Benchmarking Parallel Java

Project Proposal

Asma'u Sani Mohammed

Committee Chair: Prof. Alan Kaminsky
Reader: Prof. James Heliotis
Department of Computer Science
B. Thomas Golisano College of Computing and Information Sciences
Rochester Institute of Technology
Rochester, New York
TABLE OF CONTENTS

1. PROJECT SUMMARY ........................................................................................................... 3

2. OVERVIEW .......................................................................................................................... 3
   2.1 Background ...................................................................................................................... 3
   2.2 Glossary .......................................................................................................................... 3

3. HYPOTHESIS ....................................................................................................................... 4

4. EVALUATION AGAINST EXISTING WORK ..................................................................... 4

5. ARCHITECTURAL OVERVIEW AND DESIGN SPECIFICATION ..................................... 4
   5.1 Input Parameters/Class size ........................................................................................... 5
   5.2 Equipment Configuration ............................................................................................... 5
   5.3 Implementation Languages ............................................................................................ 5
   5.4 NAS Parallel Benchmarks ............................................................................................. 5
      5.1.1 EP ............................................................................................................................. 5
      5.1.2 CG ............................................................................................................................. 5
      5.1.3 IS ............................................................................................................................... 6
      5.1.3 BT .............................................................................................................................. 6
      5.1.3 SP .............................................................................................................................. 6
      5.1.3 LU .............................................................................................................................. 6

6. PRINCIPAL DELIVERABLES AND MILESTONES ............................................................ 6
   6.1 Deliverable Items .............................................................................................................. 6
   6.2 Milestone Identification ................................................................................................... 7
   6.3 Milestone Completion Criteria ........................................................................................ 7

7. PROJECT SCHEDULE ......................................................................................................... 7

8. CURRENT STATUS OF WORK ......................................................................................... 8

9. REFERENCES ....................................................................................................................... 8
1. **PROJECT SUMMARY**

Parallel computing has become popular amongst the scientific community and areas where large data computation is needed. The need for computationally efficient parallel middleware has become increasingly important as hardware architectures evolve to allow parallelism. My project will examine the efficiency of Parallel Java by implementing the OpenMP version of the NAS Parallel Benchmark. This will allow us to see how well the Parallel Java directives compare to the OpenMp shared memory architecture directives.

2. **OVERVIEW**

2.1 **BACKGROUND**

The level of parallelism supported by parallel computers varies significantly, from symmetric multiprocessing, distributed computing and multi-core computing. Various APIs exist for programming parallel computers. Each library or API makes assumptions based on the underlying computer architecture. This wide availability of computing systems and difference in computer architecture makes it a challenge to objectively assess the performance of these parallel middleware libraries. Parallel benchmark codes such as the SPEC HPC 2002 Benchmarks, the OMP Benchmarks, the Pallas MPI Benchmarks, and the NAS Parallel Benchmarks (NPB), have been developed to aid in comparing the performance of parallel middleware across different architectures. NPB is an Open source benchmark developed by NASA's Advanced Supercomputing (NAS) division. The NPB suite is among the most highly used open source benchmarks out there. My project will involve porting the OpenMP implementation of the NPB 3 specification [3] to Parallel Java. NPB 3 has a total of eleven benchmarks, seven of which are implemented for OpenMP version. Three of the seven, Block Tridiagonal (BT), Scalar Pentdiagonal (SP) and Lower and Upper triangular systems (LU) are simulated Computational Fluid Dynamics (CFD) applications that solve a synthetic system of nonlinear partial differential equations (PDE) using three different algorithms. The MultiGrid (MG) solves a three-dimensional discrete Poisson equation using the V-cycle multigrid method. The Fast Fourier Transform (FT) uses the Fast Fourier Transform to solve a 3-D partial differential equation. Embarrassingly Parallel (EP) generates pairs of Gaussian random variates according to a specific scheme. The last benchmark is the Conjugate Gradient (CG) which estimates the smallest eigenvalue of a matrix using the conjugate gradient method as a subroutine for solving systems of linear equations. These Benchmarks are written in FORTRAN, which I will need to translate to Parallel Java.

2.2 **GLOSSARY**

This section defines acronyms used in the proposal.

