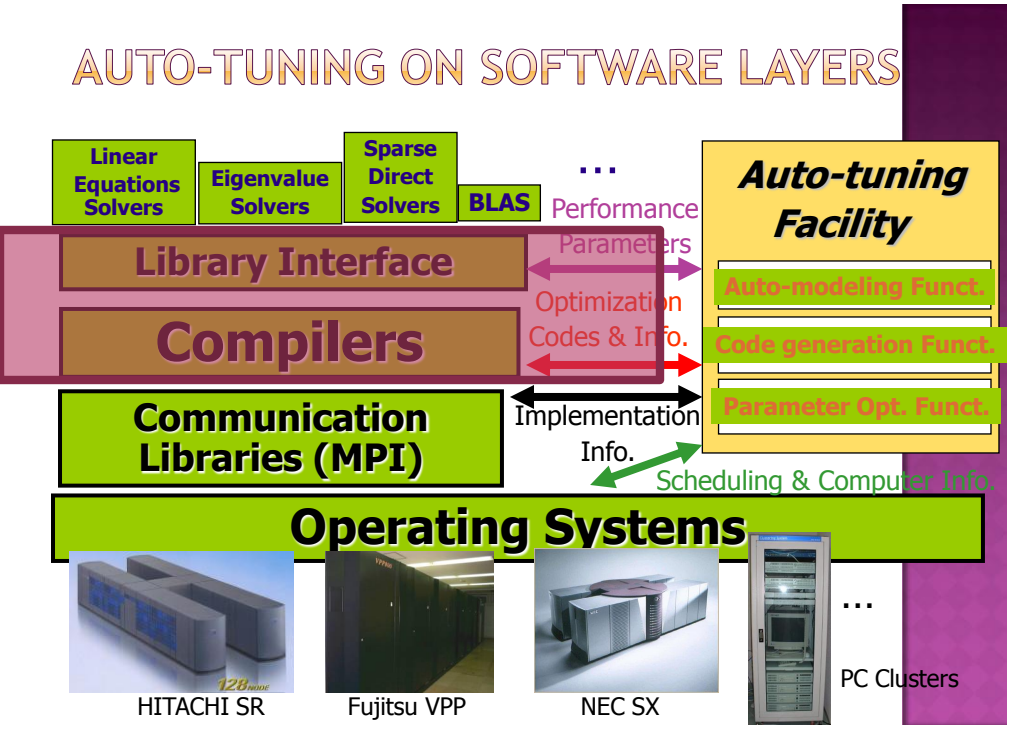


A SCENARIO OF FIBER FOR END-USERS (PART 1)



A SCENARIO OF FIBER FOR END-USERS (PART 2)





ABCLibScript:

AN AUTO-TUNING LANGUAGE

DIRECTIVE FOR LIBRARY DEVELOPERS : LOOP UNROLLING OPERATOR

- Unrolling Depth：Developer specifies using directive
  - Ex.：Matrix-matrix multiplication code

```
!ABCLib$ install unroll (i) region start
!ABCLib$ name MyMatMul
!ABCLib$ varied (i) from 1 to 8
!ABCLib$ debug (pp)
    do i=1, N
      do j=1, N
       da1 =  A(i, j)
       do k=1, N
         dc = C(k, j)
         da1 = da1 + B(i, k) * dc
       enddo
       A(i, j) = da1
      enddo
    enddo
!ABCLib$ install unroll (i) region end
```

Install-time optimization;
Unrolling process;

Unrolling Depth

Target Region
(Auto-tuning Region)

Directive for Library Developers : Loop Unrolling Operator (continued)

- After invocating pre-processor, the outer *i* loop is unrolled.

```
if (i_unroll .eq. 1)  then
        Original Code
   endif
if (i_unroll .eq. 2) then      /* i is dividable by 2 */
  im = N/2
     i = 1
     do ii=1, im
       do j=1, N
         da1 =  A(i,  j);  da2 =  A(i+1,j)
         do k=1, N
           dc = C(k, j)
           da1 = da1 + B(i,   k) * dc;  da2 = da2 + B(i+1, k) * dc; enddo
         A(i,  j) = da1; A(i+1,j) = da2
       enddo
       i = i + 2;
     enddo
   endif
  …
```

After code generation, the depth of unrolling is automatically parameterized.

DIRECTIVE FOR LIBRARY DEVELOPERS : ALGORITHM SELECTION OPERATOR

- Selecting algorithms as follows:

```
!ABCLib$  static  select region start
!ABCLib$  parameter (in CacheS, in NB, in NPrc)
!ABCLib$      select sub region start
!ABCLib$      according estimated
!ABCLib$          (2.0d0*CacheS*NB)/(3.0d0*NPrc)
                Target1 (Algorithm1)
!ABC-LIB$     select sub region end
!ABC-Lib$     select sub region start
!ABC-Lib$     according estimated
!ABC-Lib$         (4.0d0*ChcheS*dlog(NB))/(2.0d0*
                Target2 (Algorithm2)
!ABC-LIB$     select sub re
!ABC-LIB$ static select  re
```

Install-time Optimization;
Selection Operation;

Input Variables Used in
Cost Definition Funct.

