Welcome

MS1 : Auto-tuning on Numerical Libraries and Advanced Computer Systems - Part I of II

Organizers:
Takahiro Katagiri (*U. Tokyo, Japan*)
Toshiyuki Imamura (*U. Electro-Communications, Japan*)

SIAM PP08
Wednesday, March 12, 10:00 AM - 12:00 AM
Room: Roswell

AIM OF THIS MINI SYMPOSIUM

- Current computer systems are getting more and more complex:
  - Many-core Processors
  - Deeply Hierarchy and Unsymmetrical Access Memory Systems
- These imply uncontrollable, especially, tuning process on these systems is tough and formidable work.
- In this organized session, we will show the state-of-the-art for ***the auto-tuning technology***, which is a crucial to solve the tuning problem for numerical library.

SPEAKERS OF MS1

- 10:00-10:25
  (L) Towards General Auto-tuning Description Language on Advanced Computing Systems
  - Takahiro Katagiri, University of Tokyo, Japan
- 10:30-10:55
  (S) Proposal of Run-time Parameter Auto-Tuning Approach for Restarted Lanczos Method
  - Takao Sakurai, Ken Naono, and Masashi Egi, Hitachi Ltd., Japan; Mitsuyoshi Igai and Hiroyuki Kidachi, Hitachi ULSI Systems Corporation, Japan
- 11:00-11:25
  (C) Automatic Tuning for Parallel FFTs
  - Daisuke Takahashi, University of Tsukuba, Japan
- 11:30-11:55
  (C) A Bayesian Approach to Automatic Performance Tuning
  - Reiji Suda, University of Tokyo, Japan

(C) Core Technique and Principle;

(L) Language;

(S) Algorithm Selection; For Automatic Tuning on Numerical Libraries

SPEAKERS OF MS25 (THURSDAUY, MARCH 13)

- 1:30-1:55
  (C) Automatic Tuning for the LOBPCG Eigenvalue Solver
  - Toshiyuki Imamura, University of Electro-Communications, Japan
- 2:00-2:25
  (S) Auto-tuning Effect of Iterative Method Library on Windows CCS
  - Hisayasu Kuroda and Takahiro Katagiri, University of Tokyo, Japan
- 2:30-2:55
  (C) Performance of the Complex Hessenberg QR Algorithm on the CSX600 Processor
  - Takafumi Miyata, Yusaku Yamamoto, and Shao-Liang Zhang, Nagoya University, Japan; Yoshimasa Nakamura,
- 3:00-3:25
  (S) Automatic Selection of Parameters in Parallel Preconditioners for Ill conditioned Problems
  - Kengo Nakajima, University of Tokyo, Japan

(C) Core Technique and Principle;

(L) Language;

(S) Algorithm Selection; For Automatic Tuning On Numerical Libraries

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)

Towards General Auto-tuning Description Language on Advanced Computing Systems

Takahiro Katagiri (*U. Tokyo, Japan*)

SIAM PP08
Wednesday, March 12, 10:00 AM - 10:25 AM
Room: Roswell

OUTLINES

- Motivation
- Our Solutions
  - FIBER : An Auto-tuning Framework
  - ABCLibScript: An Auto-tuning Description Language
- Ongoing Projects
  - MS-MPI Run-time Auto-tuning
  - Embedded System Application Auto-tuning
- Related Projects
- Conclusion Remarks

To establish high productivity on numerical software

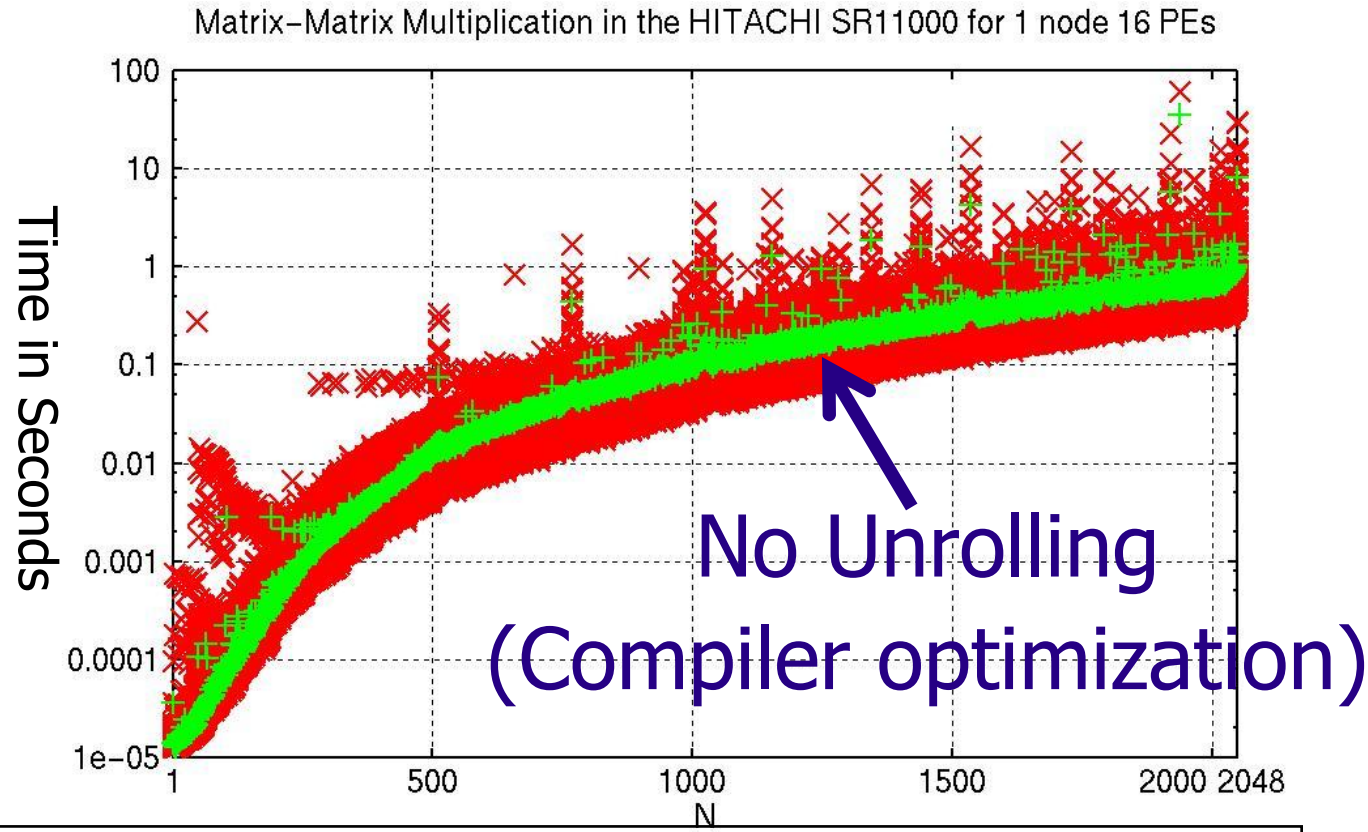
MOTIVATION

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, 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.

