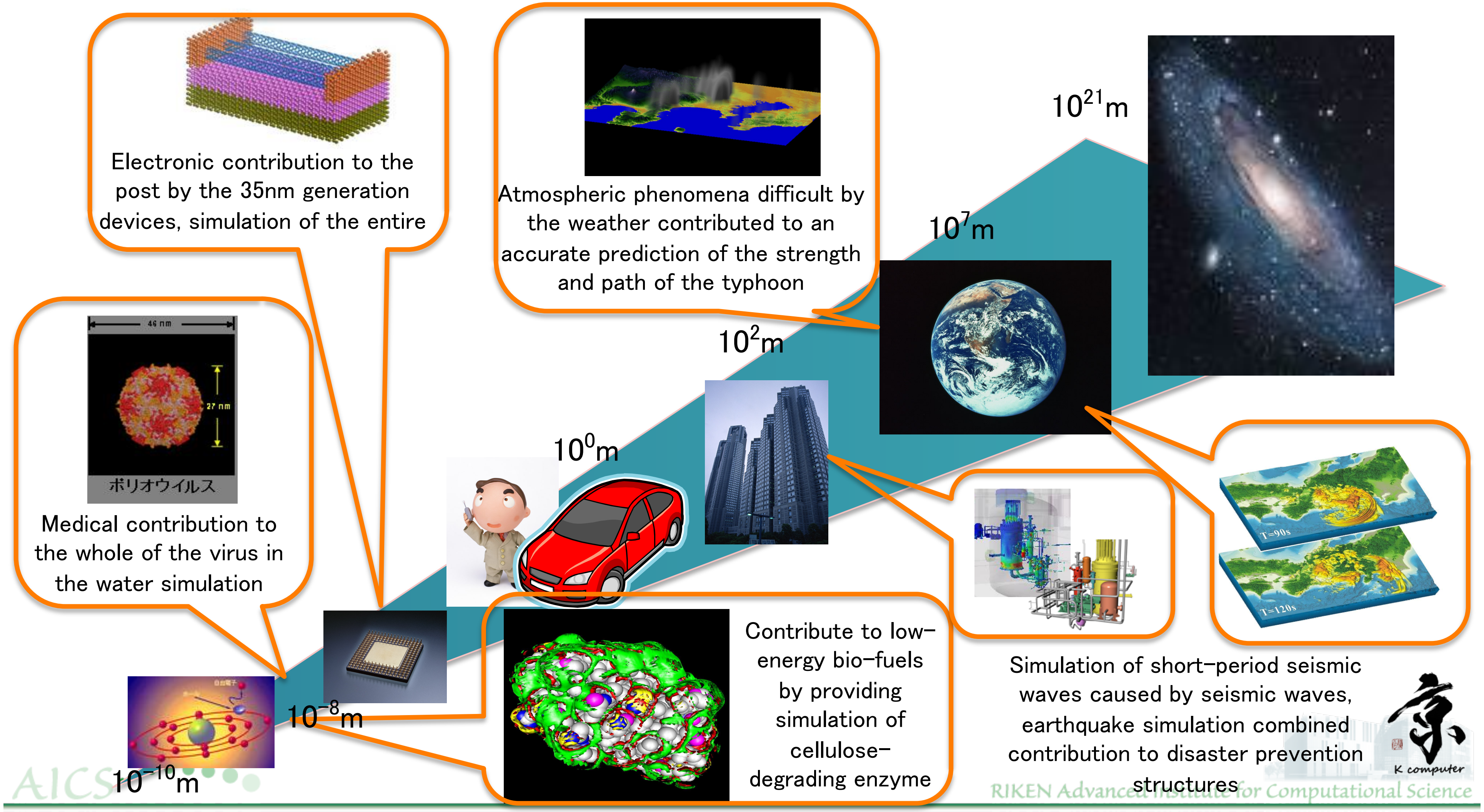
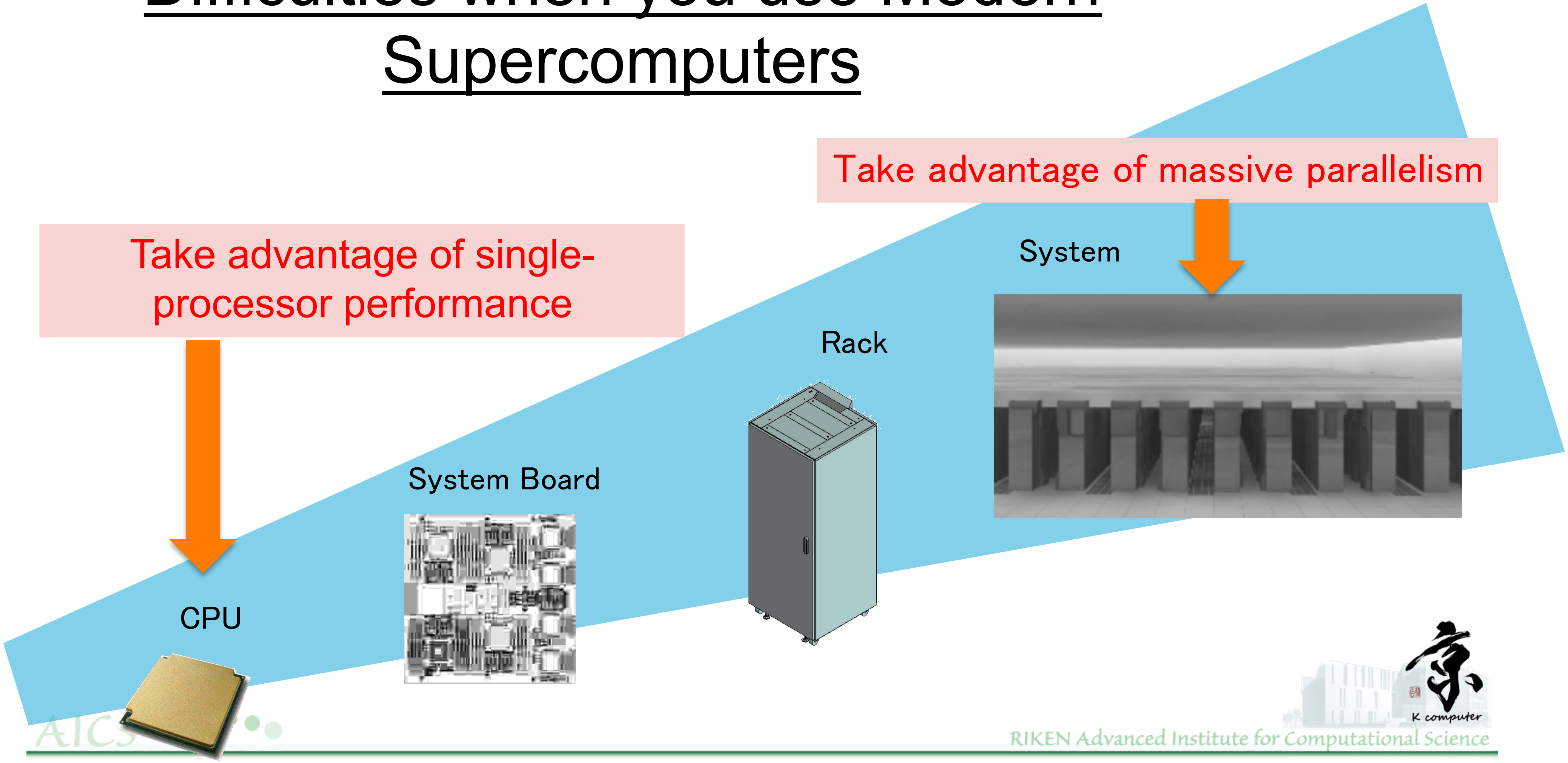
“K computer”: Toward Application Performance Improvement