Selection Base on
The Cost
Definition Function

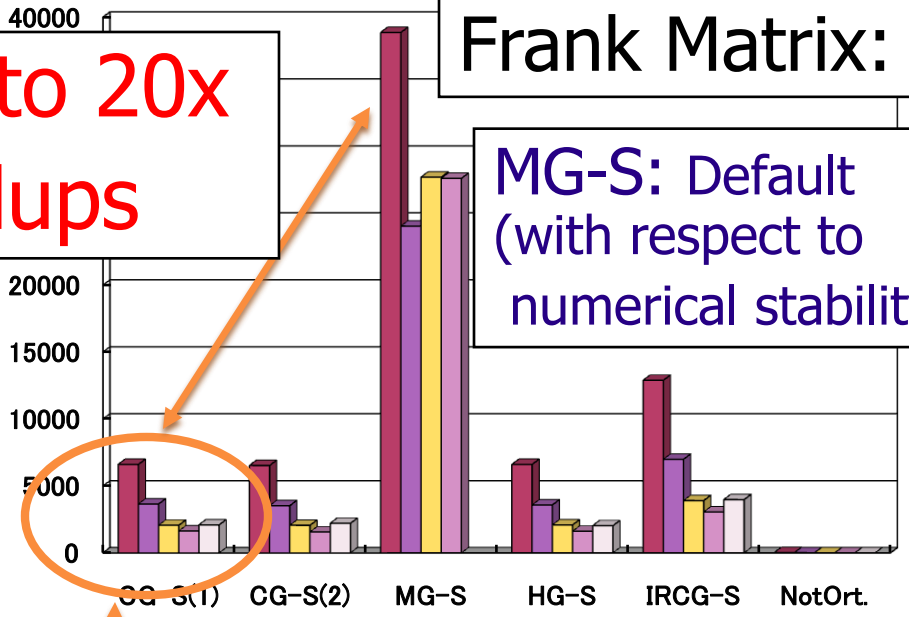
Target Region1
(Tuning Region1)

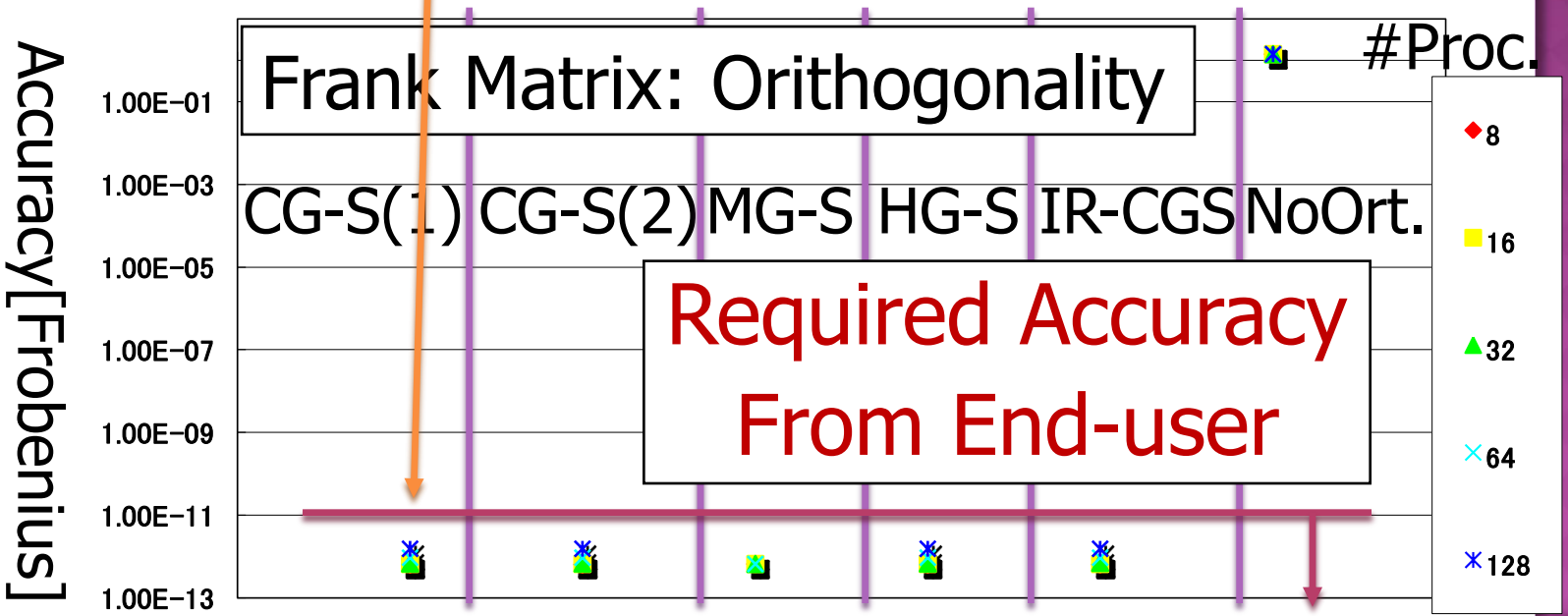
Target Region2
(Tuning Region2)

Selection information for Target 1 and 2 is parameterized.

