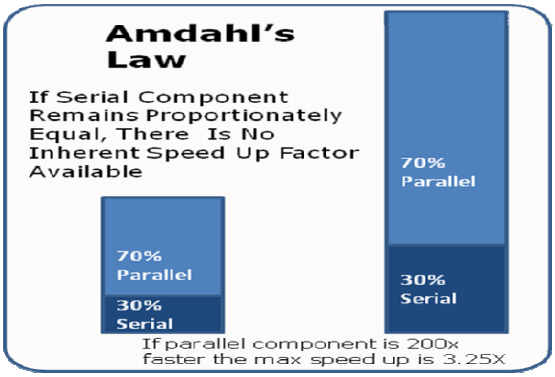
High Performance Computing, @ Intel: Do we have universal solution?

Dr Paweł Gepner – HPC Platform Architecture Specialist
Intel Corporation / Europe Middle East and Africa

Abstract:

When architecting HPC systems Seymour Cray once said performance is all about balance. The balance between processor capability, memory capacity and system bandwidth enables applications to deliver peak performance. Intel works on hardware and software technologies that will accelerate applications "total" performance and not just a part of it. When considering "total" performance it is necessary to pay attention to a number of components including: CPUs (cores), memory, interconnect, OS and others. There are many ways to accelerate performance today. The question is: do we look at total performance or is it just a fraction of the problem? It needs to be a system approach which offers users a robust computing environment capable of delivering performance for algorithms for dealing with things like intensive data movement, branching and predicting, combination of vector and scalar operations, ect. It is the combination of processing capacity, memory capacity and system bandwidth that delivers total application performance.

There is no doubt that parallelization is an important aspect of upgrading application performance but threading should not be the first step in optimizing an application. In some cases, you might gain greater performance improvements by optimizing the serial version than by creating a parallel one; this can be achieved using serial and vector optimizations. HPC hardware accelerators -- GPUs, FPGAs, the Cell processor, and custom ASICs like the ClearSpeed floating point device -- have created perception in HPC community that they provide a higher performance then general purpose CPUs. How does this look in reality when we consider broad picture of HPC code and taking in to consideration total performance and some fundamental principles e.g. Amdahl's law or Gustafson's observation?



Today's GPU's and Cells are optimized for data parallel application. While important it represents a very small percentage of the total applications. General purpose CPU will scale forward to new generations whereas GPU's/Cell will typically need coding

using proprietary extensions to standard languages, and potential recoding across new generations of architectures (for example rearchitecting is necessary to take advantage of increased memory). The GPU model today is limited to data parallel algorithms that require huge amounts of concurrency (e.g.32*240=7,680 threads). In addition it's very hard to debug such code and difficult to profile. In the Cell implementation we need to be very diligent and ensure that code fits in the memory. In addition work has to be split into 3 parts (SPE, PPC, X86) and has to map nicely to the SPE structure and memory. It is a very complex task to develop and optimized code which is not human readable at all. FPGA seems to be very good solution for the same statics application as results indicate that there is significant difference between optimized data flow and simplified application dataflow. Of course FPGAs map exactly dataflow of an application to the same exact execution dataflow but to be effective the exact dataflow must be determinate and optimized and this is not trivial task. Unfortunately for FPGA the modern CPUs provide 20x time faster clock than most FPGAs today (150-300 MHz). Taking this into consideration it looks like total performance of applications utilizing FPGA does not seem so great.

Today we have a multiple paths to increase performance which one is the best or is any? This is the same problem that the Intel Corporation strategy and positioning for HPC is addressing leading to a way to deliver scalable performance. This is going to be discussed during this talk.