POSTER: Seminars as exceptional vehicle to learn parallel programming

Rainer Keller
Hochschule für Technik Stuttgart, University of Applied Science, Stuttgart, Germany
Email: rainer.keller@hft-stuttgart.de

***Abstract*—This poster presents the concept of Seminar taught in novel ways to fifth- and sixth- semester undergraduate students of CS. The contribution lies in the way, students need to combine reading, comprehending PDC concepts by programming a given algorithm. Among the parallel programming paradigms were PThreads, OpenMP, MPI, GPU-based concepts but also ones proposed by students, like Google's Go and Haskell. The experience gained over the course of two semesters shows, that the concept fits the interests of students, however, needs to be coordinated with prior lectures and other practical courses.**

***Index Terms*—OpenMP, MPI, OpenCL, Go, Haskell**

I. Introduction

Parallel and distributed computing (PDC) has by now pervaded technology and should do so in Computer Science (CS) education. Our students have devices ranging from smart watches to laptops with multiple cores. With the engineering companies in the Stuttgart area[1] this is even more important. Hence, PDC has become an important skill-set for our CS majors. In [1] the author presented the concepts of a course primarily focused on teaching specific parallel programming concepts in a traditional course format.

During two summer terms, the Seminar was remodeled to include PDC with a practical approach where students were to benchmark their implementation on a HPC cluster (details in [1]). The NSF/IEEE-TCPP Curriculum Initiative in [2] categorizes PDC content. This Seminar, due to it's practical approach includes three of four topics: architectures (as students were required to run their parallelization on a HPC cluster), applicability (in Bloom's notation) of their specific programming paradigm and at least know (in Bloom's notation) the other parallel programming topics.

II. Didactic concept

Universities of Applied Science (former German "Fachhochschule") aim for a practical approach in teaching; courses may have supervised exercises, collected homework, a mandatory "Practical Semester" where students work within a company on a relevant problem preparing a report, as well as a bachelor thesis which again is done within companies, then with scientific scope.

The curriculum includes the Seminar in the third year as a two semester periods per week (SWS), earning three credit points (CPs), which equates 90 hours of effort – without grades. The module description lists "independent scientific research", "structuring, presentation and moderation", as well as "study of current topics in CS" as goals, specifically mentioning "interaction in application environments".

Model	2013	2015	Paradigm
PThreads	1	2	Shared Memory
OpenMP	2	2	
Cilk++	2	2	
StarSs	2	2	
UPC	2	2	PGAS
MPI	3	3	Distributed Memory
CUDA	3	2	GPU
OpenCL	1[2]	2	
Go	3	2	New Languages
Haskell	–	2[2]	

TABLE I
AVAILABLE PDC PARADIGMS WITH NUMBER OF STUDENTS.

Instead of a traditional Seminar, reflecting on a specific topic from literature, this time the author offered the students an albeit simple algorithm to parallelize in one of the proposed parallel models (see Table I). Students were encouraged to program in teams – however thesis and presentation were to be done alone. For the first Seminar the author chose dense Matrix Multiplication. For this year's Seminar, a dense linear solver using Gaussian elimination with Pivoting was chosen.

About 70% of the students had prior experience with PDC in the elective course [1]. The others had additional work to comprehend, e. g. theory of parallel processing [3], [4] as well as current standards [5] and literature [6], which was provided in part as slide-deck using Moodle [7].

Students were to decide using Moodle's Choice activity on the models (see Table I). The numbers represent the successful students. Successful completion required a paper containing six pages of content, using a provided LaTeX template, 20 minute presentation with a provided Powerpoint template.

III. Conclusion regarding pallel programming

The student's results of the Matrix Multiplication and Gaussian Elimination with pivoting were very good. Most teams had run and benchmarked their code on the Cluster. Due to lack of space, deatails are left for the poster. This abstract only includes a noteworthy solution for Matrix Mulitplication as presented in Fig. 1 and the benchmark results of the PThreads for Gaussian Elimination as is shown in Fig. 2. The poster contains more results specific per programming model.

[1]Companies such as Mercedes, Porsche, Bosch and Vector Informatik

[2]This programming model was proposed by students themselves.

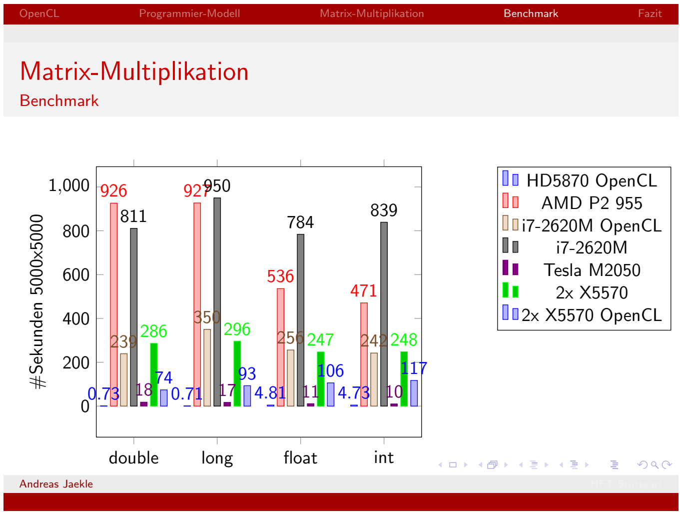

Fig. 1. Performance of various CPU and OpenCL versions (courtesy of Andreas Jaekle).