<table>
<thead>
<tr>
<th>Acronym</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>OpenMP</td>
<td>Open Multi-Processing</td>
</tr>
<tr>
<td>EP</td>
<td>Embarrassingly Parallel</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>MPI</th>
<th>Message Passing Interface</th>
</tr>
</thead>
<tbody>
<tr>
<td>PJ</td>
<td>Parallel Java</td>
</tr>
<tr>
<td>CG</td>
<td>Conjugate Gradient</td>
</tr>
<tr>
<td>BT</td>
<td>Block Tridiagonal</td>
</tr>
<tr>
<td>NPB</td>
<td>NAS Parallel Benchmark</td>
</tr>
<tr>
<td>SP</td>
<td>Scalar Pentdiagonal</td>
</tr>
<tr>
<td>LU</td>
<td>Lower and Upper triangular systems</td>
</tr>
<tr>
<td>MG</td>
<td>MultiGrid</td>
</tr>
<tr>
<td>PDE</td>
<td>Partial Differential Equation</td>
</tr>
<tr>
<td>FT</td>
<td>Fast Fourier Transform</td>
</tr>
<tr>
<td>NAS</td>
<td>Numerical Aerodynamic Simulation</td>
</tr>
</tbody>
</table>

3. **HYPOTHESIS**

Parallel Java is effective in parallelizing the NAS Parallel Benchmarks. This will be gauged by porting the OpenMP version of the NPB to PJ and compare PJ's performance based on the calculated speedup, sizeup and efficiency of the standard benchmarks to the performance of the OpenMP version.

4. **EVALUATION AGAINST EXISTING WORK**

To my knowledge Parallel Java has not been benchmarked using the NPB. Therefore evaluation of the project will be done based on results submitted on the NPB implementation in OpenMP and MPI. These results can be found in the NAS Supercomputing website under results.

In order to make an objective comparison between the OpenMP and PJ benchmark implementations, the OpenMP implementation will be run on the same 48 core machine that the PJ benchmarks will be run on. The metrics mentioned (sizeup, speedup, and efficiency) will then be computed and compared to the PJ implementation. The two benchmark implementations will be tested on 1, 2, 3, 4, 6, 8, 12, 16, 24, 32 and 48 cores. These cores were chosen to allow an even coverage on a log scale.

5. **ARCHITECTURAL OVERVIEW AND DESIGN SPECIFICATION**

The project will be divided into three phases. The first will be the analysis and development phase. This phase will take up the most amount of time. It involves understanding the algorithms and rules set by NAS for implementing every benchmark. The Second phase will be the deployment phase. In this phase, the implemented benchmarks will be run according to the
configurations mentioned in section 2.2.4. The results will be collected and tallied. The third and final phase is the reporting phase. This involves doing a comparison between the performances of the OpenMP and Parallel Java implementations in order to prove the hypothesis mentioned in section 1.2.

5.1 INPUT PARAMETERS/CLASS SIZE

The OpenMP version of NPB implemented the Class A, B, C, W and S problem size. Each Class denotes a different size in input parameters. I will be implementing the same problem sizes of NPB in order to do a valid comparison between the two parallel directives.

5.2 EQUIPMENT CONFIGURATION

The Parallel Java and OpenMP benchmarks will be run on the same machine to ensure uniform architecture and consistent performance. Dr. Matthew Fluet has agreed to provide me with access to a 48 core machine to be used in running the benchmarks. The machine has already been configured to run Parallel Java programs as well as Fortran 77 programs. Additional configurations may be made as needed.

5.3 IMPLEMENTATION LANGUAGES

The four benchmarks (CG, LU, BT, and SP) will be implemented using the Parallel Java library.

5.4 NAS PARALLEL BENCHMARKS

Below is an overview of the benchmarks mentioned in section 1.3. Of the eight benchmarks, four will be implemented in this project (CG, BT, LU and SP). The MG, IS, EP and FT benchmarks will be implemented only if time permits.

5.1.1 EP

The EP benchmark generates a system of Gaussian deviates according to a particular scheme as described in the NAS nmr-94-007 spec. This benchmark is designed to exhibit the classic embarrassingly parallel type of computation where the problem can be easily divided into independent parts that can be solved by different processors.

5.1.2 CG

The CG benchmark finds the largest eigenvalue of a symmetric positive definite sparse matrix using the inverse power method. This is one of the more straightforward algorithms for parallelization. The solution z to the linear equation Az = x is to be computed using the Conjugate Gradient. The matrix A is an unstructured sparse matrix that must be generated using the matrix generation routine described in [1]. A Fortran 77 subroutine “makea” is provided which must be used by all those implementing this benchmark. This subroutine will have to be rewritten in java.
5.1.3 IS

IS is an integer sort that is applicable in “particle method” codes. The IS benchmark is intended to test both integer operation speed and communication performance [2].