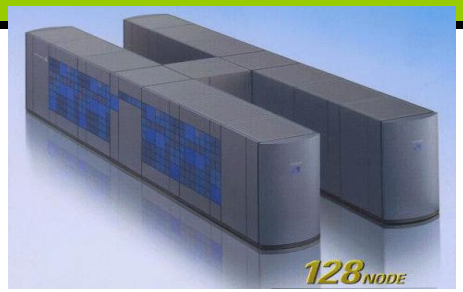
Auto-tuning Facility

Auto-modeling Funct.

Code generation Funct.

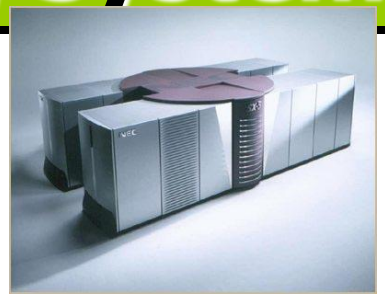
Parameter Opt. Funct.

Operating Systems



HITACHI SR

Fujitsu VPP

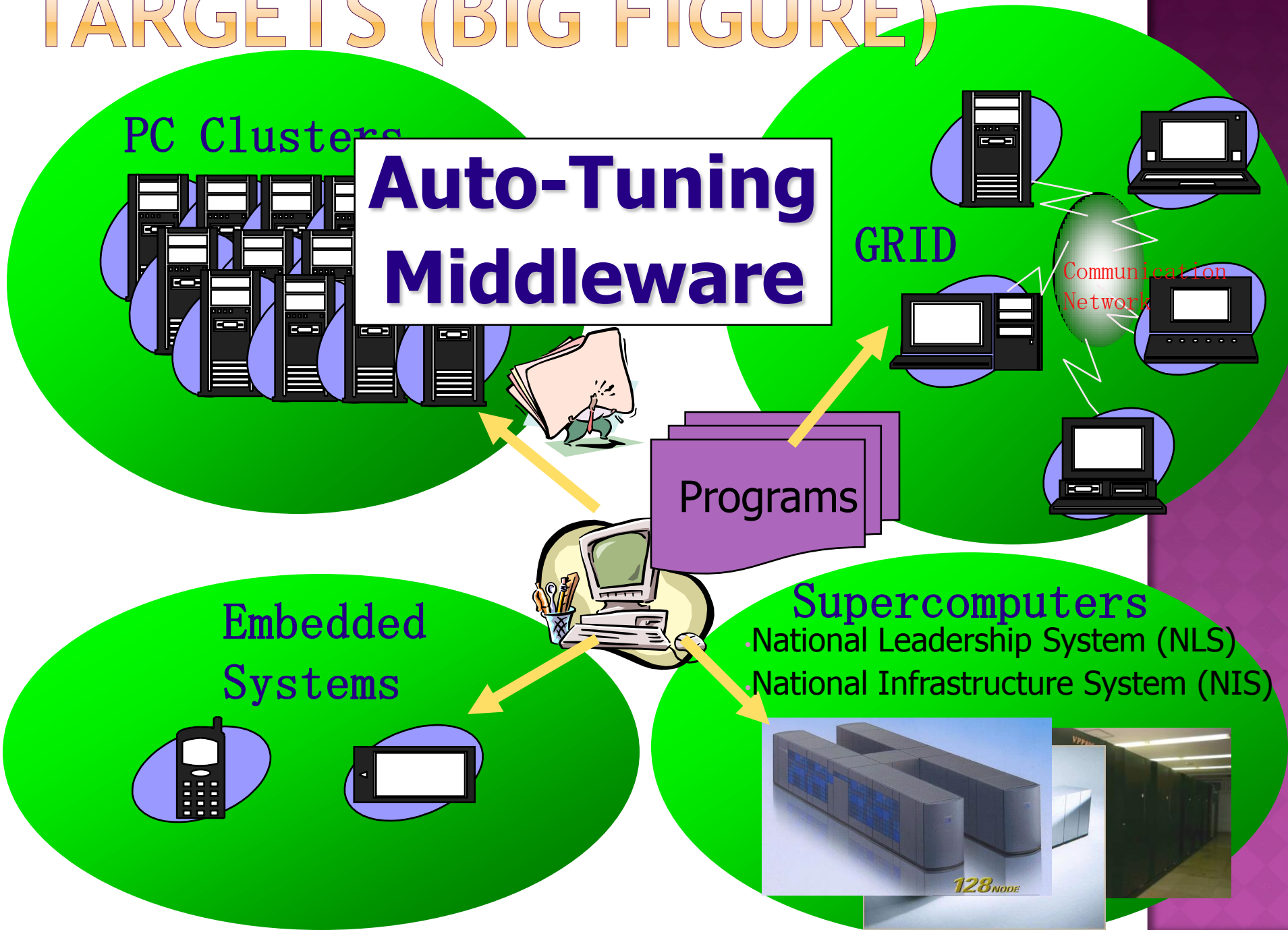


NEC SX

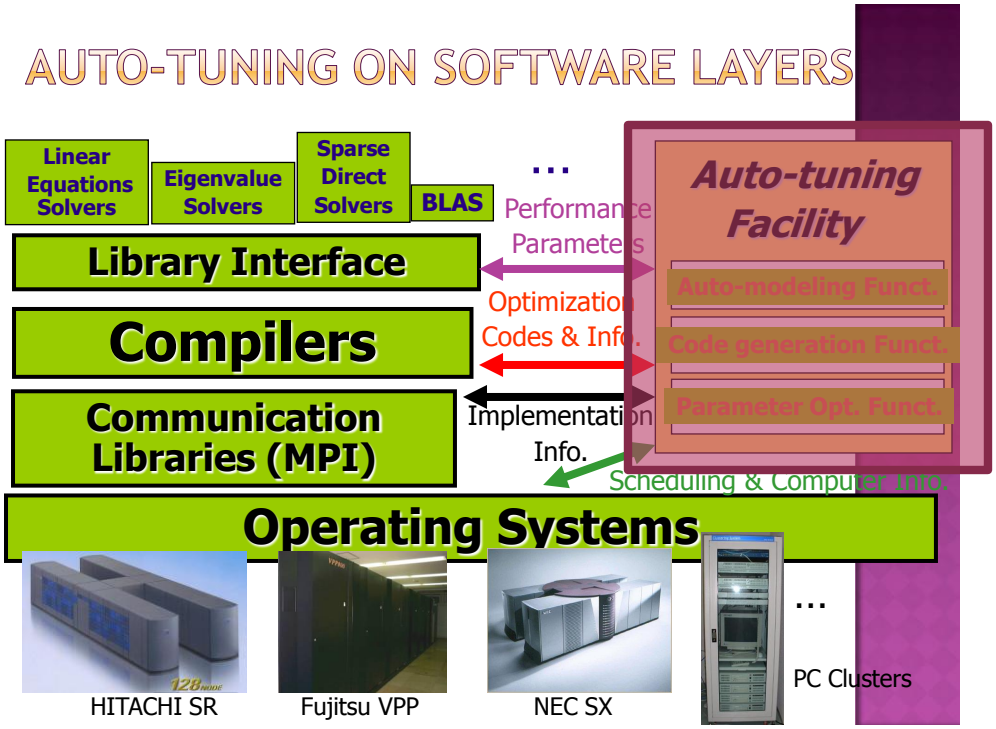
…

PC Clusters

TARGETS (BIG FIGURE)