November 16-17, 2010

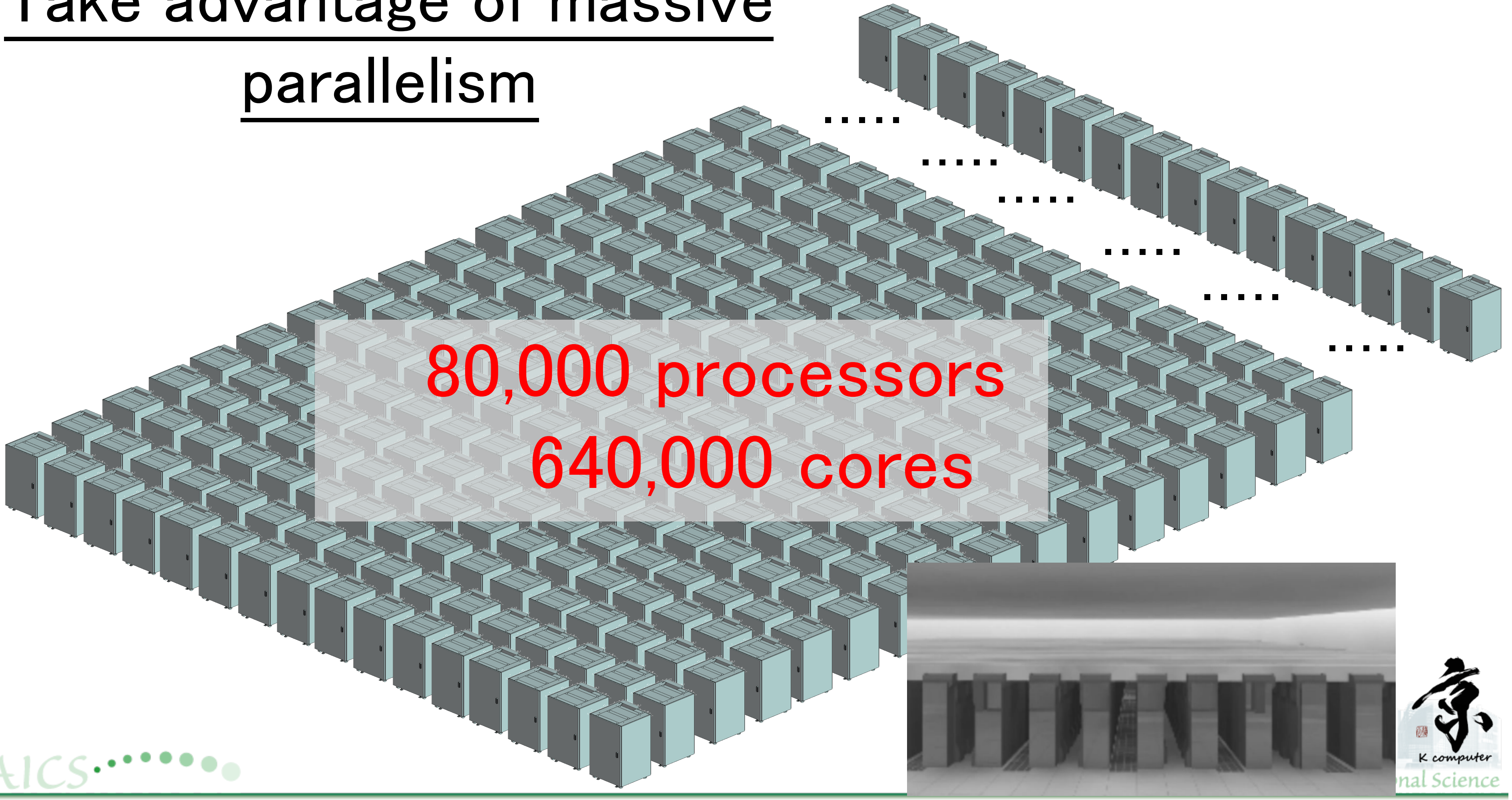
Kazuo Minami
Team Leader, Application Development Team
Research and Development Group
Next-Generation Supercomputer R & D Center
RIKEN