5.1.3 BT

BT is an I/O intensive Computational Fluids Dynamics Application that solves a system of Partial Differential Equation (the Naïve Strokes Equation).

5.1.3 SP

SP defines a solution of multiple, independent systems of nondiagonally dominant scalar pentadiagonal equations [1]. This is a computationally intensive CFD application much like BT. The Beam-warning approximate factorization is used to breakdown the equation into the x, y and z dimensions, which are then solved along each dimension. The detailed algorithm can be found in [1].

5.1.3 LU

LU solves 5 X 5 sparse matrix blocks of lower and upper triangular systems. This is another CFD application that solves the Navier-Stokes equation using the symmetric successive over-relaxation (SSOR) method.

6. PRINCIPAL DELIVERABLES AND MILESTONES

6.1 DELIVERABLE ITEMS

<table>
<thead>
<tr>
<th>Reference Number</th>
<th>Deliverable</th>
<th>Files Included</th>
<th>Description of file</th>
</tr>
</thead>
<tbody>
<tr>
<td>DI.1</td>
<td>Program executable files</td>
<td>EP.java, CG.java, BT.java, SP.java, LU.java, IS.java</td>
<td>These files are java source files that contain the implementation of the benchmarks to be implemented in this project. Other supporting files may be included here as the implementation phase begins.</td>
</tr>
</tbody>
</table>
6.2 MILESTONE IDENTIFICATION

This section contains a list of the project milestones for all three phases.

- Completion of Parallel Java source code for all six Benchmarks
- Completion of a detailed report on performance of each Parallel Java Benchmark
- Completion of a detailed report on performance comparison on Parallel Java and OMP
- Completion of the final project report

6.3 MILESTONE COMPLETION CRITERIA

Each milestone mentioned above is to be approved by my chair, Prof. Alan Kaminsky of the C.S. department. Acceptance of the milestone and advancement to another milestone will all be determined by Prof. Kaminsky.

7. PROJECT SCHEDULE

This section lists the milestones mention in section 2.3.2 above, broken into finer detail and grouped by phases.

Phase 1: Analysis/Development

<table>
<thead>
<tr>
<th>Benchmark</th>
<th>Analysis Time Estimate (in days)</th>
<th>Implementation Time Estimate</th>
<th>Deliverable</th>
</tr>
</thead>
<tbody>
<tr>
<td>CG</td>
<td>01/31/2012 – 02/2/2012</td>
<td>02/3/2012 – 02/6/2012</td>
<td>Parallel Java Source code for CG</td>
</tr>
<tr>
<td>BT</td>
<td>02/14/2012 – 02/17/2012</td>
<td>02/18/2012 – 02/20/2012</td>
<td>Parallel Java Source code for BT</td>
</tr>
<tr>
<td>SP</td>
<td>03/12/2012 – 02/23/2012</td>
<td>03/14/2012 – 02/24/2012</td>
<td>Parallel Java Source code for SP</td>
</tr>
<tr>
<td>LU</td>
<td>03/15/2012 – 02/27/2012</td>
<td>03/17/2012 – 03/1/2012</td>
<td>Parallel Java Source code for LU</td>
</tr>
</tbody>
</table>

Phase 2: Deployment

The table below contains time estimates for running each of the benchmarks and achieving best results.

<table>
<thead>
<tr>
<th>Benchmark</th>
<th>Time Estimate (in days)</th>
<th>Deliverable</th>
</tr>
</thead>
</table>

7
Phase 3: Reporting

The table below gives the time estimate of compiling a report on performance comparisons between the Parallel Java and OMP implantations of the benchmarks.

<table>
<thead>
<tr>
<th>Benchmark</th>
<th>Time estimate</th>
<th>Deliverable</th>
</tr>
</thead>
<tbody>
<tr>
<td>All benchmarks</td>
<td>04/12/2012 – 04/19/2012</td>
<td>Project report</td>
</tr>
</tbody>
</table>

Project Defense:

The target defense date will be May 18th 2011.

8. CURRENT STATUS OF WORK

Below is a list of things that have been accomplished so far.

- Looked at EP, CG, and LU more extensively
- Setup my system to be able to run and debug the OMP implementation.
- Written high level implementation of EP benchmark in Parallel Java
- Read the rules of the benchmarks extensively
- Ran the EP, CG and LU benchmarks locally
- Traced through the BT algorithm, although I need to clarify certain aspects of the algorithm. Dr. Ivona Bezakova has agreed to trace through it with me and answer some of my questions.

9. REFERENCES