AUTO-TUNING ON SOFTWARE LAYERS



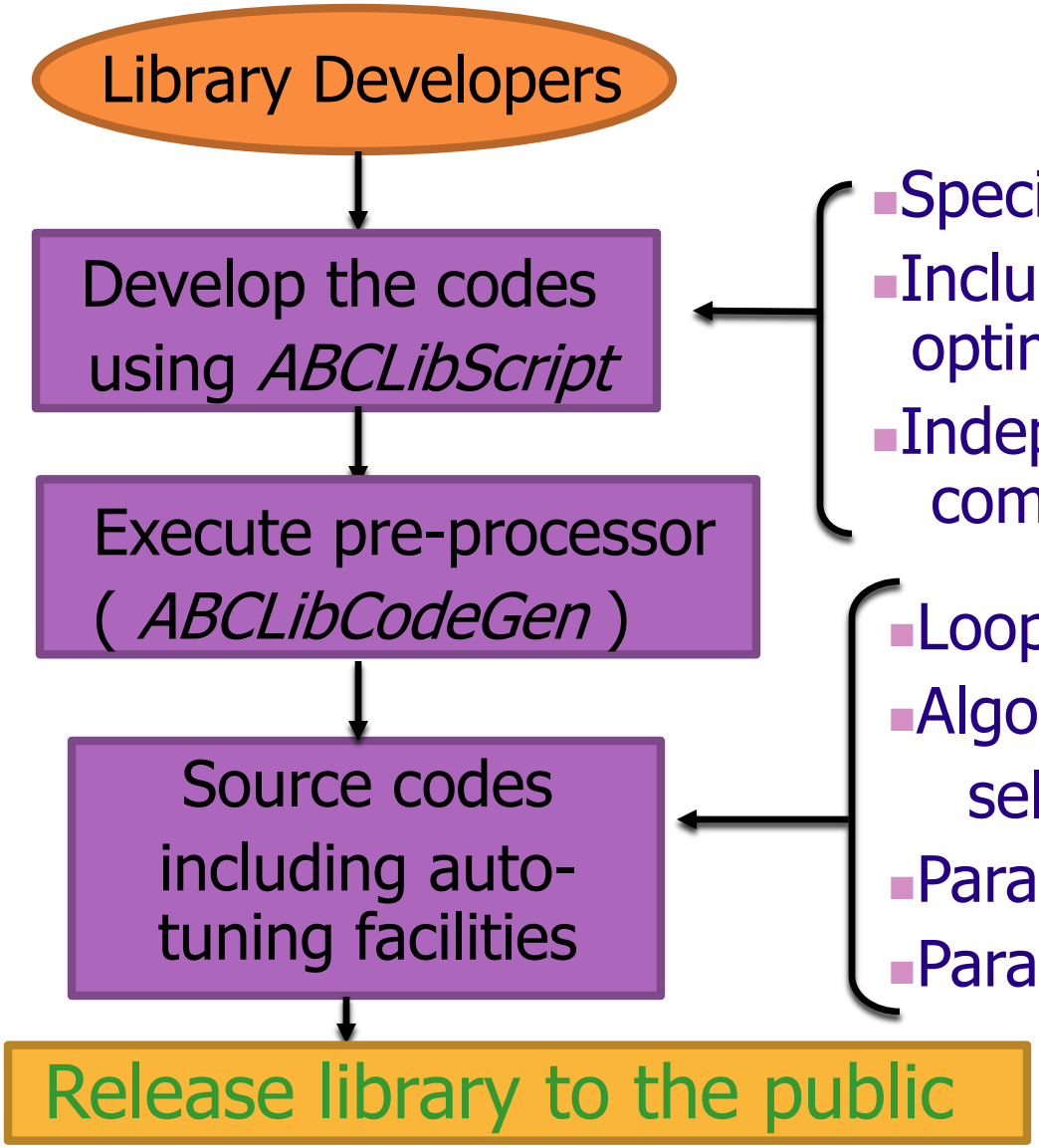
FIBER:

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

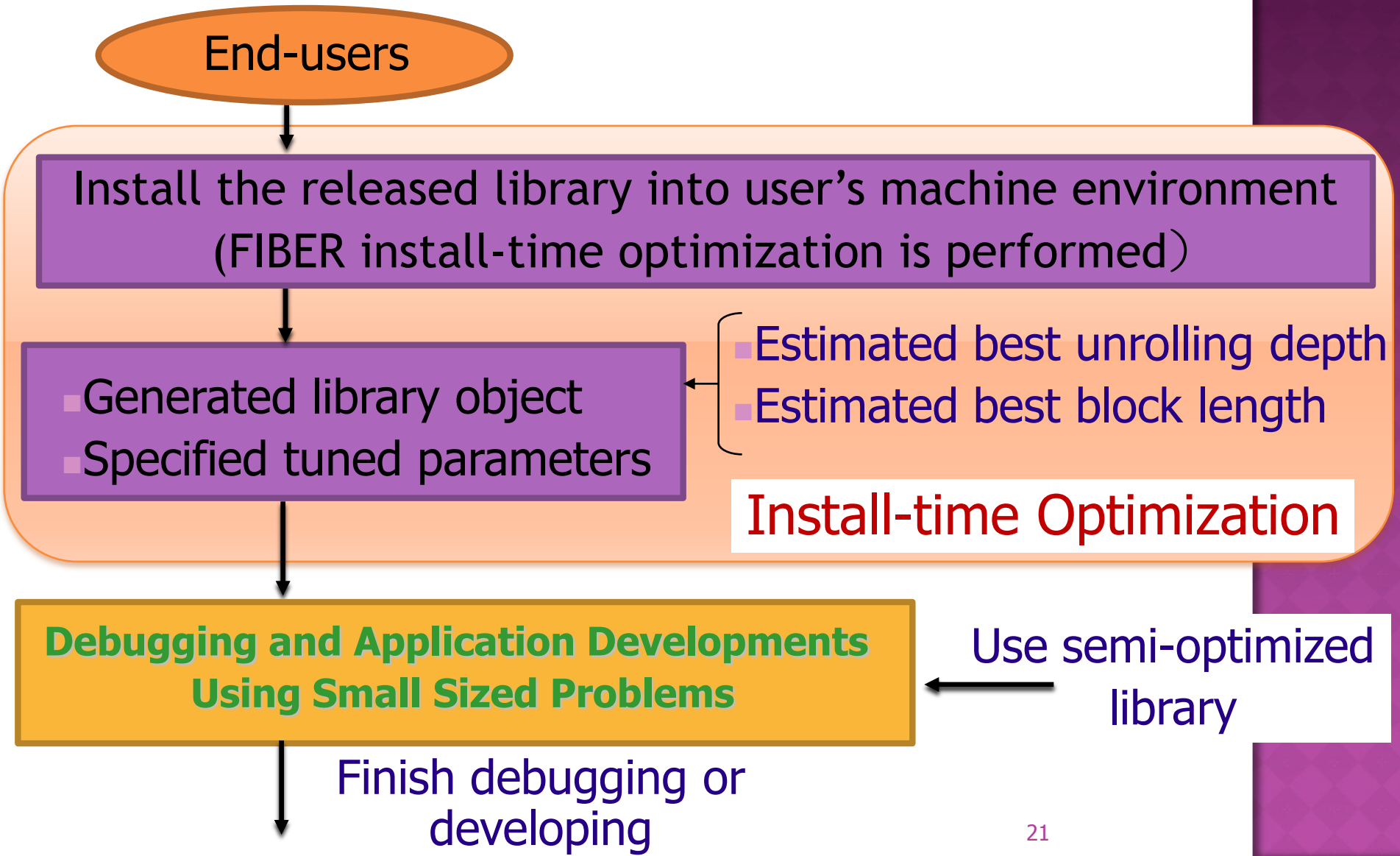
A SCENARIO OF FIBER FOR LIBRARY DEVELOPERS