Electronic contribution to the post by the 35nm generation devices, simulation of the entire
Atmospheric phenomena difficult by the weather contributed to an accurate prediction of the strength and path of the typhoon
10^{21}m
10^{7}m
10^{2}m
10^{0}m
46 nm
27 nm
ポリオウイルス
Medical contribution to the whole of the virus in the water simulation
10^{-8}m
10^{-10}m
Contribute to low-energy bio-fuels by providing simulation of cellulose-degrading enzyme
T=90s
T=120s
Simulation of short-period seismic waves caused by seismic waves, earthquake simulation combined contribution to disaster prevention structures
AICS
京
K computer
RIKEN Advanced Institute for Computational Science

Difficulties when you use Modern Supercomputers



Take advantage of massive parallelism



Improvement of Applications Performance at RIKEN

Purpose

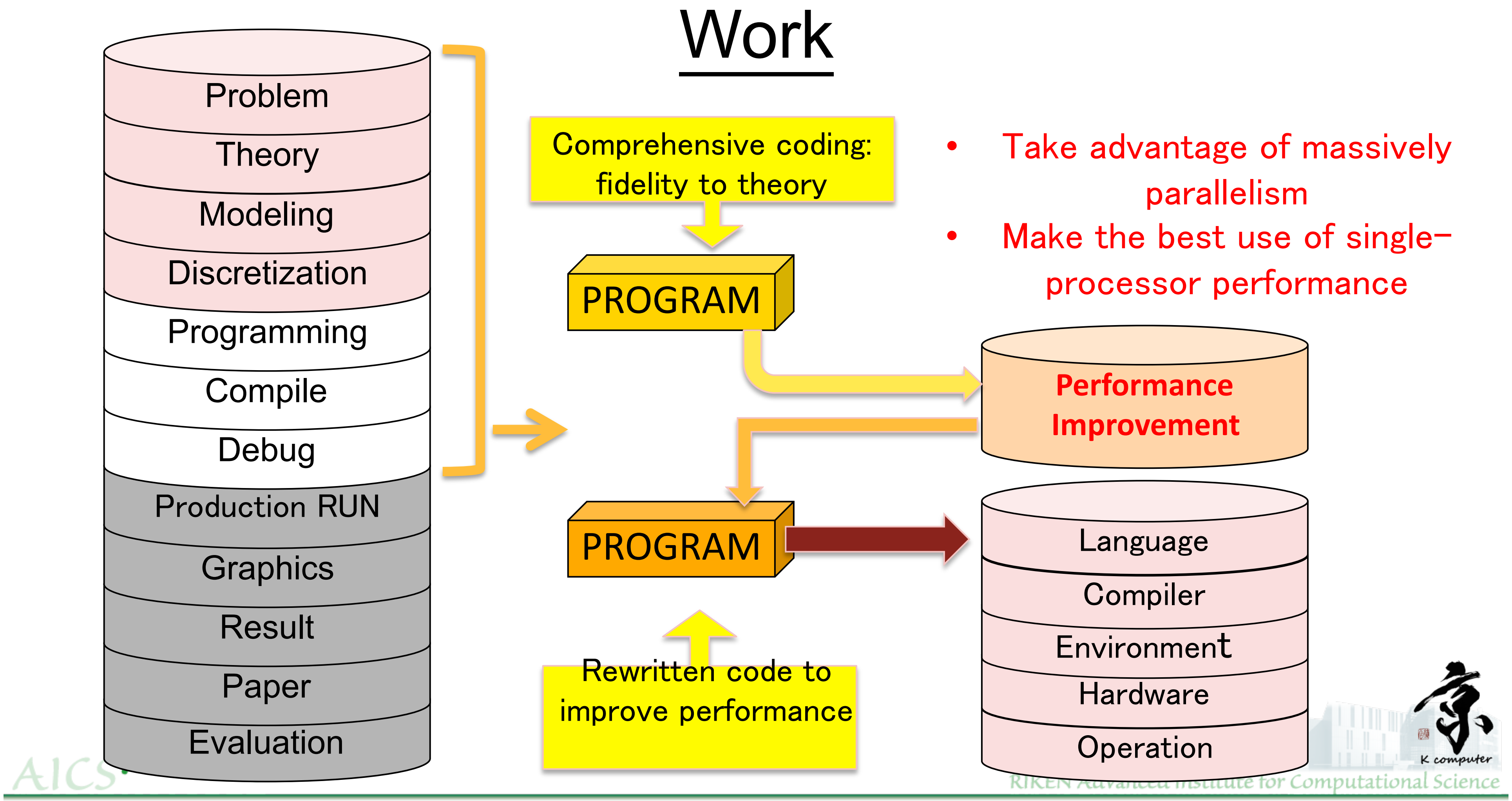
- To check application’s performance, prior to the operation of K computer