AN EXAMPLE OF ALGORITHM SELECTION
(ORITHOGONALIZATION ON EIGENSOLVER, HITACHI SR8000/MPP)

From 7x to 20x Speedups



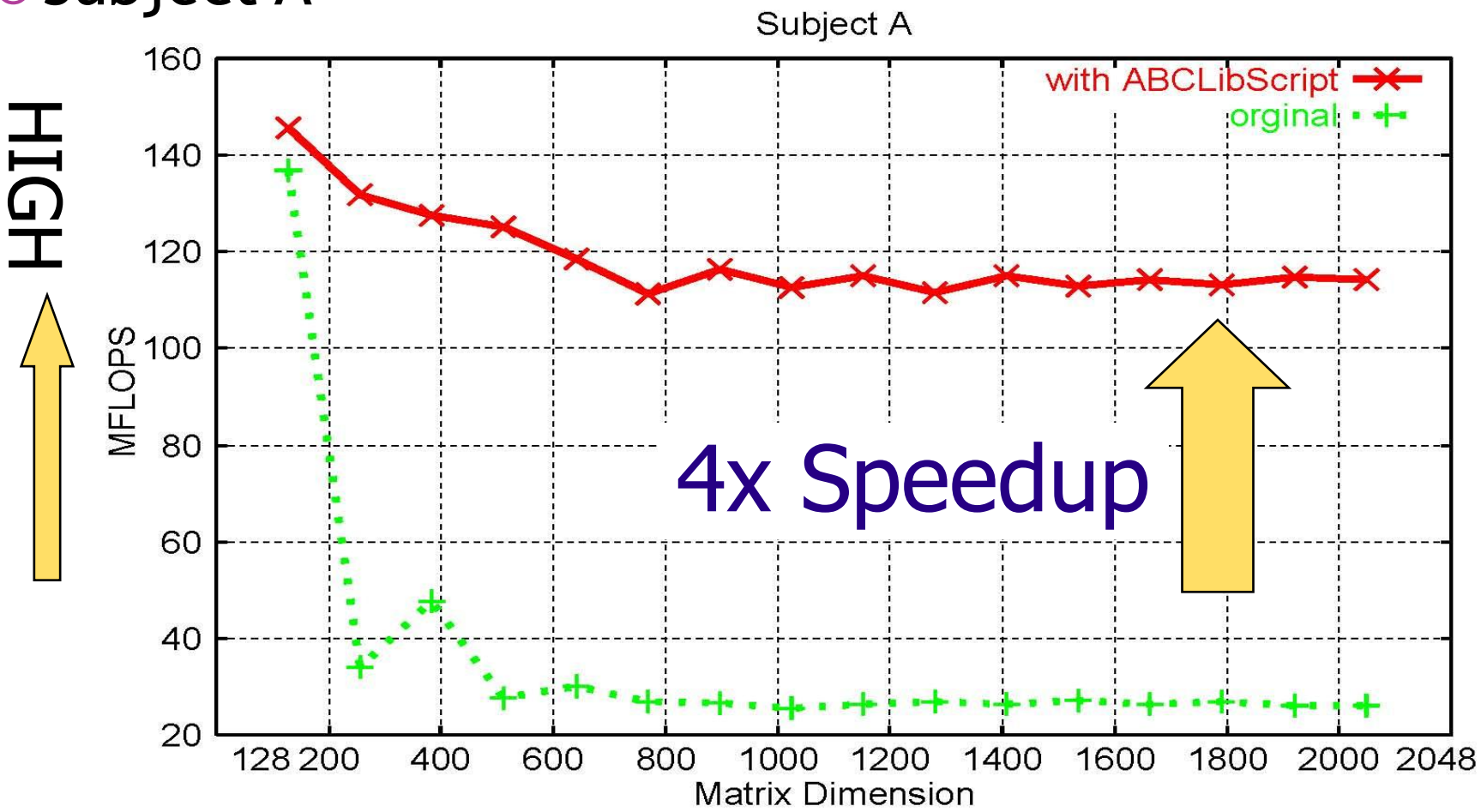


EXPERIMENT FOR EFFECT ON ABCLIBSCRIPT

- Target Application
  - Matrix-Matrix Multiplication
- ABCLibScript Directive
  - Unroll operator only
- Computer Environment
  - Intel Pentium4 (2.0GHz), PGI compiler
- Subjects
  - Subject A : Non-expert
  - Subject B : Semi-expert (He knows block algorithm.)
- Experiment term
  - 2 weeks for hand tuning
  - 2 hours for ABCLibScript programming