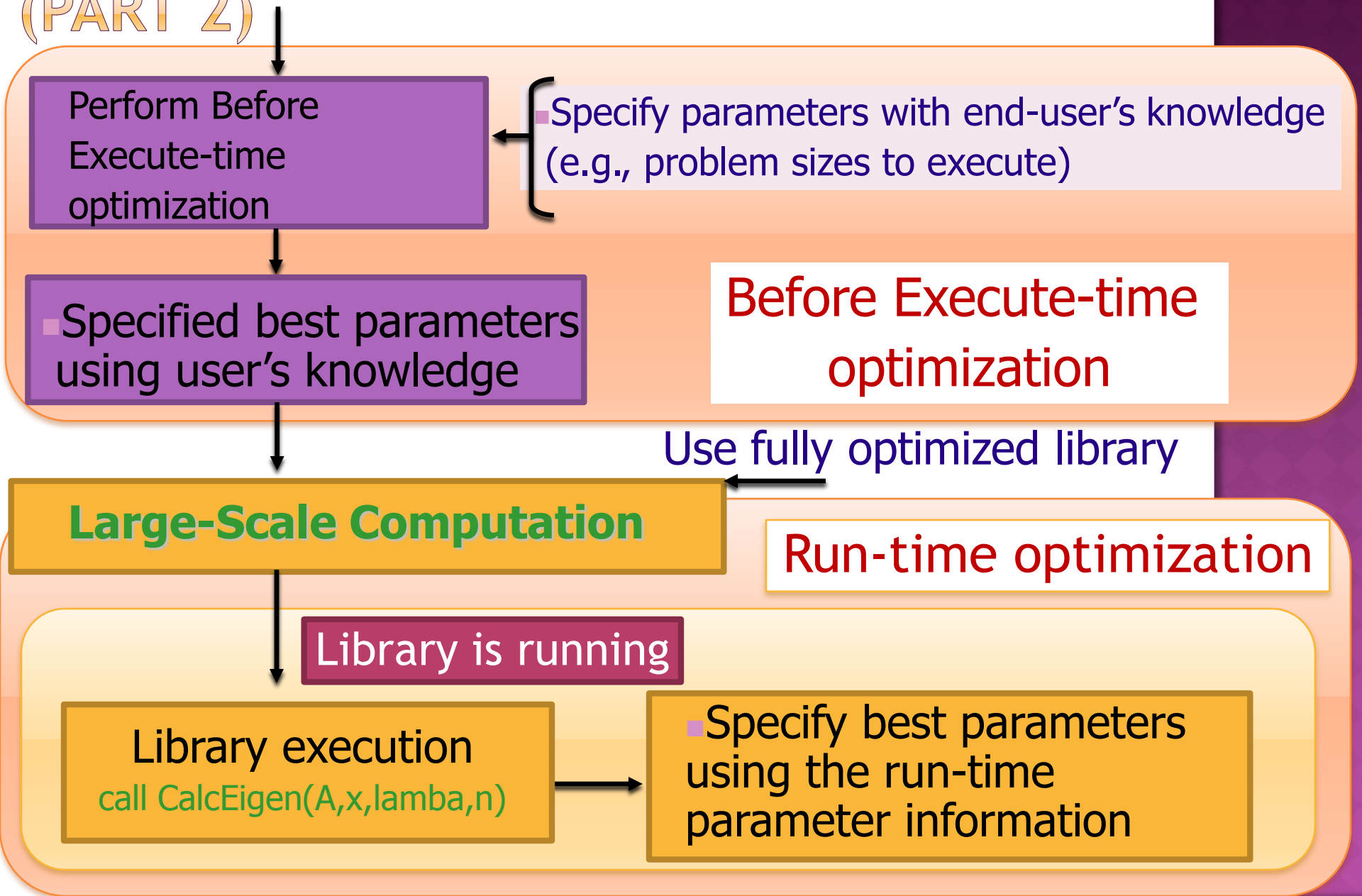
- Specified by library developers
- Includes instructions for optimization
- Independence of computer environments

