21st International Conference on Computing in High Energy and Nuclear Physics **CHEP2015** Okinawa Japan: April 13 - 17, 2015

Contribution ID: **243** Type: **poster presentation**

ROOT/RooFIT optimizations for Intel Xeon Phi's - first results

In the past, grid worker nodes struck a reasonable balance between the number of cores, the amount of available memory, available diskspace and the maximum network bandwidth. This led to an operating model where worker nodes were "carved up" into single core job slots, each of which would execute a HEP workload job. Typical worker nodes would have up to 16 computing cores , with roughly 2 GB RAM and 32 GB scratch space per job slot.
In the last few years the number of threads and cores per processor has risen rapidly. With the advent of multi-core, many-core and GPU computing the number of cores is expected to rise even more. This leads to an imbalance on modern worker nodes, where the "job per core" model is no longer a valid option.
In view of these development the desire, or perhaps even the necessity, to run programs in parallel has also increased. However, the process of transforming the HEP software to run in parallel has been slow and painful.

One of the more recent development in the GPU and many-core computing field has been the introduction of the Intel Many Integrated Core architecture, known now as the Intel Xeon Phi. The Xeon Phi is a PCI-Express adapter with up to 16 GB of RAM on board and with up to 61 cores, each capable of running 4 threads. The Xeon Phi has a theoretical performance of more than 1 TFLOPS.

Initial attempts to port HEP code to the Xeon Phi has not produced the expected results and often a performance *decrease* has been seen, instead of an increase. This has also been reported by non-HEP researchers, and slowly the reasons for this are becoming known. Using the proper optimization techniques it is possible to achieve 1 TFLOPS, but this requires substantial effort in rewriting and optimizing the application. The main bottleneck here is that the Xeon Phi should not be regarded "61 independent cores" but instead it should be seen as a vector instruction machine. By properly vectorizing an application the performance can improve significantly. This also applies when porting an application to GPUs.

In this talk we will focus on porting and optimizing ROOT, and in particular RooFIT to the Intel Xeon Phi. A zero-order port does indeed show a decrease in performance, but by analyzing and optimizing the 'hot spots' in the code the performance can be improved. It is vital to validate the results of the ported application, however: the Xeon Phi version of RooFIT must produce the same results as the single-core version.

Most interesting is the fact that code optimized for the Xeon Phi also performs better on "regular" Xeon processors. As part of the results we will also discuss the performance increase of the optimized code on an Xeon E5 processor. In that way the Xeon Phi porting effort can be seen as a catalyst to transform the HEP code to run in parallel on modern CPUs.

The main goal of this research and of this talk is to provide a direction for optimizing the HEP software for modern and future computing platforms. The

main goal of parallelizing RooFIT is not to make it scale to 60 cores or more, but to find a way to restore the imbalance seen in modern worker nodes. If a relatively simple code optimization leads to HEP software that can scale to roughly 10 cores then the "core imbalance" can be restored. Worker nodes could then be carved up into "10 core job slots". This is a different approach from true High Performance Computing centers, who are usually interested in getting an application to scale beyond 32 cores and more.

Primary author: KEIJSER, Jan Justinus (NIKHEF)

Presenter: KEIJSER, Jan Justinus (NIKHEF)

Track Classification: Track8: Performance increase and optimization exploiting hardware features