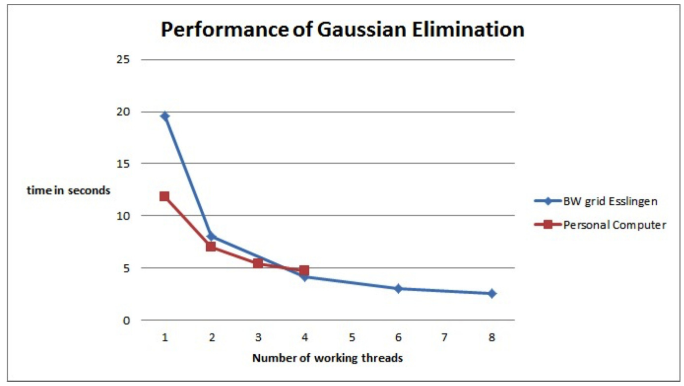

Fig. 2. Performance of PThreads (courtesy of Paul Bühler).

IV. Conclusion regarding educational concept

The didactic concept may be summarized as follows:

- The general change of assigning an algorithm for parallelization and report on the programming was well received.
- Giving students a choice and having them select using Moodle and changed was used actively – this allowed students to trade, and led to create "working teams".
- Working in teams (of up to three students as with MPI or CUDA) most of the time worked well – in few cases, students chose to work alone.
- Prescribing LaTeX was well received. The introductory course showed that only few had prior knowledge, yet many again chose it for their Bachelor thesis.
- Providing fixed input data sets for test and benchmark purposes, as well as code to verify the results is very important, especially for the not so strong students.
- Only the better students were able to finally benchmark and compare their result on the HPC Cluster – others were coping on their laptops.
- The student's evaluation of this Seminar showed very good results – the poster will contain more information and provides a list of positive and negative remarks.
- One remarkable outcome of the evaluation is the lack of grades: better students were dismayed, even though a lot of time was invested, this of course wouldn't show up in their grade score.

In the discussion part of each talk the difficulties of each group's parallel programming model was discussed. As was expected the groups working on MPI had difficulties with data distribution and communication. The groups working on GPU models had best performing implementations, but invested a lot of time. Teams working on new languages had a hard time to convert the provided C code to read the data-set into their language – as may be expected. However, their contribution was most worthwhile. Go provides goroutines and channels for communication. Haskell as a functional language has very different approach, but allows side-effect free algorithms.

V. Outlook

The author will repeat Seminars in a similar style. Areas of improvement include providing reference implementation or better base-line numbers to beat – to increase the challenge. Further algorithms may include sorting large data-sets where the benchmark input does not fit laptop memorys.

As a next topic, commonly used open source software such a GIMP in the frame of a student project would help the community and benefit the students themselves.

Acknowledgment

Foremost, I would like to thank my students for their work and valuable discussion. The author would like to acknowledge the HS Esslingen for the support of the BW-Grid cluster, specifically Prof. Väterlein. This paper is made possible by a grant from the Ministry for Science, Research and Art of the state of Baden-Württemberg in the frame of the bwHPC-C5 project.

References

[1] R. Keller, "Teaching parallel programming to undergrads with hands-on experience," in *Proceedings of Workshop on Parallel, Distributed, and High-Performance Computing in Undergraduate Curricula (EduPDHPC)*, Nov. 2013.

[2] "IEEE computer society Technical Committee on Parallel Processing (TCPP)," Internet, 2015, http://www.computer.org/portal/web/TCPP.

[3] G. M. Amdahl, "The validity of the single processor approach to achieving large scale computing capabilities," in *Proc. of the Spring Joint Computing Conference (AFIPS)*, vol. 30, 1967, pp. 483–485.

[4] J. L. Gustafson, "Reevaluating Amdahl's Law," *Comm. ACM*, vol. 31, no. 5, pp. 532–533, May 1988.

[5] Message Passing Interface Forum, *MPI: A Message-Passing Interface Standard, Version 3.0*, http://www.mpi-forum.org, Sep. 2012.

[6] B. R. Gaster, L. Howes, D. R. Kaeli, P. Mistry, and D. Schaa, *Heterogeneous Computing with OpenCL*. Morgan Kaufmann, 2011.

[7] "Moodle project website," Internet, 2015, http://www.moodle.org.