Procedia Computer Science

Volume 29, 2014, Pages 2076–2079

ICCS 2014. 14th International Conference on Computational Science

Evaluating Parallel Programming Tools to Support Code Development for Accelerators

Rebecca Hartman-Baker[1], Valerie Maxville[1] and Daniel Grimwood[1]

[1] iVEC, Perth, Western Australia.

r.hartman-baker@ivec.org, v.maxville@ivec.org, d.grimwood@ivec.org

Abstract

As the Pawsey Centre project continues, in 2013 iVEC was tasked with deciding which accelerator technology to use in the petascale supercomputer to be delivered in mid 2014. While accelerators provide impressive performance and efficiency, an important factor in this decision is the usability of the technologies. To assist in the assessment of technologies, iVEC conducted a code sprint where iVEC staff and advanced users were paired to make use of a range of tools to port their codes to two architectures. Results of the sprint indicate that certain subtasks could benefit from using the tools in the code-acceleration process; however, there will be many hurdles users will face in migrating to either of the platforms explored.

1 Introduction

In 2009 as part of the Super Science Initiative, iVEC was granted $80 million to establish the Pawsey Centre, a high-performance computing facility to support the Australian research community. iVEC has commissioned Phase 1 of the Pawsey Supercomputer, a Cray XC30 supercomputer called Magnus, in Q2 2013. In Q2-3 2014 iVEC will accept delivery of the second phase of Magnus, in which the machine will be upgraded to the petaflops level. Given budgetary and power constraints, the most cost effective way to reach the petascale is through the use of accelerator technology. The choice of accelerator technologies offered by the vendor was between NVIDIA GPUs or Intel Xeon Phi Coprocessors, both of which present challenges for migration of science codes.

The question of which of the leading accelerator technologies would be most appropriate for iVEC users needed to be answered. One important factor in the decision was the usability of the technologies. To assess the usability, we devised a Code Sprint event, in which advanced iVEC users were paired with iVEC expert programmers to port their codes to these architectures over the course of five days. The pairs used existing tools that will be available on the second phase of Magnus independent of accelerator technology, such as Cray Reveal (Cray parallel analysis and scoping tool),

 Selection and peer-review under responsibility of the Scientific Programme Committee of ICCS 2014

doi:10.1016/j.procs.2014.05.191

CrayPat (Cray performance profiling tool), and Allinea DDT (a parallel debugger), along with platform-specific NVIDIA and Intel tools to assist in porting to these platforms. The Sprint was deliberately short to represent the limited programming resources available to most academics.

Comparisons between accelerators have appeared regularly in the literature. However, these generally focus on the performance and power efficiency of a few codes, with the porting and optimization usually done by experts. At supercomputing centres that have hundreds of users with dozens of applications, we cannot take this approach. To scale to a large user base we need to push more of the porting and optimization burden back onto the user base, focusing on tools and an assumption of just a small amount of assistance from experts for each code. We could not find any publications in the literature that focus on the development aspects of accelerator choice, particularly around tools.

2 Approach

The code sprint was designed to evaluate the tools and platforms for the Magnus supercomputer, which will be used by researchers across Australia and across disciplines, codes, languages and libraries. With this in mind, we called for interest in being part of the code sprint, while targeting key user groups. This resulted in four codes being put forward in the areas of quantum physics, computational chemistry, radio astronomy and gravitational waves. This sample consists of users who were experienced developers with a solid understanding of their codes and who have the time to participate in the code sprint, so is not representative of the overall user base. We took the position that if our experienced and enthusiastic team struggled with any tools and technologies then our overall user base would certainly struggle.

It is the belief of our staff that programming supercomputers into the future will continue to be dominated by Fortran and C/C++, will be distributed across nodes using MPI, and have an OpenMP/OpenACC threading model within nodes. Although CUDA and OpenCL give excellent performance in some cases, we see no justification to force these upon a large community of users who have legacy codes. Thus we believe the user base must adopt OpenMP and/or OpenACC, which are very similar. The choice then is driven largely by usefulness of tools, ease of use of tools, any porting issues that may result from specific hardware technologies, and performance.

Prior to the code sprint we surveyed the participants to provide a self-assessment of their experience and background. All have over four years programming experience (most over ten years) and classed themselves at “Master”, or near Master level of parallel programming expertise. Half of the participants have qualifications in computing. Experience with MPI was medium, OpenMP was medium, a few were OpenCL and CUDA experts, however none had experience with Xeon Phi. Tools were not a strong area of expertise – DDT expertise was assessed as “none” or “some”, while most had “some” expertise with profiling tools. We paired an iVEC staff member with a code expert to make up each group, with one group assigned to each of the four codes. In addition, a mentor supported all of the groups contributing strong experience in tools and with the target systems used for the sprint.

