Plasma Meets Portability

A journey to performance portability and productivity in a Particle-in-Cell physics code

Joy Kitson, University of Delaware
Stephen Lien Harrell, Purdue University
Mentor: Dr. Robert Bird, CCS-7

Motivation

Exascale computing brings with it diverse machine architectures and programming approaches which challenge application developers. Applications need to perform well on a wide range of architectures while simultaneously minimizing development and maintenance overheads. Current metrics of application efficiency focus on machine-specific performance, rather than addressing performance, portability, and productivity (PPP). We create and apply a methodology for capturing the data necessary to assess productivity and use an established method for performance and portability.

Metric for Performance Portability

Performance portability (PP) can be defined as "A measurement of an application's performance efficiency for a given problem that can be executed correctly on all platforms in a given set." [1]. One metric proposed to quantify PP uses the harmonic mean, shown in Equation 1.

$$PP = \begin{cases} \dfrac{1}{\sum_{p \in H} \frac{1}{e(a,p)}} \\ 0 \end{cases}$$

Equation 1: Performance Portability

On a given set of platforms, the Performance Portability of an application is the harmonic mean of the performance efficiencies on each platform or 0 if any platform is unsupported [1]

Path to Performance Portability and Productivity

In order to maximize PP each application must be optimized to run well on each platform but be general enough to run on many platforms. A way to implement this is to have a separate code path for each platform. This method, however, increases maintenance overhead with every platform added; it forces developers to fix every bug and implement every feature separately for each platform. There are several approaches that have been developed to reduce the number of code paths, such as OpenMP, Kokkos, and RAJA.

Measure for Productivity

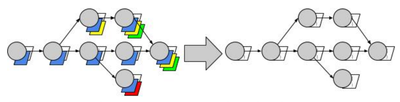

Figure 1: Logs are made during normal work, then extracted in post-processing

We create a framework to measure the productivity of development of an application. Data is collected as part of a git workflow, in which developers answer questions after each commit, directly correlating data to a snapshot of the project state. The questions track the amount of time spent on and the difficulty of specific types of tasks and include a modified NASA Task Load Index (NASA-TLX) [2] to capture user perceptions of progress for the commit. The information is combined to expose the relationships between components of a developer's work and measure the overall productivity while improving an application's PP. All variables collected can be seen in Figure 5.

Case Study: VPIC

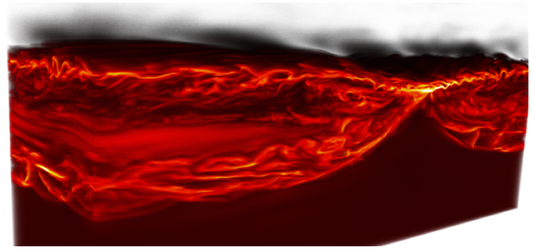

Figure 2: A VPIC rendering of volume, density and electron mix. [3]

VPIC is a particle-in-cell plasma physics model. The code tracks particles and electric and magnetic fields through a structured grid. VPIC runs at large scales and on many CPU platforms, operating with upwards of 2 million MPI ranks and 7 trillion particles [4], as well as leveraging threads. VPIC currently lacks a method to offload work to most accelerators. This is the first reported work towards porting VPIC to graphics processing units (GPUs). As a case study, the performance portability and the effort (Productivity) of porting VPIC into a PP framework was measured. For the project Kokkos [5] was chosen as the PP framework as it is the most mature with the features required. One kernel in VPIC (advance_b) was converted and analyzed in Table 1.

Platform	Original		Kokkos	
	Performance Timing (smaller is better)	Performance Efficiency	Performance Timing (smaller is better)	Performance Efficiency
IBM Power 9 OpenMP 40 Threads	0.13s	62%	0.058s	74%
NVidia V100 CUDA	N/A	0%	0.044s	98%
Intel Skylake OpenMP 44 threads	0.08s	100%	0.043s	100%
Performance Portability		0%		89%

Table 1: Performance Portability of the advance_b Kernel in VPIC

As seen in Table 1, VPIC was originally portable on 2 of the 3 target platforms. When evaluated on the original set of platforms using the harmonic mean, the PP metric is 76%. When the Nvidia V100 platform is introduced the PP of all three platforms goes to 0. It is shown above that the Kokkos port on V100 is very close to the efficiency on Intel Skylake. The performance efficiency of the IBM Power 9 is higher on the Kokkos port than the original version. It is important to note that the Kokkos implementation provides good application efficiency and it accomplishes this with one code-path which may dramatically increase productivity if the given PP framework supports new architectures in the future.