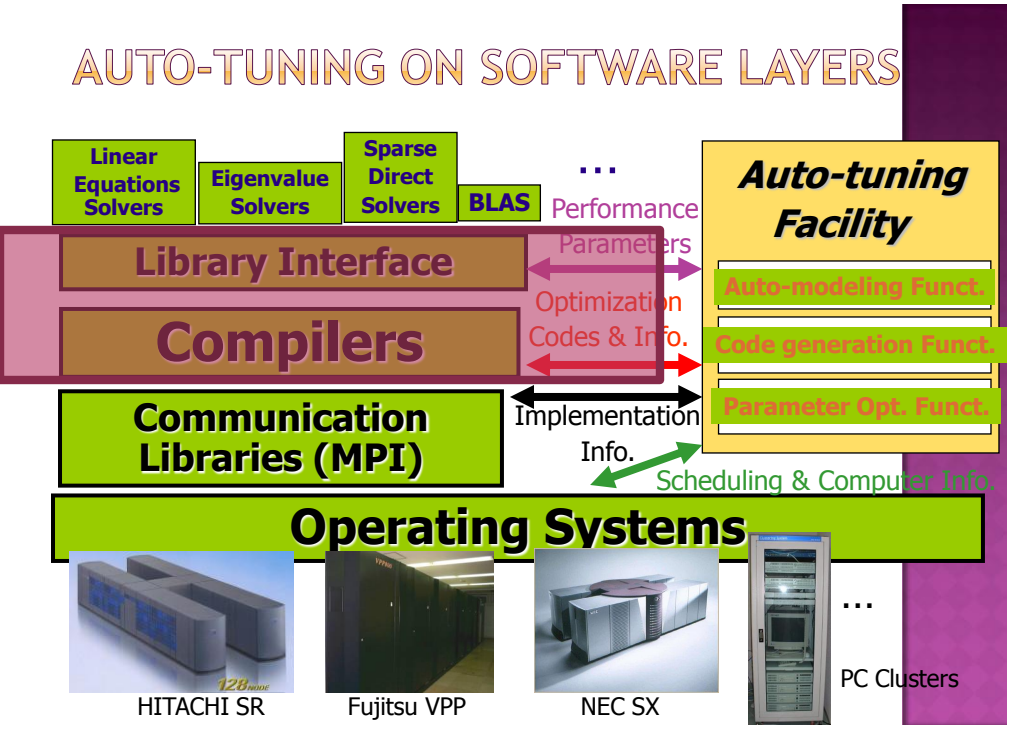
- Loop unrolled code
- Algorithm (sub-routine) selection code
- Parameter optimization function
- Parameter search function

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;

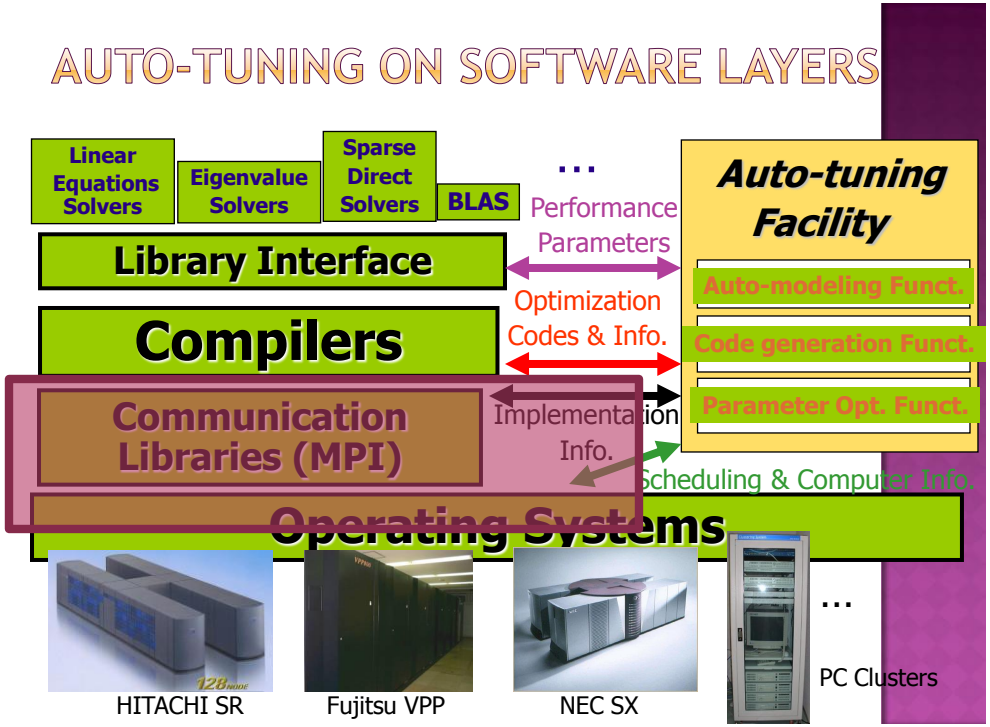
Input Variables Used in **Cost Definition Funct.**

Selection Base on The Cost Definition Function

Target Region1（Tuning Region1）

Target Region2（Tuning Region2）

Selection information for Target 1 and 2 is parameterized.



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 for jobs are determined by the scheduling policy on the Windows CCS 2003.
  - The physical topology for the allocated nodes 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

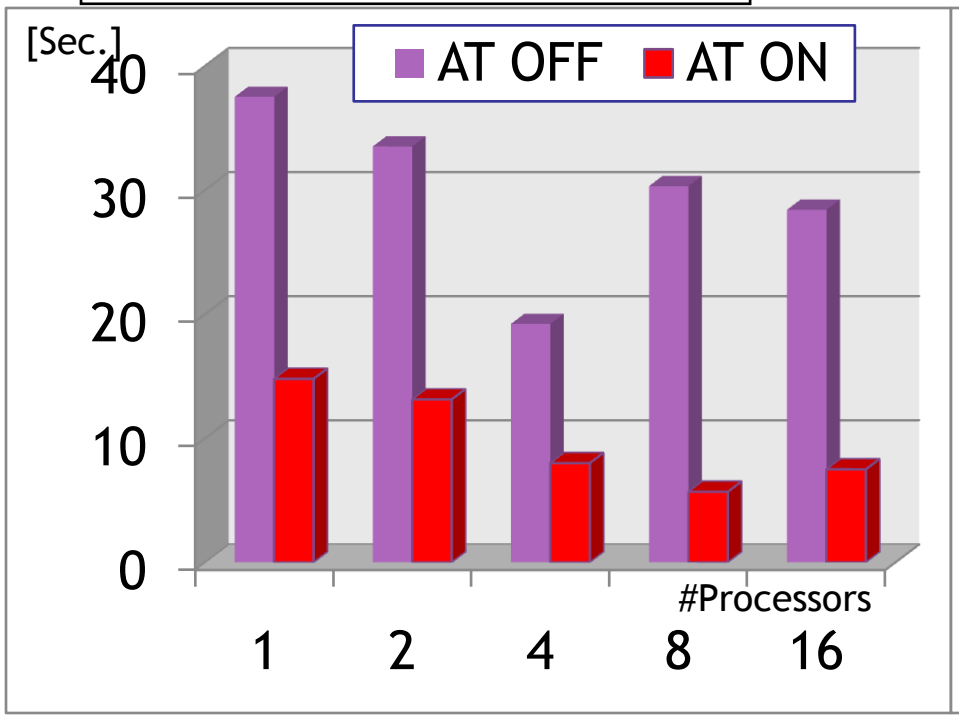
- Logging for past calls is performed at run-time.
  - Main target: Sparse iterative solver.
    - Same MPI function is called many times.
- Communication implementation selection is performed at run-time.
  1. **Ring sending vs. Binary three sending**
  2. **Synchronous vs. Asynchronous**
  3. **Overlapping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