Due to the prior experience of OpenMP among code sprint participants but minimal experience with OpenACC, all participants attended NVIDIA OpenACC training the week before the sprint. Participants also attended a briefing session on the way the sprint would be organised. The high level plan was:

1. Benchmark code on target machine
2. Profile using CrayPat
3. Run Cray Reveal

4. Work with Reveal-enhanced version
5. Add accelerator options
6. Optimise code
7. Document experience

Profiling and analysis were performed on the supercomputer Todi, which is a Cray XK7 courtesy of the Swiss National Supercomputing Centre, CSCS. Although it is an AMD Opteron based system, it is still suitable for testing the tools to identify and fix bottlenecks.

Each day began and ended with a short meeting about the goals for the day and any issues that had arisen. A git repository was set up for the code sprint to allow analysis of the various versions of the group codes. Guidelines were set out for when to commit to the repository and conventions for tagging versions. These were documented on the sprint wiki. A log of activities was kept, tracking time taken on reading, discussing, coding, using tools and "other", with subcategories within each of these activities. We also tracked the command history for each participant. A survey on perceptions of the different tools was taken at the end of each day to determine how their sprint experience affected their opinions of the tools.

3 Results

Since we were primarily interested in personal experiences with tools and porting difficulties, the responses are subjective. The resulting opinions of the tools and environments were consistent among the participants. We are deliberately not publishing performance results since this was not a major objective of the code sprint, and further time spent in optimization may or may not have given significant improvements. Within the one-week timeframe of the code sprint, time spent porting was time lost on optimization.

On reflection, the sprint participants felt that OpenMP within the Xeon Phi environment and tools were not mature enough, that they were cumbersome and need more revision. Participants found fewer basic issues with OpenACC on GPU, thus working with them was more straightforward. In both environments there is a need for restructuring of data, which could have a major impact throughout the code. Similar data restructuring techniques appeared necessary for both OpenMP on Xeon Phi and OpenACC on GPU. Cray Reveal was considered useful for analysis and OpenMP guidance, which is its purpose, and it did a reasonable job of suggesting OpenMP directives. The OpenMP guidance from Cray Reveal can be used to help with OpenACC porting, although this additional effort is not suitable for a large user base. An OpenACC-specific version should improve results for a general user base if undertaking a similar code porting exercise.

Difficulties encountered by the groups included: differences in compilers; runtime issues on the Xeon Phi; incompatible memory structure; remote graphical display for Reveal; Cray thread placement (CPU); excessive data movement; and inconsistent documentation. Although few found strong performance improvement, the exercise of reworking the code for accelerators improved the CPU-only performance.

4 Conclusions

The sprint was of great value to iVEC and the user base, as it allowed us to experience the tools and platforms and get a feel for what the migration process would entail. Results of the sprint

indicated that the tools could be useful for certain subtasks in the code-acceleration process. However, there will be many hurdles users will face in migrating to either of the accelerated platforms, such as differences in the way codes are compiled and run, the need to expose new levels of parallelism that the accelerator can exploit, and a fundamental shift in parallel programming paradigm from process-centric to data- or task-centric.

For the choice of architecture of Magnus, we decided that neither accelerator technology was supported by tools that could assist a significant portion of the iVEC user base with a practical level of iVEC staff support. Due to roadmap changes the availability of Intel Haswell CPUs became available within the procurement timeframe, and we elected to go with a smaller peak-performance supercomputer in an all-CPU configuration. The tools and techniques used to port to and improve performance with OpenACC and accelerator-based OpenMP can still and should be applied to all-CPU systems. Thus we can introduce the user base to these tools and incrementally improve performance while still having a fully utilized supercomputer.

5 Acknowledgements

We wish to acknowledge the contributions of the participants in the code sprint: Igor Bray (Curtin University), Shinkee Chung (UWA), Daniel Grimwood (iVEC), Christopher Harris (iVEC), Paul Ryan (CSIRO), Nicola Varini (iVEC), and Jason Wang (UWA). We are grateful for access to systems provided by CSCS (Dommic - Xeon Phi, Todi – Cray tools), The University of Tennessee (Beacon - Xeon Phi) and ORNL (Titan – NVIDIA Kepler).