Preliminary Productivity Results

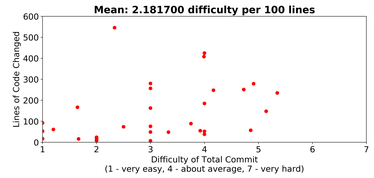

Figure 3: Lines of Code Changed vs Difficulty of Total Commit

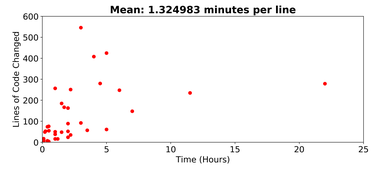

Figure 4: Lines of Code Changed vs Total Time Spent

The data used in Figures 3, 4, and 5 consists of 16 variables over 36 commits made by one author. In Figures 3, 4 each dot is representative of a single commit.

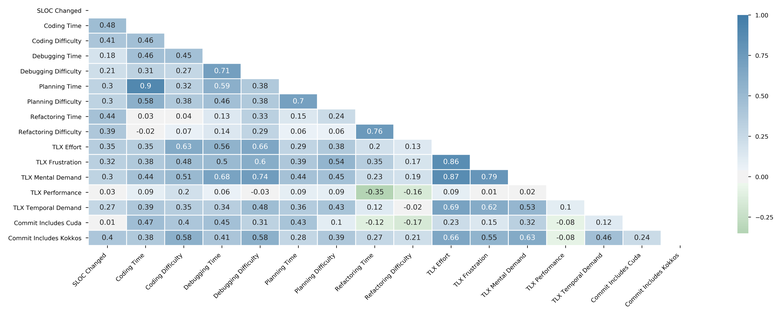

Figure 5: This figure includes all productivity variables captured on each commit. The coloring varies from dark blue (positively correlated) to white (randomly correlated) to dark green (negatively correlated)

In Figures 3, 4, and 5 it is shown that we are able to capture and analyze metrics and show correlation between some variables. In Figure 5 it is shown that Planning Time and Coding Time have a correlation of 0.9. TLX-Effort, TLX-Frustration and TLX-Mental demand are also highly correlated. This may indicate that some of these questions can be removed without impact to the overall data which would decrease the amount of time to complete the survey. In future work, the productivity measures that were created will be combined into a productivity metric that could be predictive of effort needed for PP.

Acknowledgements

Support for this work was provided by U.S. Department of Energy at Los Alamos National Laboratory supported by Contract No. DE-AC52-06NA25396. The Darwin cluster was used for this work. We would like to especially thank Dr. Hai Ah Nam, Dr. Bob Robey, and Dr. Kris Garrett for their help and advice over the course of the summer. This publication has been assigned the LANL identifier LA-UR-18-25930.

References

S. J. Pennycook, J. Sewall, and V. Lee, "A metric for performance portability," *arXiv preprint arXiv:1611.07409*, 2016.

S. G. Hart and L. E. Staveland, "Development of nasa-tlx (task load index): Results of empirical and theoretical research," in *Human Mental Workload* (P. A. Hancock and N. Meshkati, eds.), vol. 52 of *Advances in Psychology*, pp. 139 – 183, North-Holland, 1988.

A. Le, W. Daughton, O. Ohia, L.-J. Chen, Y.-H. Liu, S. Wang, W. D. Nystrom, and R. Bird, "Drift turbulence, particle transport, and anomalous dissipation at the reconnecting magnetopause," *Physics of Plasmas*, vol. 25, no. 6, p. 062103, 2018.

S. Byna, J. Chou, O. Rubel, Prabhat, H. Karimabadi, W. S. Daughter, V. Roytershteyn, E. W. Bethel, M. Howison, K. J. Hsu, K. W. Lin, A. Shoshani, A. Uselton, and K. Wu, "Parallel i/o, analysis, and visualization of a trillion particle simulation," in *High Performance Computing, Networking, Storage and Analysis (SC), 2012 International Conference for*, pp. 1–12, Nov 2012.

H. C. Edwards, C. R. Trott, and D. Sunderland, "Kokkos: Enabling manycore performance portability through polymorphic memory access patterns," *Journal of Parallel and Distributed Computing*, vol. 74, no. 12, pp. 3202–3216, 2014.