Seminars as exceptional vehicle to learn parallel programming

Rainer Keller
Hochschule für Technik Stuttgart, University of Applied Science, Faculty of Geomatics, Computer Science and Mathematics, Stuttgart, Germany

Introduction

Parallel processing has by now become pervasive throughout computing. Our students own devices ranging from mobile phones to laptops with multiple cores:

Even a car's engine-management system has CPUs with multiple cores to process data by various sensors. Engineering being important for employers in the Stuttgart area (with companies such as Porsche, Mercedes, Bosch and Vector Informatik), PDC topics are relevant for our graduates.

The Seminar is a integral part of the Bachelor's curriculum in the 3rd year of study, i.e. right before the Bachelor thesis. Successful completion earns three Credit-Point (CP), which equals 90h of effort.

The usual procedure to hold a Seminar is focus is to provide a topic or book and have Students research this topic. As outcome they produce a scientific report actively code with it.

Aim & Setup

The best ways to learn a new PDC paradigm is to actively code with it. The author for his Class provided two albeit simple algorithms to read only few & pairs. One of the parallel programming mechanisms described in the following table.

Students were allowed to choose using Moodle’s “Choice Activity” [1] and have them work in teams. Allowed team sizes varied according to complexity of the paradigm.

Provided to students:
- An introduction into the algorithm
- Powerpoint and \LaTeX\ Templates
- Collection of slides and papers on basics of PDC
- Input dataset for testing and benchmarking
- C-code to read the CSV dataset

Students were to run their code on an HPC [2]. The Seminar was held in 2013 and 2015 with good results.

Programming models selected

<table>
<thead>
<tr>
<th>Model</th>
<th>2013</th>
<th>2015</th>
<th>Paradigm</th>
</tr>
</thead>
<tbody>
<tr>
<td>Pthreads</td>
<td>2</td>
<td>2</td>
<td>Shared Paradigm</td>
</tr>
<tr>
<td>OpenMP</td>
<td>2</td>
<td>2</td>
<td>Memory Paradigm</td>
</tr>
<tr>
<td>Cilk++</td>
<td>2</td>
<td>2</td>
<td>Distributed memory</td>
</tr>
<tr>
<td>UPC</td>
<td>2</td>
<td>2</td>
<td>GPUs</td>
</tr>
<tr>
<td>MPI</td>
<td>3</td>
<td>3</td>
<td>New languages</td>
</tr>
<tr>
<td>CUDA</td>
<td>3</td>
<td>2</td>
<td></td>
</tr>
<tr>
<td>OpenCL</td>
<td>1</td>
<td>2</td>
<td></td>
</tr>
<tr>
<td>Go</td>
<td>3</td>
<td>2</td>
<td></td>
</tr>
<tr>
<td>Haskell</td>
<td>2</td>
<td>1</td>
<td></td>
</tr>
</tbody>
</table>

Number of students choosing a particular paradigm. Values with * represents topics proposed by students.

Student’s findings: DGEMM

Students were to parallelize the algorithms:
- 2013: Matrix-Matrix multiplication (DGEMM)
- 2015: Gauss-Seidel solver with Pivazition

Findings were summarized in a paper each and presented in class in person. The following shows noteworthy contributions:

<table>
<thead>
<tr>
<th>Model</th>
<th>2013</th>
<th>2015</th>
<th>Paradigm</th>
</tr>
</thead>
<tbody>
<tr>
<td>C-code to read the CSV dataset</td>
<td>2700K using MPI. Result in seconds.</td>
<td>1024x1024. Result in seconds.</td>
<td>600</td>
</tr>
<tr>
<td>Performance of DGEMM on various GPUs and CPUs using OpenCL. Result in seconds.</td>
<td>74</td>
<td>17</td>
<td></td>
</tr>
<tr>
<td>Performance of DGEMM on various GPUs and CPUs using OpenCL. Result in seconds.</td>
<td>200</td>
<td>4</td>
<td></td>
</tr>
<tr>
<td>Performance of DGEMM on various GPUs and CPUs using OpenCL. Result in seconds.</td>
<td>500</td>
<td>4</td>
<td></td>
</tr>
</tbody>
</table>

Student’s findings: GS solver

Gauss-Seidel solver using Pivoting solves \( Ax = b \) in two phases. The first phase “forward substitution” transforms the matrix into upper triangular form, followed by a “backward substitution”, by inserting values solves the system. This is certainly more complicated than the previous year’s algorithm. Good students dived into literature and cited that “According to (7) (sic) (here [4]), this algorithm can only be efficient, if the computer, on which the algorithm is executed, has a small latency.” (Courtesy of M. Fecker).

The team working on MPI therefore had performance issues, not getting any speedup.

Conclusion Parallel Paradigms

The programming paradigms are in general easily accessible: apart from open standards and open source implementations, there are a number of examples and tutorials available on the Internet.

The easier programming models certainly are the shared memory, annotation-based paradigms. Adding keywords to programming languages (such as Cilk++ and UPC follow). It is worth noting, that PGAS runs were completed on 1 node, only.

Difficulty increased with the library-based approaches, i.e. PThreads and MPI being the most complicated to use for these algorithms.

GPU models are arguably the most difficult perceived ones. GPU and the new programming languages received attention by the best students; solutions were well thought out. Especially the functional language Haskell was well presented, even though run-times could not compete with GPU solutions.

Conclusion Educational Concept

The didactic concept may be summarized as follows:
- The general idea of assigning an algorithm for parallelization was well received.
- Allowing to select using Moodle enabled students to trade topics, and led to “working teams”.
- Working in teams (of up to 3 students) worked very well, only few students chose to work alone.
- Prescribing \LaTeX\ was well received. The intro showed only few had prior knowledge, yet many chose it for their Bachelor thesis.
- Providing input data sets for test and benchmarks, as well as code to verify the results is important, especially for the not so strong students.
- Only the better students benchmarked and their result on the Cluster, others coped on laptops.
- Evaluation was good (averaged grade ~1.7):
  - “Practicals great, interesting topic” (multiple times)
  - “Was able to do my favorite one”
  - “Prof. is actively working with the code”
  - “Topic is very much bound to Linux”
  - “Programming models vary in complexity”
- One remarkable outcome is the lack of grades: better students were dismayed, even though this was time intensive, this course wouldn't show up in their grade score.

References


Acknowledgments

This work was supported by a grant from the Ministry for Science, Research and Art of the state of Baden-Württemberg in the frame of the bwHPC-CE project.