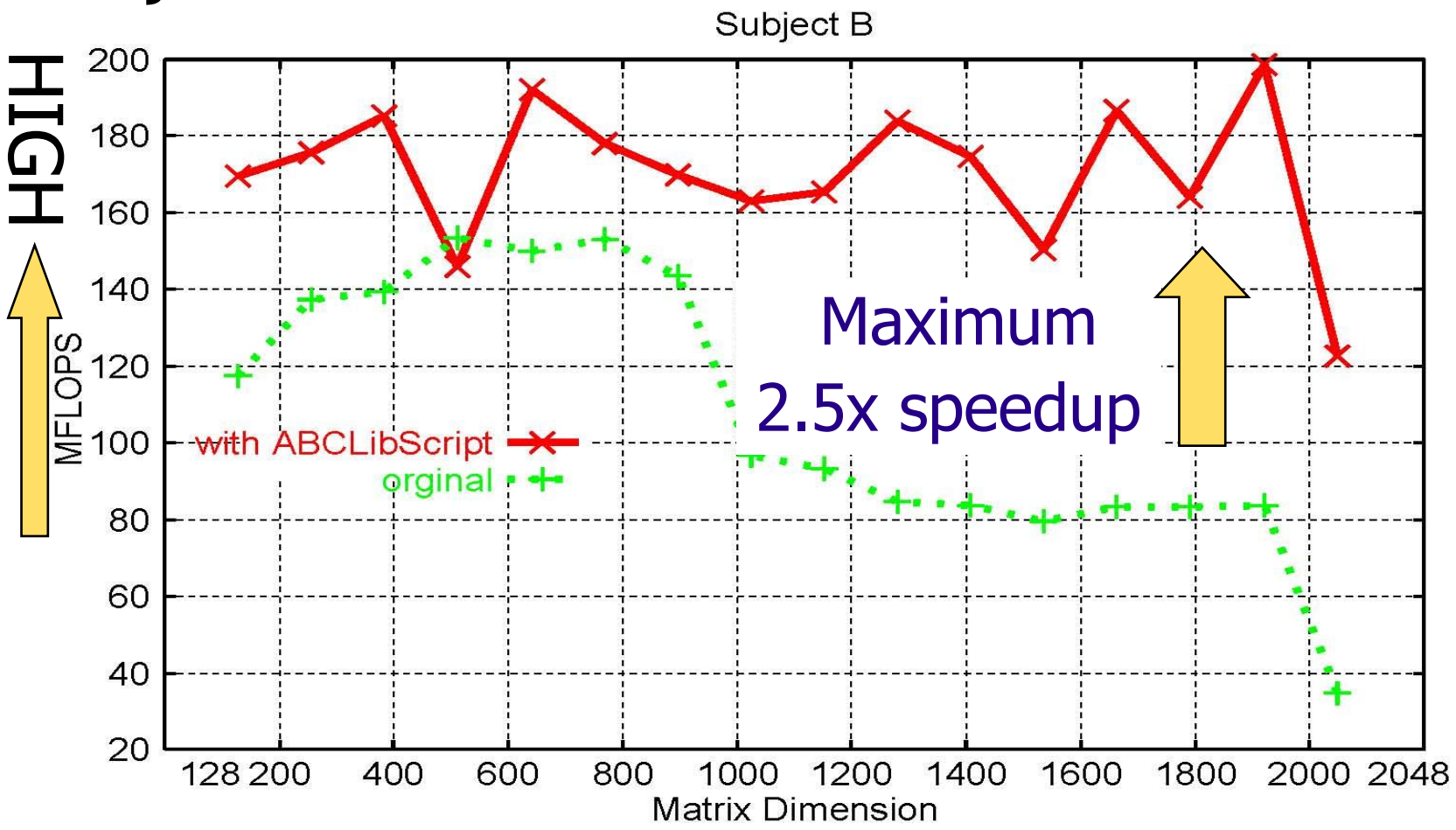
EXPERIMENT RESULT (1/2)

- Subject A



EXPERIMENT RESULT (2/2)

- Subject B

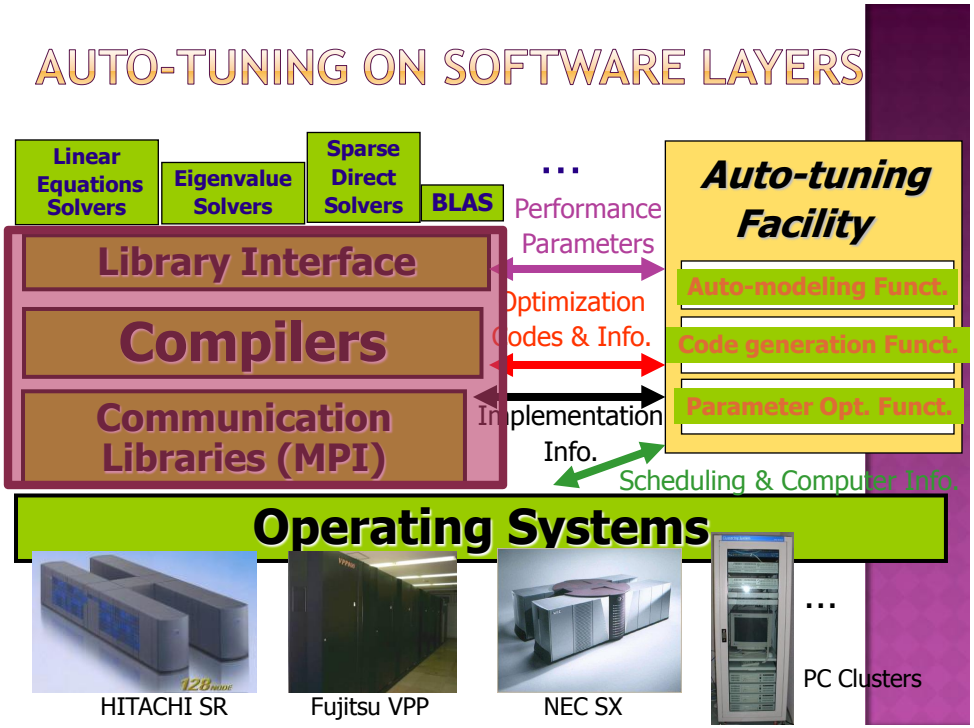


Effect on ABCLibScript (Summary)

- The performance was increased on between non-expert and semi-expert developers.
- The development term was reduced **from 2 weeks to 2 hours** with keeping better performance.