Select some applications
(Nano/Engineering/Earth Science/Physics)

- By considering, application’s domain and computing characteristics; *B／F or parallelism*

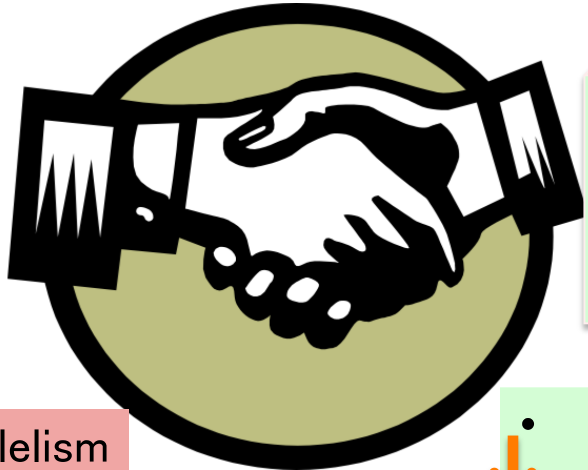
Target Applications

Program Name	Discipline	Outline	Behavior in Computational Science	Scheme
NICAM	earth science	Nonhydrostatic ICosahedral Atmospheric Model (NICAM) for Global-Cloud Resolving Simulations	In Earth Simulator the peak performance ratio was 25-40%, however, large value of Byte/FLOP is required. The single CPU tuning is essential by using K computer.	FDM (atmosphere)
Seism3D	earth science	Simulation of Seismic-Wave Propagation and Strong Ground Motions		FDM (wave)
FrontFlow/ Blue	engineering	Unsteady Flow Analysis based on Large Eddy Simulation (LES)		FEM (fluid)
PHASE	material science	First-Principles Simulation within the Plane-Wave Pseudo potential formalism	Single processor tuning is available by applying matrix multiplication to the kernel. However, the lack of parallelism occurs in the original parallel approach. The development of parallelism is required.	DFT (plane wave)
RSDFT	material science	Ab-initio Calculation in Real Space		The real-space DFT
LatticeQCD	physics	Study of elementary particle and nuclear physics based on Lattice QCD simulation	Single processor tuning by using K computer and parallel tuning based on Tofu-topology are necessary.	QCD


Work
Problem
Theory
Modeling
Discretization
Programming
Compile
Debug
Production RUN
Graphics
Result
Paper
Evaluation
Comprehensive coding: fidelity to theory
PROGRAM
Take advantage of massively parallelism
Make the best use of single-processor performance
Performance Improvement
PROGRAM
Language
Compiler
Environment
Hardware
Operation
Rewritten code to improve performance
AICS
K computer
RIKEN Advanced Institute for Computational Science

Collaboration

Application Developers



RIKEN

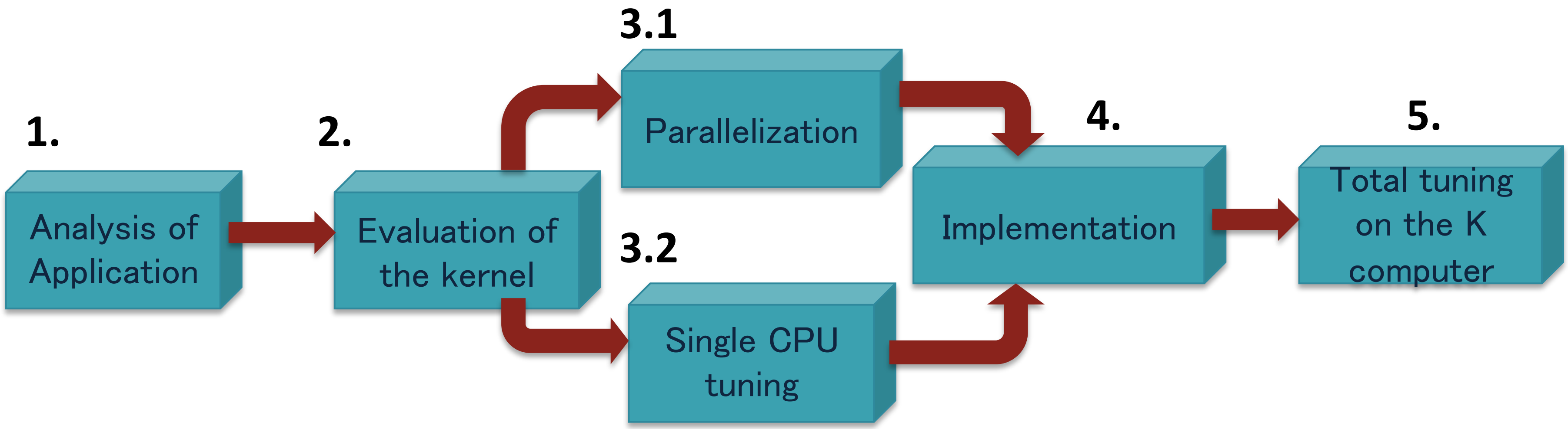
Application Developers:

- Studying massive parallelism and high-performance
- Developing the code using test samples

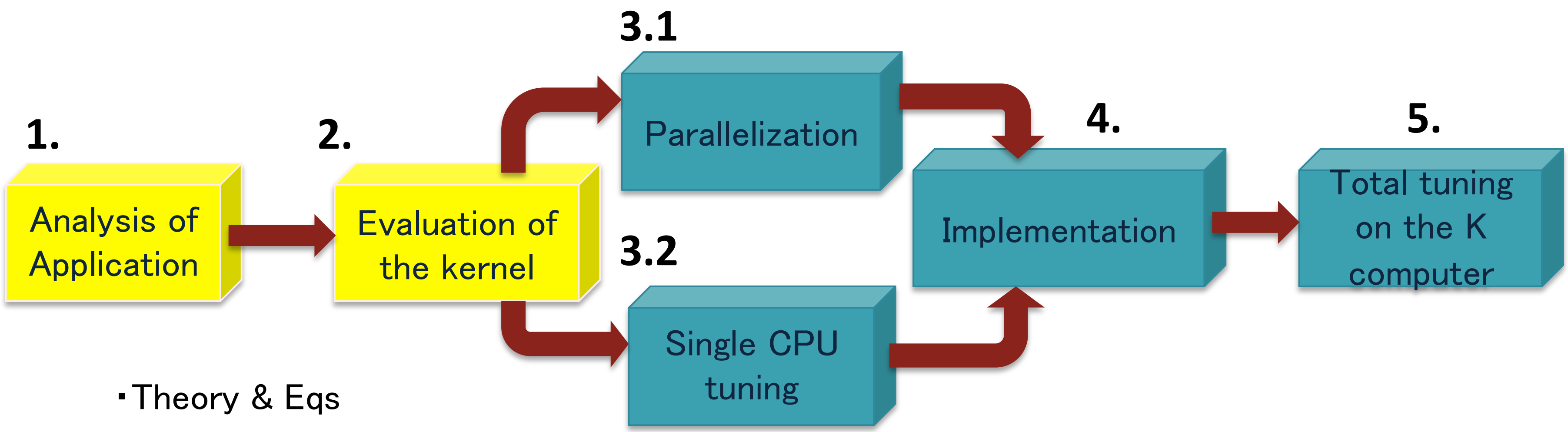
RIKEN:

- Studying massive parallelism and high-performance
- Trial parallelization and performance tuning based on knowledge of hardware

Procedures of Performance Improvement



Analysis of Application
Evaluation of the kernel



- Theory & Eqs
- Structure of code
- Algorithms

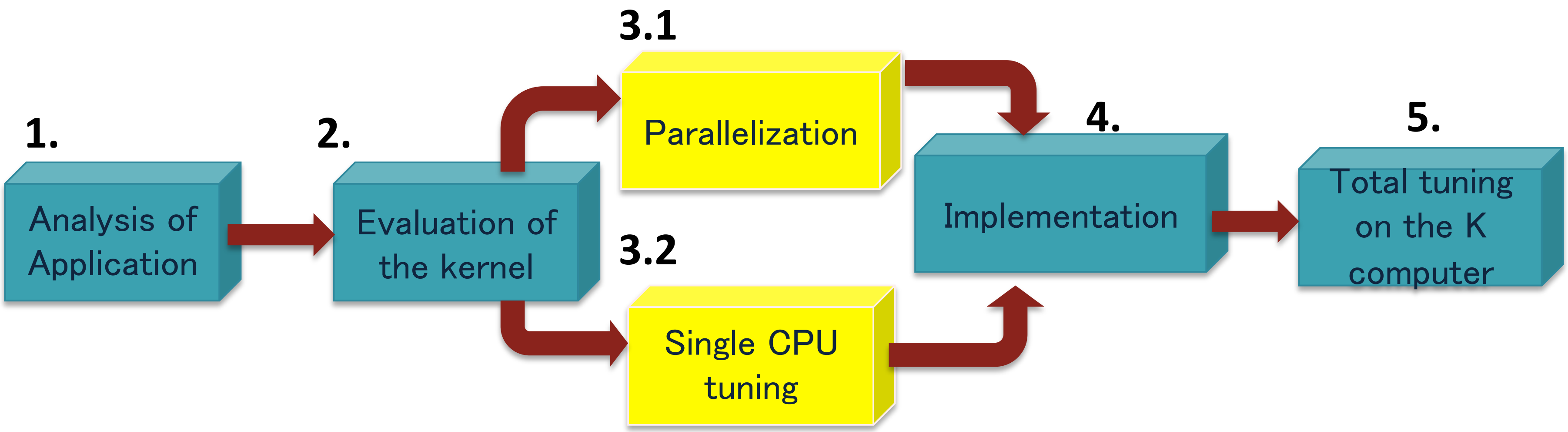
1. Analysis of Application

```
mainA
└─ subB
    ├─ subC
    ├─ subD
    ├─ DO loop
    ├─ subE
    │   ├─ Proc.1
    │   │   └─ Proc.1.1
    │   ├─ DO loop
    │   ├─ Proc.2
    │   │   └─ Proc.2.1
    │   ├─ Proc.3
    │   └─ Comm.1
    ├─ subF
    │   └─ subG
    └─
```