AN PRELIMINARY EXPERIMENT ON A WINDOWS CLUSTER

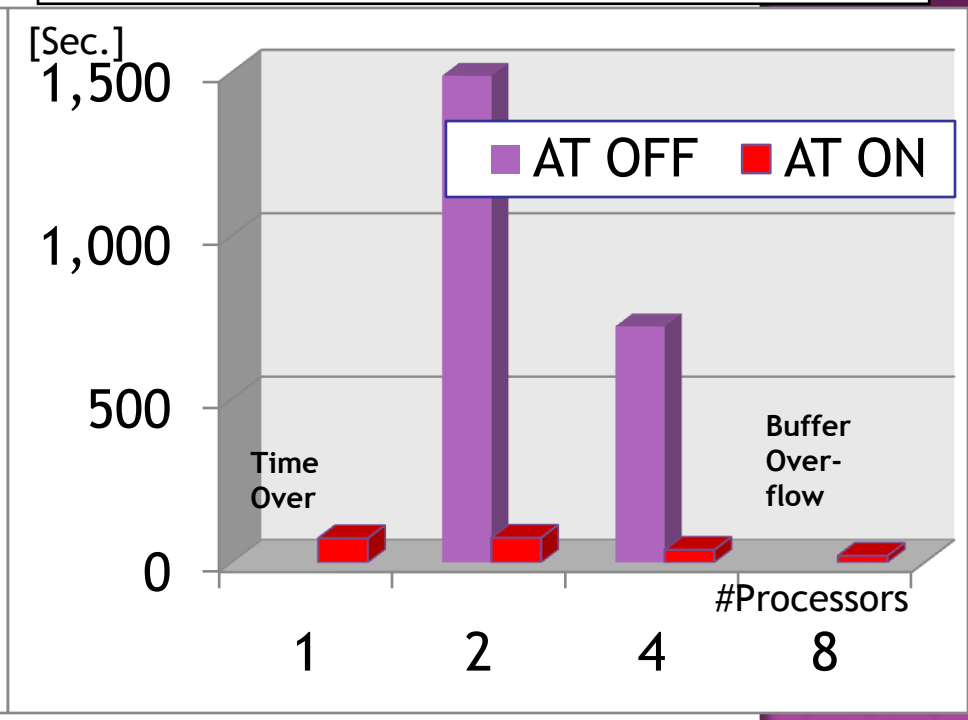
- Target Application
  - A Parallel Sparse Iterative solver using GMRES(m).
    - Developed by Dr. H.Kuroda (U. of Tokyo)
  - Followings are auto-tuned according to input matrix:
    1. **Selection of preconditioner (Scaling, Jacobi, …)**
    2. **Adjustment of loop unrolling depth for sparse matrix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

PRELIMINARY RESULTS

The Toeplitz Matrix

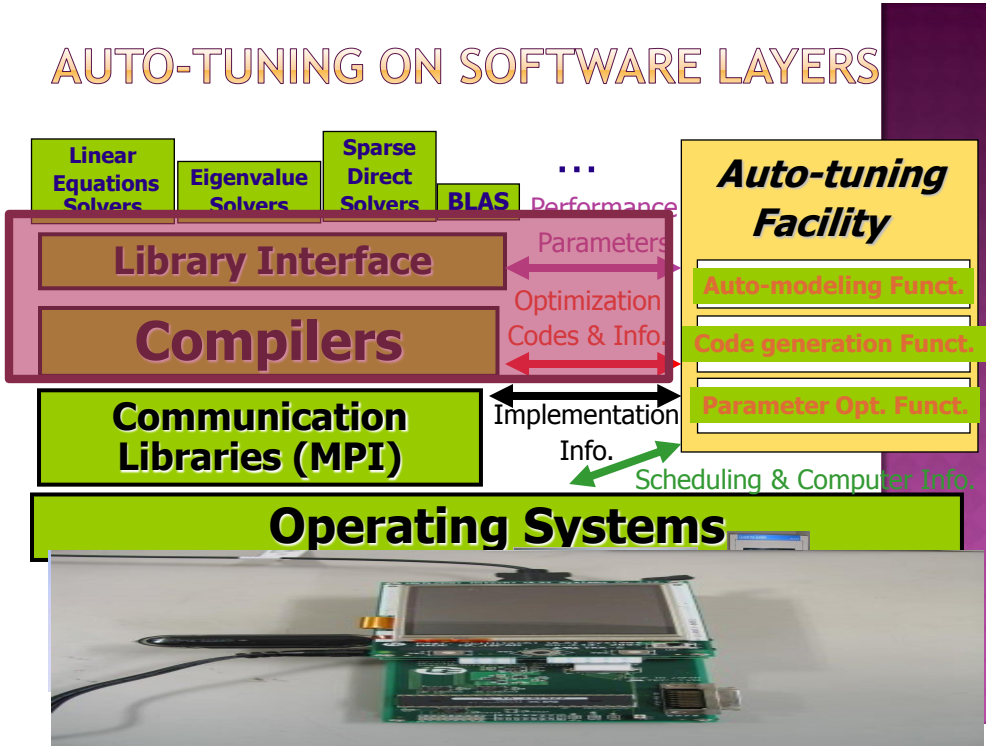


5 Points Deference Matrix



Maximum 20x speedup

(For details: Please attend MS25 tomorrow)



Embedded System project:

EMBEDDED SYSTEM APPLICATION SOFTWARE AUTO-TUNING

T-Engine Development Kit

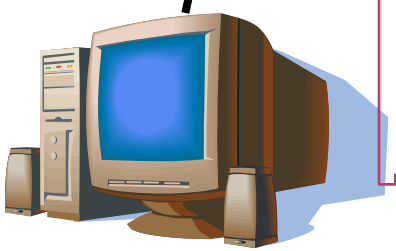
TARGET EMBEDDED SYSTEM FOR DEVELOPERS

TARGET EMBEDDED SYSTEM

T-Engine/SH775R Development Kit

Serial port

Transport
Execution objects



2G Bytes Memory
USB disk

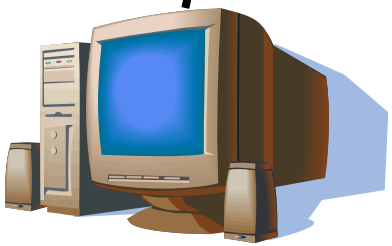
Program Development PC

- Cygwin
- GNU toolkit (gcc)

TARGET SOFTWARE WITH AUTO-TUNING

T-Engine/SH775R Development Kit

Serial Port



```
!ABCLib$ install unroll (i) region start
!ABCLib$ name MyMatMul
!ABCLib$ varied (i) from 1 to 8
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
```