ABCLib:

A LIBRARY WITH AUTO-TUNING FACILITY



ABCLib_DRESSED:

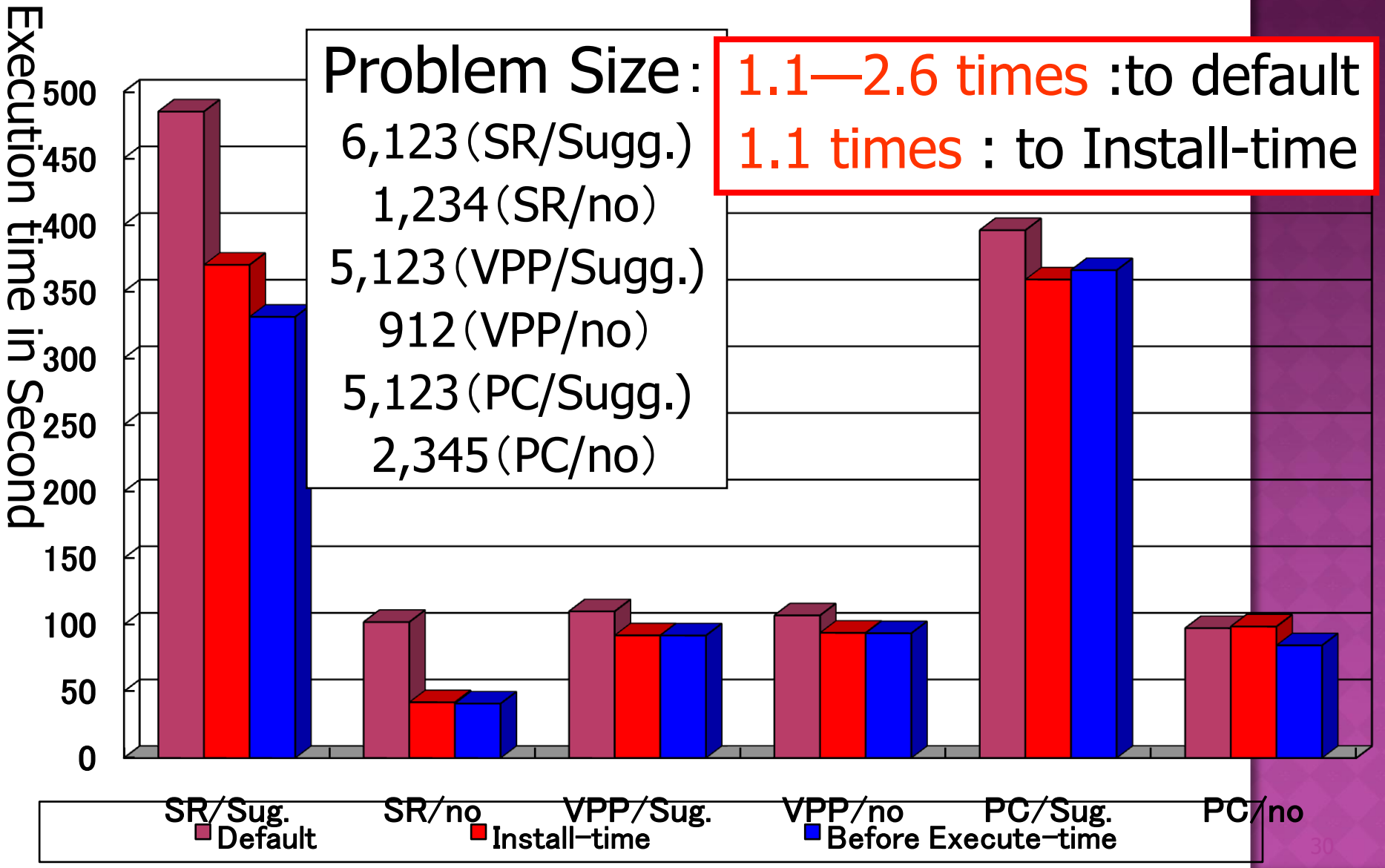
AN EIGENSOLVER WITH AUTO-TUNING FACILITY