	Calc. Time Scalability	physical processes	Characte-ristics	estimation	Kernel
Block 1 (Calc.)			Partial parallel	N proportional	
Block 2 (Calc.)			Fully parallel	N^3 proportional	○
(Comm.)			Neighbor comm.	Proportional to the adjacent surface	○
Block 3					

2. Evaluation of the kernel

Parallelization & Single CPU Tuning



3.1 Parallelization

Key points for aiming at High Parallel

(1) Does non-parallel parts remain there？ If so, no problem？
(2) Is load imbalance getting worse at high parallel？
(3) How much does the neighboring comm. time occupy at high parallel？
(4) How much does the global comm. time increase at high parallel ？

⇒ *These evaluations are crucial*

Strong scaling：measuring performance by increasing parallelism under the constant scale of the whole problem
Weak scaling： measuring performance by increasing parallelism under the constant scale of one CPU problem

Approach

(1) Setting a target problem
(2) Making a test sample (100 parallelism)
(3) Measuring and evaluating the test sample; execution time, load imbalance, communication time between neighbors, global communication
(4) If no problem, measuring parallel performance through weak scaling
(5) If not, measuring parallel performance through strong scaling and find its cause

Performance measurement for massive parallelism

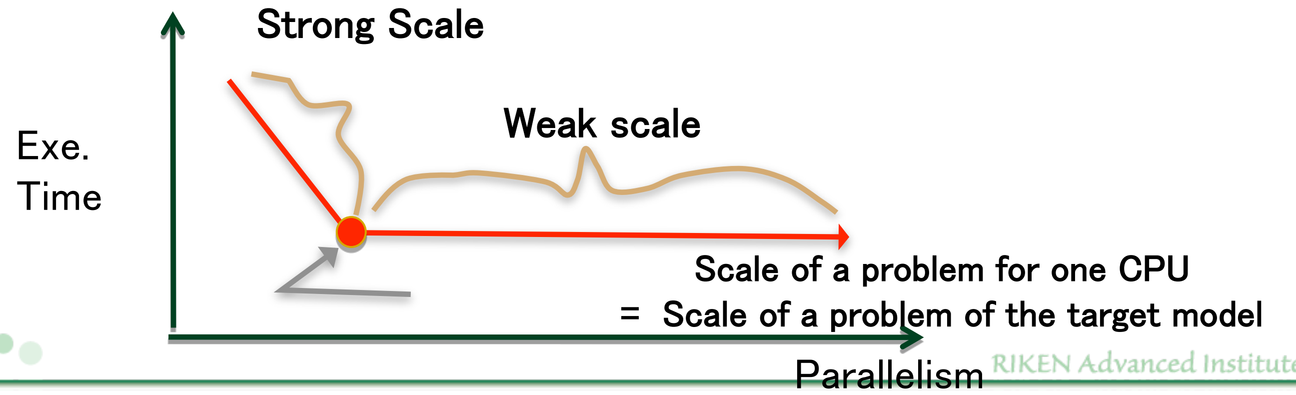
(1)Use available machine resource. Weak Scaling measurement, varying parallelism of O(100 – 1000 – massive) and observe the characteristics.

(2) Recommend Weak scaling measurement even though some difficult cases

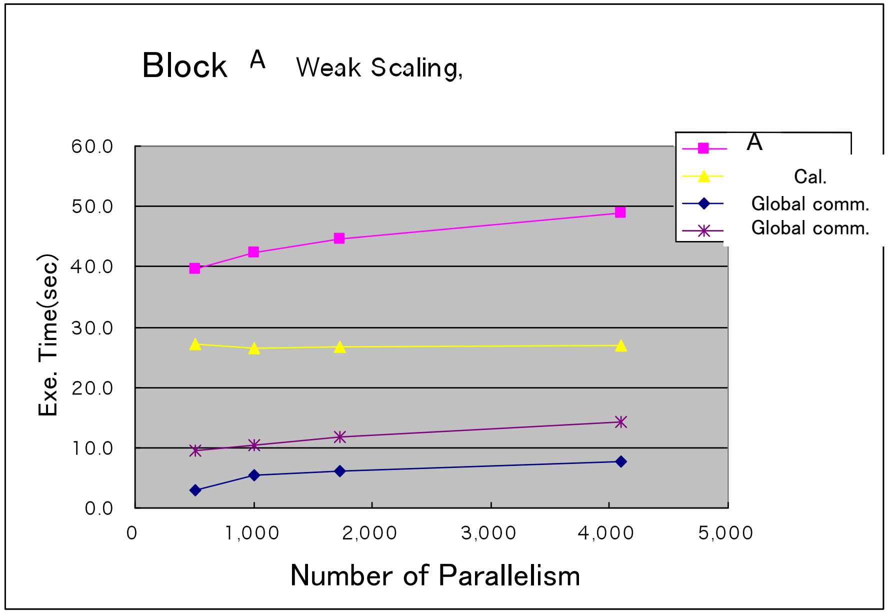
(3) Observe performances of exec time, load imbalance, neighboring and global communication, focusing on