Program Development PC

- Using ABCLibScript to develop their codes

TARGET SOFTWARE WITH AUTO-TUNING

T-Engine/SH775R Development Kit

Serial Port

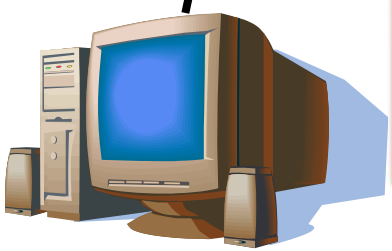
Implementation Candidates

#1

#2

#3

#4



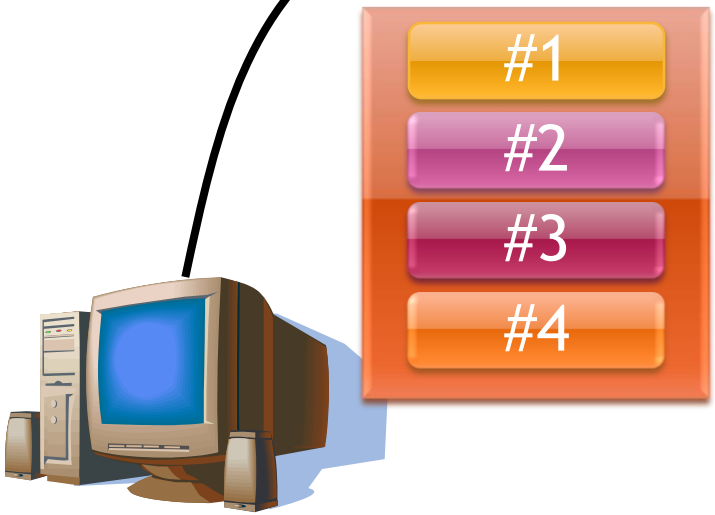
Program Development PC

- Invoke ABCLibScript Preprocessor
- Codes for Auto-tuning are automatically generated.

TARGET SOFTWARE WITH AUTO-TUNING

T-Engine/SH775R Development Kit

Serial Port



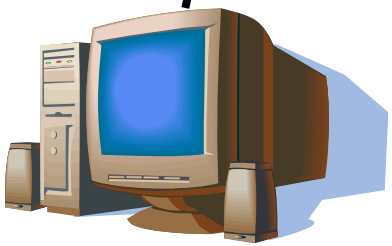
Program Development PC

- Compile
- Transport the object code to the development kit.

TARGET SOFTWARE WITH AUTO-TUNING

T-Engine/SH775R Development Kit

Serial Prot



Program Development PC

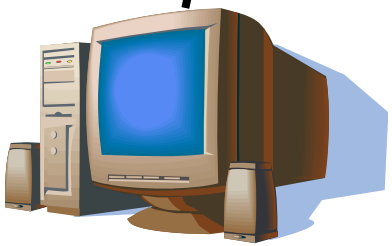
- Invoke auto-tuning mode
- Select the best implementation according to measured target speed.
- The USB disk is used for temporary work space for the system.

#1
#2
#3
#4

TARGET SOFTWARE WITH AUTO-TUNING

T-Engine/SH775R Development Kit

Serial Port



Program Development PC

Development-time Auto-tuning

Optimized:

#2

PERFORMANCE EVALUATION

TARGET HARDWARE DETAILS

T-engine Development Kit

CPU	Renesas Technology Corp. SH7751R(SH-4 Core,240MHz)
Flash Memory	8M Bytes
SDRAM	64M Bytes
Work Disk	USB Storage Device 2G Bytes
Execution Program	Process base on the CLI (Command Line Interpreter).

- Execution program was compiled on standard environment mode for the T-Engine/SH7751R development kit.
- Compiler: gcc version 3.0.4 Option: -O2

TARGET SOFTWARE

1. Matrix-Matrix Multiplication
   - N=128
   - Single precision
2. FFT on the MiBench:
   An Embedded System Benchmark
   - FFT/IFFT for Telecom
   - N=8192
   - Single precision

RESTRICTION FOR ABCLIBSCRIPT

- Current version of ABCLibScript: Only support for Fortran90 and MPI-1
- The target is written in C.
- We made the codes must be generated by ABCLibScript by hand coding.

SPECIFIER FOR ABCLIBSCRIPT ON MAT-MAT MULTIPLICATION

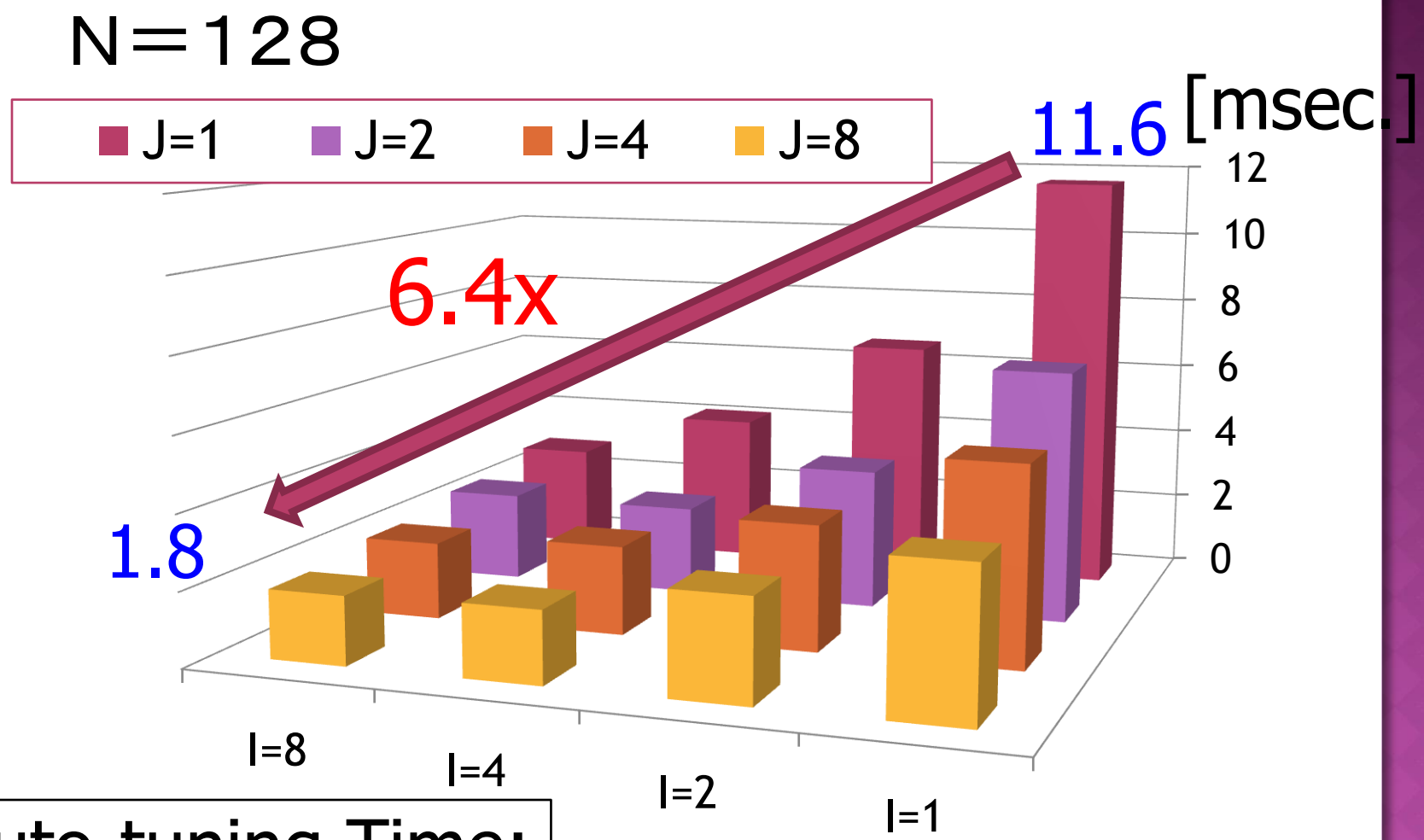
```
#pragma ABCLib install unroll (i,j) region start
#pragma ABCLib varied (i,j) from 1 to 8 sampled  1,2,4,8
for (i=0; i<n; i++) {
   for (j=0; j<n; j++) {
     for (k=0; k<n; k++) {
       C[i][j] += A[i][k] * B[k][j];
     }
   }
 }
#pragma ABCLib install unroll (i,j) region end
```