ABCLib: An Auto-Tuning Library with FIBER Framework

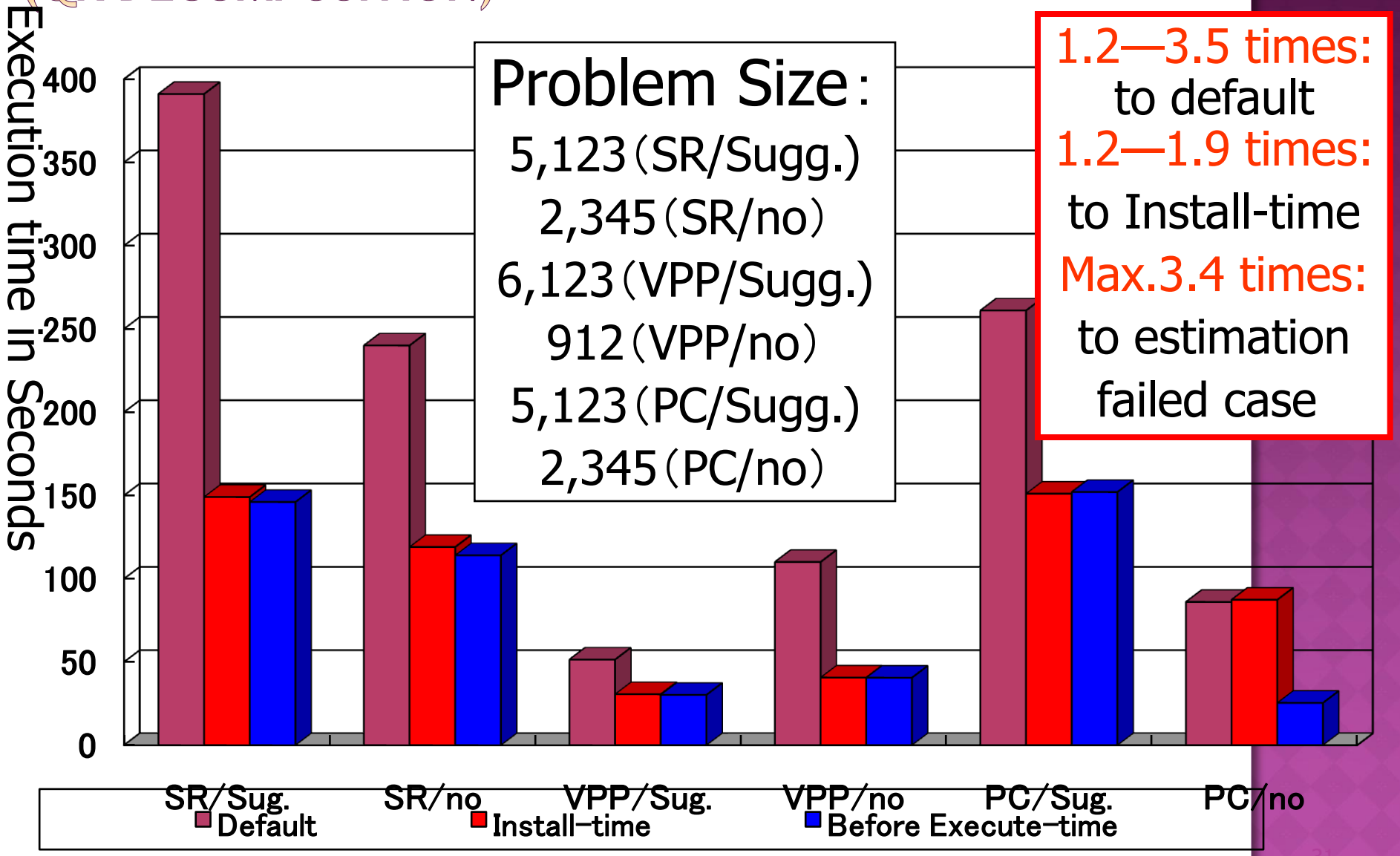
- Automatically Blocking-and-Communication adjustment LIBrary
- Timing for auto-tuning: Install-time
- Kernels for auto-tuning: about 30,000 lines.

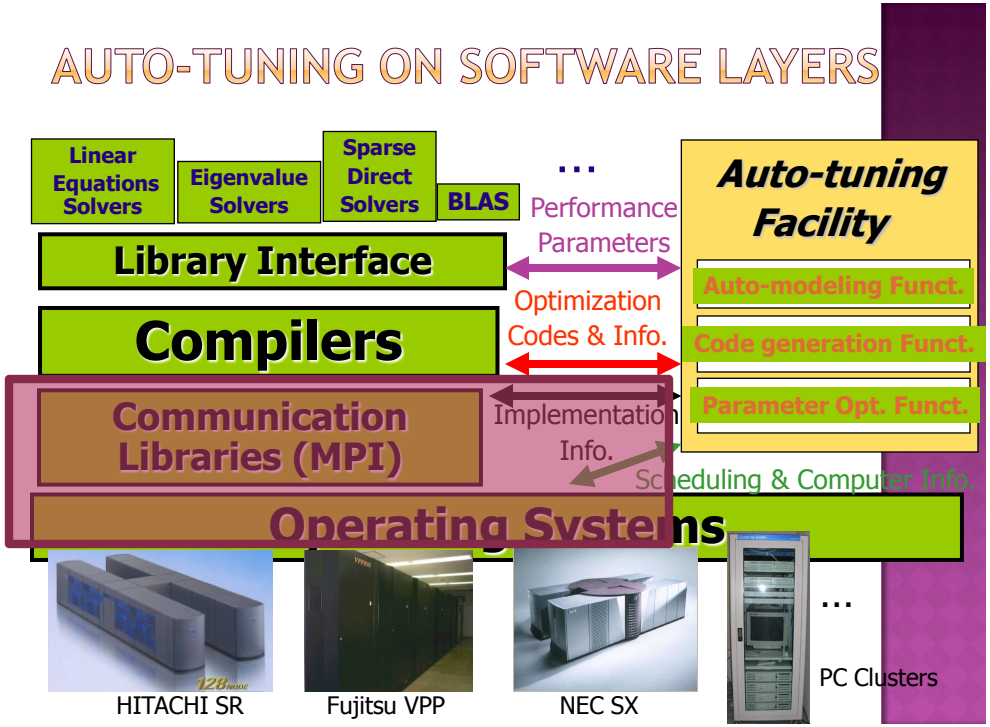
1. **Eigensolver (Real, Symmetric, Dense matrix)**
   - **Householder Tridiagonalization (Tri)**
     1. BLAS2 Unrolling Depth: Matrix-vector product ; 8 kinds;
     2. BLAS2 Unrolling Depth: Matrix updating process; 8 kinds;
     3. Communication Implementations: (One-to-one, Collective)
   - **Householder Inverse Transformation (Inv)**
     1. BLAS2 Unrolling Depth: Matrix updating process; 8 kinds;
     2. Communication Implementations:
        (Blocking one-to-one, Non-blocking one-to-one, Collective)
2. **QR Decomposition (Gram-Schmidt)**
   1. BLAS3 Unrolling Depths:
      Matrix updating process; 4(outer) * 8(second) = 32 kinds * 2 parts;
   2. Block Length for Algorithm: From 1 to 8;
   3. Communication Frequency (According to the block length)