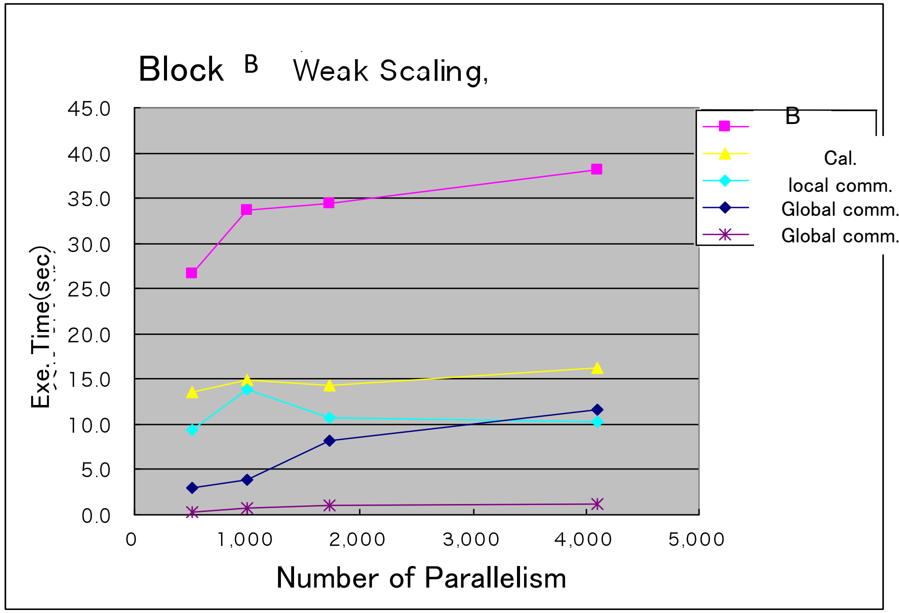
- → **Increase of exec. time ? ･･･ there still remains non-parallel parts**
- → **Increase of load imbalance ?**
- → **Increase of neighboring communication ?**
- → **Rate of increase of global communication ?**

(4) if communication counts and volume are well evaluated after measurements through step1 and 2, we can estimate the communication load on the K computer



Weak Scaling measurement





Cal.
Global comm.

3.2 Single CPU Tuning

(1) Extract kernels

→making them the independent test programs

(2) Trials for increasing performance

→ applying ideas using K computer

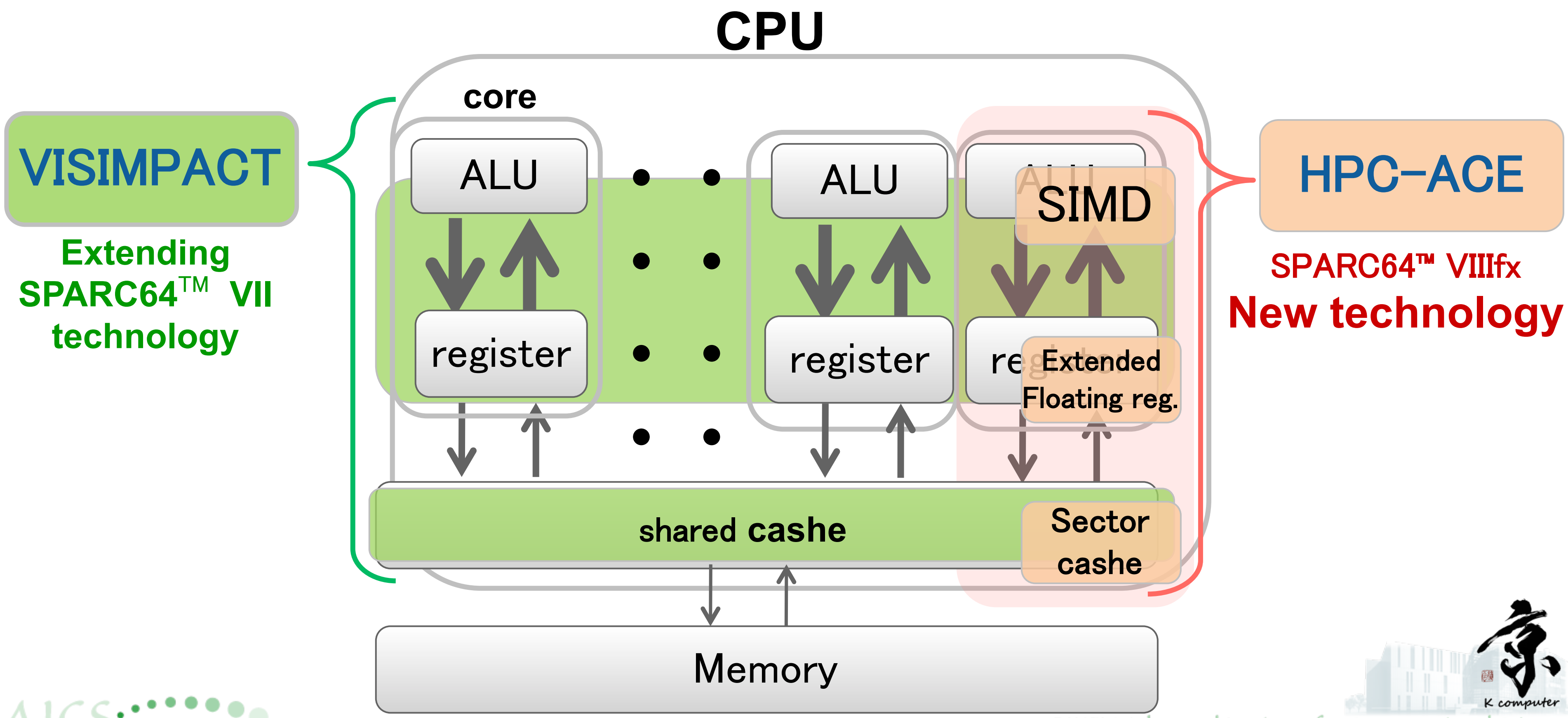
(3) Estimate the work volume

→ make it clear that what the impact on the whole code is and estimate the amount of work volume for introducing the performance model.

