“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
Difficulties when you use Modern Supercomputers

Take advantage of single-processor performance

Take advantage of massive parallelism
Take advantage of massive parallelism

80,000 processors
640,000 cores
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
<table>
<thead>
<tr>
<th>Program Name</th>
<th>Discipline</th>
<th>Outline</th>
<th>Behavior in Computational Science</th>
<th>Scheme</th>
</tr>
</thead>
<tbody>
<tr>
<td>NICAM</td>
<td>earth science</td>
<td>Nonhydrostatic ICosahedral Atmospheric Model (NICAM) for Global–Cloud Resolving Simulations</td>
<td>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.</td>
<td>FDM (atmosphere)</td>
</tr>
<tr>
<td>Seism3D</td>
<td>earth science</td>
<td>Simulation of Seismic–Wave Propagation and Strong Ground Motions</td>
<td></td>
<td>FDM (wave)</td>
</tr>
<tr>
<td>FrontFlow/Blue</td>
<td>engineering</td>
<td>Unsteady Flow Analysis based on Large Eddy Simulation (LES)</td>
<td></td>
<td>FEM (fluid)</td>
</tr>
<tr>
<td>PHASE</td>
<td>material science</td>
<td>First-Principles Simulation within the Plane-Wave Pseudo potential formalism</td>
<td>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.</td>
<td>DFT (plane wave)</td>
</tr>
<tr>
<td>RSDFT</td>
<td>material science</td>
<td>Ab-initio Calculation in Real Space</td>
<td></td>
<td>The real–space DFT</td>
</tr>
<tr>
<td>LatticeQCD</td>
<td>physics</td>
<td>Study of elementary particle and nuclear physics based on Lattice QCD simulation</td>
<td>Single processor tuning by using K computer and parallel tuning based on Tofu–topology are necessary.</td>
<td>QCD</td>
</tr>
</tbody>
</table>
Work

- Take advantage of massively parallelism
- Make the best use of single-processor performance

Comprehensive coding: fidelity to theory

Rewritten code to improve performance

Performance Improvement

Language
Compiler
Environment
Hardware
Operation

Production RUN
Graphics
Result
Paper
Evaluation

Debug
Programming
Discretization
Modeling
Theory
Problem
Collaboration

**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

1. Analysis of Application
2. Evaluation of the kernel
3.1 Parallelization
3.2 Single CPU tuning
4. Implementation
5. Total tuning on the K computer
Analysis of Application
Evaluation of the kernel

1. Analysis of Application
   - Theory & Eqs
   - Structure of code
   - Algorithms
2. Evaluation of the kernel
3.1 Parallelization
3.2 Single CPU tuning
4. Implementation
5. Total tuning on the K computer
1. Analysis of Application

<table>
<thead>
<tr>
<th>subA</th>
<th>subB</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
</tr>
</tbody>
</table>

- subC
- subD
- DO loop
- subE
  - Proc.1
    - Proc.1.1
  - Proc.2
    - Proc.2.1
  - Proc.3
- Comm.1
- subF
- subG

<table>
<thead>
<tr>
<th>Calc. Time</th>
<th>physical processes</th>
<th>Characteristics</th>
<th>estimation</th>
<th>Kernel</th>
</tr>
</thead>
<tbody>
<tr>
<td>Block 1</td>
<td>Partial parallel</td>
<td>N proportional</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Block 2</td>
<td>Fully parallel</td>
<td>N³ proportional</td>
<td>Proportional to the adjacent surface</td>
<td></td>
</tr>
<tr>
<td>Block 3</td>
<td>Neighbor comm.</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

2. Evaluation of the kernel
Parallelization & Single CPU Tuning

1. Analysis of Application
2. Evaluation of the kernel
3.1 Parallelization
3.2 Single CPU tuning
4. Implementation
5. Total tuning on the K computer
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

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

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
Performance measurement for massive parallelism

(1) Use available machine resource. Weak Scaling measurement, varying parallelism of $O(100 - 1000 \rightarrow \text{massive})$ 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

**Block A**

<table>
<thead>
<tr>
<th>Number of Parallelism</th>
<th>Exec. Time (sec)</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>10.0</td>
</tr>
<tr>
<td>1,000</td>
<td>10.0</td>
</tr>
<tr>
<td>2,000</td>
<td>20.0</td>
</tr>
<tr>
<td>3,000</td>
<td>30.0</td>
</tr>
<tr>
<td>4,000</td>
<td>40.0</td>
</tr>
<tr>
<td>5,000</td>
<td>50.0</td>
</tr>
</tbody>
</table>

**Block B**

<table>
<thead>
<tr>
<th>Number of Parallelism</th>
<th>Exec. Time (sec)</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>5.0</td>
</tr>
<tr>
<td>1,000</td>
<td>10.0</td>
</tr>
<tr>
<td>2,000</td>
<td>15.0</td>
</tr>
<tr>
<td>3,000</td>
<td>20.0</td>
</tr>
<tr>
<td>4,000</td>
<td>25.0</td>
</tr>
<tr>
<td>5,000</td>
<td>30.0</td>
</tr>
</tbody>
</table>

**Legend**

- 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™ VIII fx

VISIMPACT
Extending SPARC64™ VII technology

HPC-ACE
SPARC64™ VIII fx
New technology
Mounting high-performance model and Tuning for K computer

(1) Measurement
- Parallel, single CPU performance

(2) Checking and finding out
- Performance problems

(3) Problem solving
- Solving the problems and apply the solution

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.
Summary

1. Analysis of Application
2. Evaluation of the kernel
3.1 Parallelization
3.2 Single CPU tuning
4. Implementation
5. Total tuning on K computer

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