SPECIFIER FOR ABCLIBSCRIPT ON FFT / IFFT

```
#pragma ABCLib install unroll (j)
region start
#pragma ABCLib varied (j) from 1 to 8
sampled  1,2,4,8
BlockEnd = 1;
for (BlockSize = 2;
     BlockSize <= NumSamples;
     BlockSize <<= 1 )  {
  double delta_angle = angle_numerator
      /(double)BlockSize;
  double sm2 = sin ( -2 * delta_angle );
  double sm1 = sin ( -delta_angle );
  double cm2 = cos ( -2 * delta_angle );
  double cm1 = cos ( -delta_angle );
  double w = 2 * cm1;
  double ar[3];  double ai[3];
```

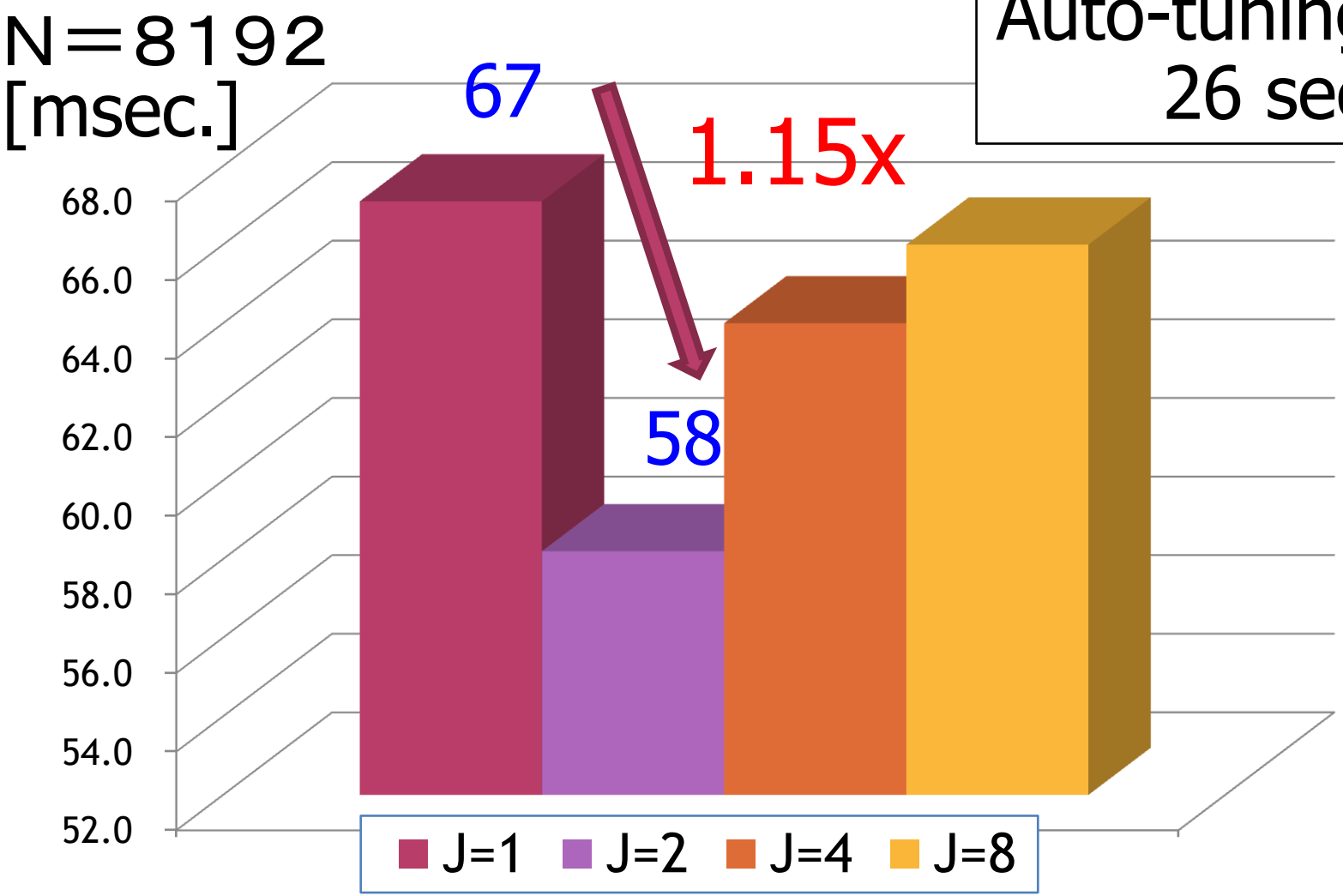
```
for ( i=0; i < NumSamples; i += BlockSize ) {
   ar[2] = cm2; ar[1] = cm1;
   ai[2] = sm2; ai[1] = sm1;
   for ( j=i, n=0; n < BlockEnd; j++, n++ ) {
      ar[0] = w*ar[1] - ar[2];
      ar[2] = ar[1]; ar[1] = ar[0];
      ai[0] = w*ai[1] - ai[2];
      ai[2] = ai[1]; ai[1] = ai[0];
      k = j + BlockEnd;
      tr = ar[0]*RealOut[k] − ai[0]*ImagOut[k];
      ti = ar[0]*ImagOut[k] + ai[0]*RealOut[k];
      RealOut[k] = RealOut[j] − tr;
      ImagOut[k] = ImagOut[j] - ti;
      RealOut[j] += tr; ImagOut[j] += ti;
   }
 }
 BlockEnd = BlockSize;
}
#pragma ABCLib install unroll (j)
region end
```

RESULT (MAT-MAT MULTIPLICATION)



Auto-tuning Time:
2 min. 32 sec.

RESULT (FFT/IFTT)



Auto-tuning Time
26 sec.

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 to establish 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.

FOR MORE INFORMATION

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