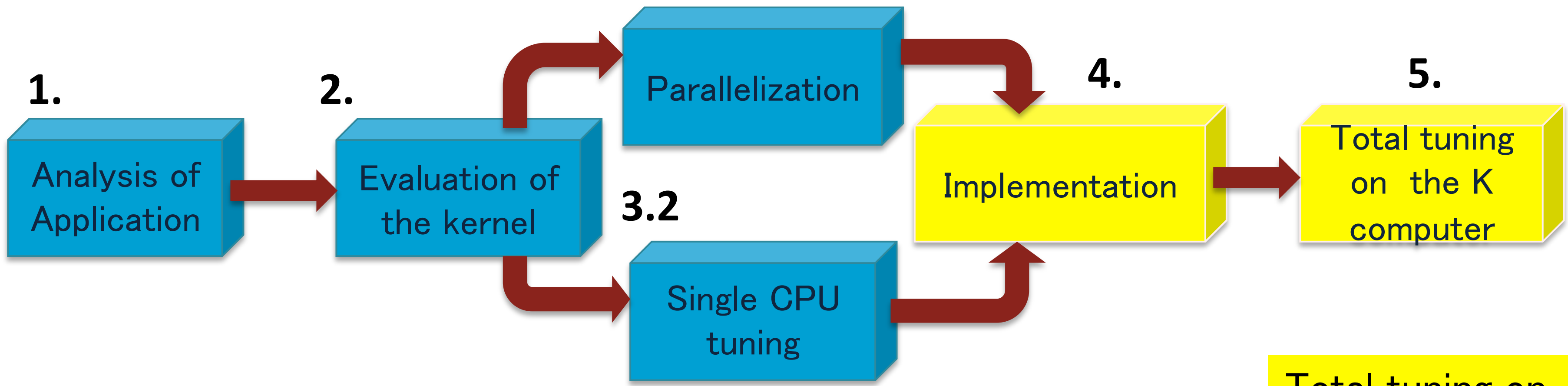
(4) Fix the model of increasing performance

→ evaluate the tests and select and fix the best.

Increase performance of SPARC64™ VIIIfx



Mounting high-performance model and Tuning for K computer



Implementation (Mounting performance model)

Apply the single CPU performance model to the massive parallelized code.
Estimated to be large amount of work volume for design , programming, debugging.

Total tuning on the K computer

Tuning the code for K computer

(1)Measurement

・Parallel, single CPU performance .

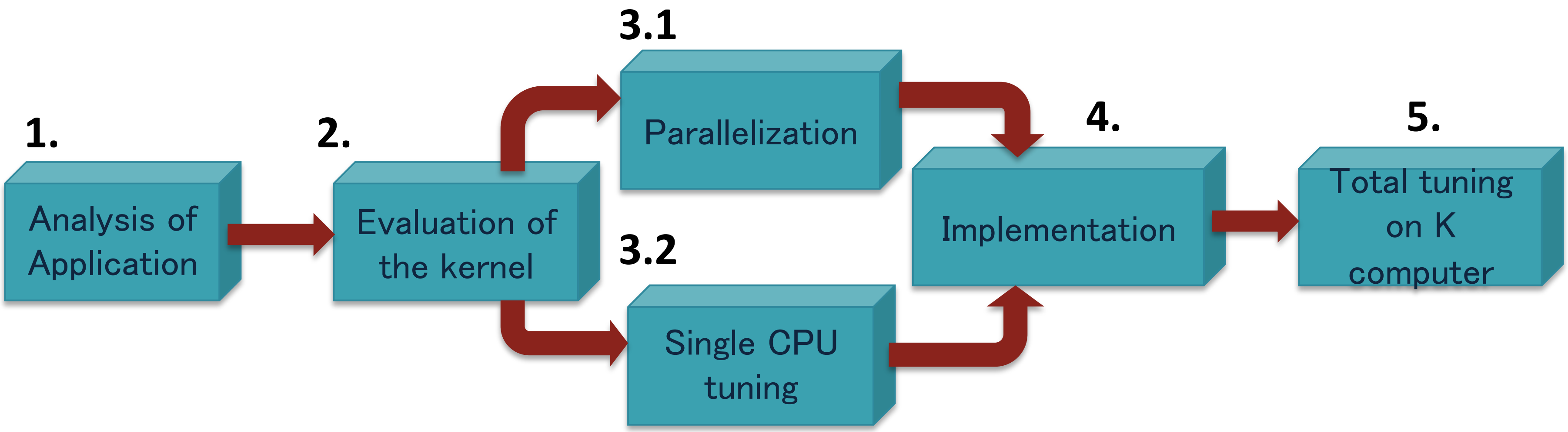
(2)Cheking and finding out

・Performance problems

(3)Problem solving

・Solving the problems and apply the solution

Summary



- It is important for users to respond massive parallelism over 10K–100K and make efforts to increase single CPU performance for K computer