INSTALL-TIME OPTIMIZATION VS. BEFORE EXECUTE-TIME OPTIMIZATION (EIGENSOLVER)



Install-time optimization vs. before execute-time optimization (QR decomposition)





MS-MPI Auto-tuning project:

A MPI LIBRARY WITH RUN-TIME AUTO-TUNING

MS-MPI RUN-TIME AUTO-TUNING

Assumption:

1. PC crusted with the Windows CCS 2003
2. Using MPI
   - Windows CCS 2003 provides MS-MPI

Problem:

- Nodes to be allocated are determined by scheduling policy on the Windows CCS 2003.
  - The physical topology for the allocated node affects communication performance.
- Communication pattern depends on the distribution of zero elements for input matrices.

-> It is impossible to find the best communication implementation before the running!

Challenge on MPI run-time implementation selection

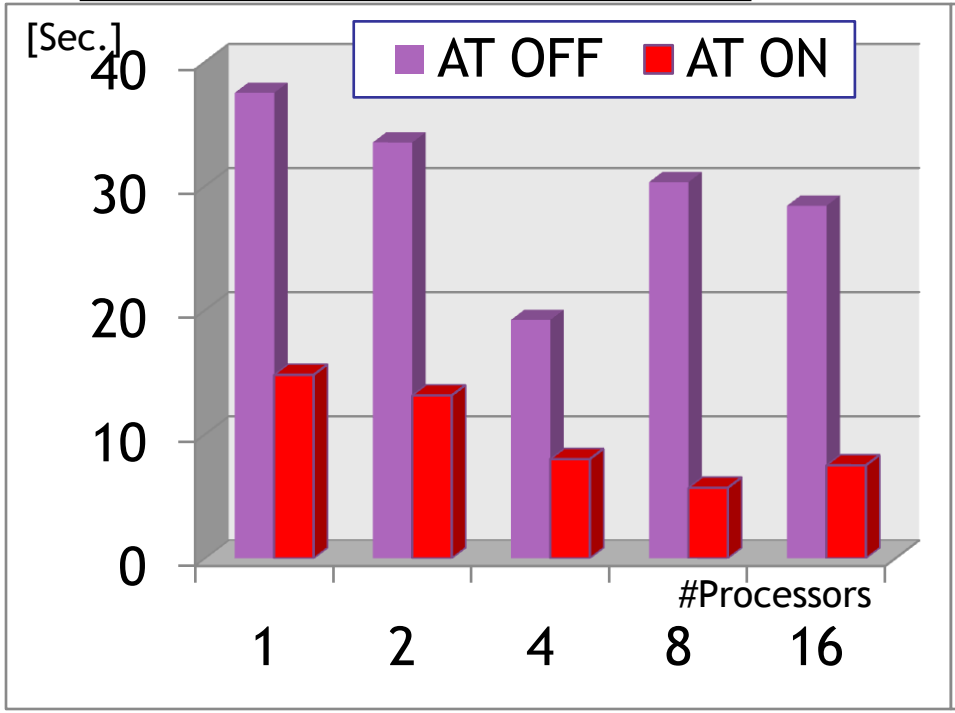
- Logging for past calls is performed at run-time.
  - Main target: Sparse iterative solver.
    - Same MPI function is called many times.
- Communication implementation selection is performed at run-time.
  1. **Ring sending vs. Binary three sending**
  2. **Synchronous vs. Asynchronous**
  3. **Overlapping vs. Non-overlapping**
  4. **Recursive halving vs. Normal**
- Final goal: Implementing a MPI lapper
  - **No modification of codes for end-user.**

