A Platform-Independent Tool for Modeling Parallel Programs

ACM Southeast 2011
Kennesaw State University
March 2011

Ferosh Jacob, Jeff Gray, Yu Sun, Purushotham Bangalore

Department of Computer Science
University of Alabama
fjacob@crimson.ua.edu, gray@cs.ua.edu

Department of Computer and Information Sciences
University of Alabama at Birmingham
{yusun, puri}@cis.uab.edu
Overview

Problem formulation
- Parallel programming models
- Current Challenges
- Possible solution approach

PPmodel
- From Developers viewpoint
- Implementation details
- Related work

Conclusion
Main contributions
Programming Models

CUDA
OpenCL
OpenMPI
p-threads
Cg
OpenMP
• Which programming model to use?

<table>
<thead>
<tr>
<th></th>
<th>19</th>
<th>20</th>
<th>21</th>
<th>22</th>
<th>23</th>
<th>24</th>
</tr>
</thead>
<tbody>
<tr>
<td>OpenMP</td>
<td>0.04</td>
<td>0.1</td>
<td>0.22</td>
<td>0.48</td>
<td>0.79</td>
<td>1.82</td>
</tr>
<tr>
<td>MPI</td>
<td>0.04</td>
<td>0.1</td>
<td>0.19</td>
<td>0.36</td>
<td>0.75</td>
<td>1.55</td>
</tr>
</tbody>
</table>

Parallel Programming Challenges
- **Why are parallel programs long?**

<table>
<thead>
<tr>
<th>No</th>
<th>Program Name</th>
<th>Total LOC</th>
<th>Parallel LOC</th>
<th>No. of blocks</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>2D Integral with Quadrature rule</td>
<td>601</td>
<td>11 (2%)</td>
<td>1</td>
</tr>
<tr>
<td>2</td>
<td>Linear algebra routine</td>
<td>557</td>
<td>28 (5%)</td>
<td>4</td>
</tr>
<tr>
<td>3</td>
<td>Random number generator</td>
<td>80</td>
<td>9 (11%)</td>
<td>1</td>
</tr>
<tr>
<td>4</td>
<td>Logical circuit satisfiability</td>
<td>157</td>
<td>37 (18%)</td>
<td>1</td>
</tr>
<tr>
<td>5</td>
<td>Dijkstra’s shortest path</td>
<td>201</td>
<td>37 (18%)</td>
<td>1</td>
</tr>
<tr>
<td>6</td>
<td>Fast Fourier Transform</td>
<td>278</td>
<td>51 (18%)</td>
<td>3</td>
</tr>
<tr>
<td>7</td>
<td>Integral with Quadrature rule</td>
<td>41</td>
<td>8 (19%)</td>
<td>1</td>
</tr>
<tr>
<td>8</td>
<td>Molecular dynamics</td>
<td>215</td>
<td>48 (22%)</td>
<td>4</td>
</tr>
<tr>
<td>9</td>
<td>Prime numbers</td>
<td>65</td>
<td>17 (26%)</td>
<td>1</td>
</tr>
<tr>
<td>10</td>
<td>Steady state heat equation</td>
<td>98</td>
<td>56 (57%)</td>
<td>3</td>
</tr>
</tbody>
</table>
1. Goals
   - Separate the parallel sections from the sequential parts of a program, which allows a programmer to focus more on the parallelism.
   - Define a new execution strategy for the computation intensive part of the program without changing the flow of the program.

2. PPmodel
   - Using PPmodel, the parallel part of the program can be separated from the sequential part of the program, re-designed, and then regenerated.
   - Programmers can switch between technical solution spaces (e.g., MPI, OpenMP, CUDA and OpenCL) without actually changing the program.

Modeling Parallel Programs
PPmodel is explained from a user’s perspective with the Circuit Satisfiability problem example.

1. **Model creation for Circuit Satisfiability problem**
   - Create a model from an existing program: *The parallel part of the program is extracted from the sequential part and stored separately.*
   - Create a visual representation of the model: *The model representation of the cluster is generated.*

2. **Modeling the Circuit Satisfiability problem:** *Modeling helps the programmer specify the automatically detected blocks to execute in a different platform.*

3. **Code generation:** *Code integration involves replacing the OpenMP code with the newly added MPI code, adding libraries to execute MPI code, and some code to initialize MPI-specific variables.*

---

**Working with PPmodel**
Working with PPmodel
Working with PPmodel
Implementation Details of PPmodel

Source code

Updated parallel blocks

Original parallel blocks

#pragma omp for schedule(dynamic,chunk)
for (i=0; i<N; i++)
{c[i] = a[i] + b[i];
printf("Thread %d: c[%d]= %f\n",tid,i,c[i]);}
}  /* end of parallel section */

#pragma omp for schedule(dynamic,chunk)
for (i=0; i<N; i++)
{c[i] = a[i] + b[i];
printf("Thread %d: c[%d]= %f\n",tid,i,c[i]);}
}  /* end of parallel section */

#pragma omp for schedule(dynamic,chunk)
for (i=0; i<N; i++)
{c[i] = a[i] + b[i];
printf("Thread %d: c[%d]= %f\n",tid,i,c[i]);}
}  /* end of parallel section */

#pragma omp for schedule(dynamic,chunk)
for (i=0; i<N; i++)
{c[i] = a[i] + b[i];
printf("Thread %d: c[%d]= %f\n",tid,i,c[i]);}
}  /* end of parallel section */

#pragma omp for schedule(dynamic,chunk)
for (i=0; i<N; i++)
{c[i] = a[i] + b[i];
printf("Thread %d: c[%d]= %f\n",tid,i,c[i]);}
}  /* end of parallel section */

#pragma omp for schedule(dynamic,chunk)
for (i=0; i<N; i++)
{c[i] = a[i] + b[i];
printf("Thread %d: c[%d]= %f\n",tid,i,c[i]);}
}  /* end of parallel section */
There have been a few modeling efforts in the parallel programming domain.

- **Graphical programming languages:** CODE programming language is based on a generalized dependency graph to express the computation in a unified parallel computation model without any implementation details.

  CODE is a graphical programming environment, but PPmodel is a complete modeling tool to create parallel programs from sequential or parallel programs written for another target platform. GASPARD is another visual parallel programming environment supporting task and data parallelism.
There have been a few modeling efforts in the parallel programming domain.

- **OpenMP to GPGPU**: converts OpenMP programs to CUDA code
  
  *The goal of PPmodel is to express the parallel part of a program in a way separated from the sequential part so as to allow the programmers to focus more on the parallel problem than the program as a whole.*

- **Program transformation tools**: Other related works include program transformations from sequential to parallel and abstractions in parallel programs.
Limitations and Future work

- The tool currently can model only C OpenMP programs and generate target code for the MPI library.
- PPmodel can be extended to support a GPU programming language like CUDA.
- Similar implementations can be created for other programming languages and platforms.
- The programming language determines the refactoring framework to use and the platform decides the code to be inserted or re-factored
• PPmodel is a tool that can be used to separate the parallel part from the sequential part of a program.
• Using the modeling PPmodel framework, programmers can execute the parallel blocks in a different platform without actually rewriting the program.
• The approach is independent of any platform or language and can be extended to any language.
• An OpenMP program written to solve the Circuit Satisfiability problem was redesigned to execute in multiple nodes using MPI.

Conclusion
Questions

Thanks