Intel Fellowship: Cross-disciplinary Xeon Phi code optimization

Summary

Intel Corporation has provided funding for one Research Assistant position at the University of Florida for a project that optimizes useful research code to run better on the Xeon-Phi

Application and review process

Number of awards: 1

Duration of award: 1 year

Funds: RA salary (max. 20 hours per week) $25,000 plus tuition $10,000

Resources: Computing resources and access to Intel Xeon Phi processors will be provided by UFIT Research Computing on HiPerGator.

Earliest start date: May 1, 2016

Proposal submission deadline: March 15, 2016

Proposal requirements:

1. One-two page description of the team:
    a. Each team must consist of at least two faculty members: one with demonstrated experience in parallel programing, and one with domain expertise in any area of scholarship.
    b. Each team must have two students: one who will carry out the code optimization on the Intel Xeon Phi, and another who will bring the domain expertise and will use the software for their research.
2. Two-page description of the scholarly, science, or engineering application problem that uses the software that will be ported and optimized for the Intel Xeon Phi. Work on existing open source software is preferred so that the benefit of the code optimization goes back to the larger community beyond UF. But new open source code projects are not excluded to allow the awards committee to support innovative work.
3. One-page project plan spanning one-year with goals and projected accomplishments at least every quarter. A way to measure and demonstrate the accomplishments must be defined.
4. Two letters of recommendation for the students involved in the project.
5. Curriculum vitae of all team members. (Maximum of 4 page CVs for each individual.)

Submission requirements:

Submit proposals as a single PDF file to deumens@ufl.edu

Reporting requirements:

The student who receives the fellowship is responsible for quarterly reporting of progress on the project to UFIT Research Computing, Erik Deumens deumens@ufl.edu. The master's or PhD theses of both students will serve as the final report.

The projects need to prepare quarterly progress reports that includes:

1. partial code delivery with test cases, and
2. the student needs to provide verbal 30 minute to 1 hour status report to the mentoring faculty and the project team, which includes the faculty mentors of the students and the project manager.

Questions:

Direct any questions about the fellowship or the procedures to Erik Deumens deumens@ufl.edu.

Review process

The review process will be done by members of the UFIT Research Computing Advisory Committee, UF Informatics Institute Steering Committee and by engineers in the Intel research division.

The criteria for review are a direct consequence of the purpose for this fellowship, which are explained in detail in the next section, and can be summarized as follows:

1. Cross disciplinary work involving (a) computer science and software engineering for Xeon Phi parallel computer systems, and (b) any application domain in science, engineering, or humanities that can benefit from accelerated execution on Xeon Phi processors.
2. The software being created or enhanced, and optimized must not be proprietary, but must be made available under some form of open source license at the completion of the project.
3. Clear formulation of the problem to solve within the one-year duration of the fellowship with intermediate steps and goals described and measurement of progress defined.
4. The most valuable projects are expected to include some other domain of science, engineering, or big data research in the mathematics, statistics, social science, or humanities. However, computer engineering applications are not excluded.

Guidelines for purpose and content of proposed projects

Why code optimization?

Xeon Phi programming requires work to get proper performance enhancements compared to running on Xeon processors. With general purpose graphical processor cards (GPGPU) the programming model is different and hence code must be analyzed and parts rewritten. Once the commitment is made to make that effort, programmers usually do a great job and get the expected 10-fold performance increase of running on Intel Xeon processors. Large performance increases of 20, 50, or 100 are always due to comparison with really bad code running on the Xeon. When the GPU code, in CUDA, is ported back to the Xeon, the Intel compiler will recognize the opportunities to use vector instructions, the data flow has been optimized for the GPU and thus flows better through the cache hierarchy. All this results in

significant speed-up of the code on the Xeon processor. The result is that the performance increase from Xeon to GPU roughly matches the ratio of the theoretical peak performance between the GPU and the Xeon.

The Intel Xeon Phi architecture can be programmed using standard languages. This simplifies porting, but now programmers are not incentivized and motivated to analyze and optimize the data flow as they are when facing porting to CUDA. Similarly, loops are not analyzed and reorganized. Thus the performance of the Xeon Phi of old code is suboptimal, similar to its suboptimal performance on the Xeon.

Commercial software vendors make the code optimization investment only if the business case is clear and compelling. For academic research software the benefits are not always clear and the analysis is often not even carried out. In academic research new science and engineering results matter, not the performance optimization of existing code. Performance optimization is carried out only if it leads to new research results that are not reachable with old code. However, if the code optimization effort takes longer than running the old code in real time, the code optimization is considered to be not justified.

Complementary interests

There is a lot of existing code that contains a valuable legacy of algorithms developed for sometimes extremely complex problems. To port and optimize the performance of such codes requires two complementary sets of expertise, skill, and methodology:

1. The domain scientists, engineers, and digital humanities researchers know the algorithms and their strengths and limitations. They also know what problems are important for science and engineering and where and how approximations can and should be introduced.
2. Computer scientists and engineers know the capabilities and pitfalls of the hardware, compilers, and programming libraries. They also know the programming methodology to create, expand, and maintain complex software systems to generate reliable and reproducible results.

Team research effort

Building on the agile programming approach of pairs, or teams, of programmers to get more reliable results in software development faster, “Intel Code Optimization Fellowships” are targeted to fund the students enrolled in a computer science or electrical and computer engineering master or PhD program that are part of a cross-disciplinary project that involves at least one domain specialist student.

1. The computer science student benefits from the team effort because (s)he learns advanced programming and algorithm optimization in a real-life application instead of a textbook exercise or example.
2. The domain science, engineering, digital humanities student benefits from the team effort because (s)he will be able to carry out more extensive, more complex, more relevant computations within a reasonable time with the optimized code than would be possible with code written by herself or himself.
3. In addition the community of researchers in that domain may benefit from the development of open source software for that community that is written using better software development methodology than what is the common practice for code written by domain specialists.