AN PRELIMINARY EXPERIMENT ON A WINDOWS CLUSTER

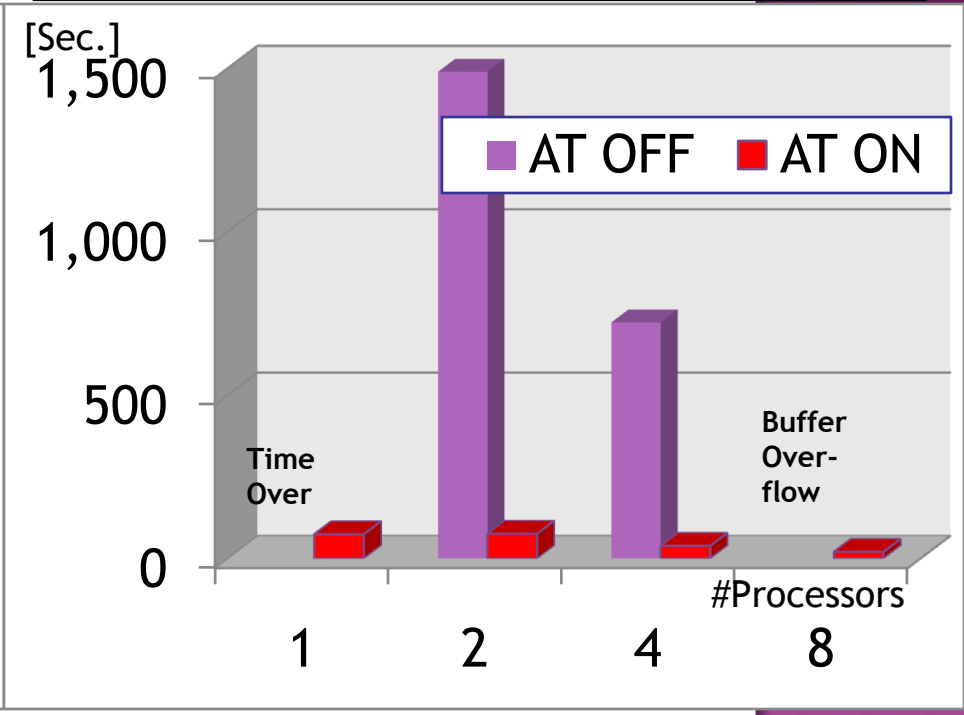
- Target Application
  - Parallel Sparse Iterative solver (GMRES Method)
    - Developed by Dr. H.Kuroda (U. of Tokyo)
  - Following performance parameters are auto-tuned according to input matrix:
    1. **Selection of preconditioner (Scaling, Jacobi, …)**
    2. **Adjustment of loop unrolling depth for sparse matrix multiplication**
    3. **Selection of MPI implementations (Gather, Overlap, Collective matter, …)**
- Experimental environment
  - Microsoft Innovation Center (MIC) at Chou-fu
  - AMD Athelon 64 X2 Dual, Cell Processor 3800+ (2.01GHz,2GByte RAM)
  - Windows CCS, MS-MPI, Visual Studio2005 C++

PRELIMINARY RESULTS

The Toeplitz Matrix



5 Points Deference Matrix



Maximum 20x speedup

RELATED PROJECTS

- SaNS (Self-adapting Numerical Software) Project @ University of Tennessee at Knoxville
  - SaNS Agent：
    - Provide intelligent components for the behavior of data, algorithms, and systems
    - Adapt computational Grid
    - Provide data repository for performance data
  - Provide a simple scripting language
- BeBOP (Berkeley Benchmarking and Optimization Group) Project @ University of California at Berkeley
  - **OSKI : Optimized Sparse Kernel Interface**
    - A collection of low-level primitives that provide automatically tuned computational kernels on sparse matrices, for use by solver libraries and applications.
- SPIRAL Project @ Carnegie Mellon University
  - **Software/Hardware Generation for DSP algorithm**

CLOSING REMARKS

- To establish high productivity on numerical libraries, auto-tuning facility is needed.
  - ***FIBER*** is one of the promising frameworks for establishing high productivity.
  - ***ABCLibScript*** is the computer language to describe auto-tuning process based on FIBER for general applications.
- Next generation supercomputers must have..
  - complicated architectures (multicore,…)
  - more than 10,000 processors

  -> we need somehow intelligent and automated tuning systems.

ACKNOWLEDGEMENTS

- Auto-Tuning Research Group in JAPAN
  - Chair: Toshitsugu Yuba (U. of Electro-comm.)
  - Vice Chair: Takahiro Katagiri (U. of Tokyo)
  - Reiji Suda (U. of Tokyo)
  - Toshiyuki Imamura (U. of Electro-comm.)
  - Yusaku Yamamoto (Nagoya U.)
  - Ken Naono (HITACHI Ltd.)
  - Kentaro Shimizu (U. of Tokyo)
  - Hiroyuki Sato (U. of Tokyo)
  - Shoji Ito (RIKEN)
  - Takeshi Iwashita (Kyoto U.)
  - Kazuya Terauchi (Japan Visual Numerics Inc.)
  - Masashi Egi (HITACHI Ltd.)
  - Takao Sakurai (HITACHI Ltd.)
  - Hisayasu Kuroda (U. of Tokyo)

FOR MORE INFORMATION

- If you are interested in ABCLib project, please visit:
  - http://www.abc-lib.org/