

# BOF Report: Early Experiences Developing and Debugging on the Intel Xeon Phi Coprocessor

---

This BOF highlighted user experiences debugging on systems which incorporate the Intel® Xeon Phi™. We started with 5 talks summarizing early experiences from 5 different sites.

Vince Betro, UtK/NICS presented "*Debugging on Beacon: Porting Codes and Finding Bugs on the Intel Xeon Phi*". He talked about their experiences porting a range of 7 important scientific codes to the Xeon Phi. He showed a series of different optimizations and how much they improved application performance. He also showed what level of thread parallelism is required to get performance and discussed issues that they found debugging these applications.

Simon Hammond from Sandia National Lab presented "*The Path to Exascale and Why We Will Need Scalable Debuggers*". He talked about their strategy of exploring the architecture using mini apps on a test bed Xeon Phi system. He highlighted the importance of OpenMP and vectorization, as well as affinity and careful control of environment to performance. Debugging techniques and tools are critically importance. Printf style debugging is inadequate due to the overall scale and importance of multi-threading.

Dave Hiatt from Citigroup presented "*Debugging C++ Memory Consumption Issues on the Xeon Phi*". He discussed their experiences working with a Xeon Phi port of a Monte Carlo cash flow engine. While they were able to port it with a recompile they encountered stability issues on the Xeon Phi which they tracked down to the memory allocator. He showed how he was able to use the MemoryScape memory analysis tool to explore what was happening in the heap on the Xeon Phi.

Olli-Pekka Lehto from CSC in Finland presented "*Using Totalview via SLURM on a Xeon Phi cluster*". He reviewed some of the issues that they found in enabling interactive debugging on their cluster. They have been able to enable debugging for offload, native, and symmetric mode applications through a combination of carefully configured vendor tools and locally developed scripts.

Youngsung Kim from NCAR presented "*NCAR Experiences Optimizing for Xeon and Xeon Phi*". He gave a detailed discussion of their experiences optimizing the 3d Discontinuous Galerkin kernel from the HOMME global shallow water model for the

Xeon Phi. They performed a series of 4 optimizations and highlighted how each one improved performance at different scales.

The talks provided a range of perspectives and generated a good discussion with the audience, particularly about the memory allocation and optimization techniques.

We had approximately 75 people in total at the BOF. We distributed a 6 question survey and got 28 responses. These are too many to present in full here, so I've tried to highlight interesting responses.

### **How much experience do you have developing and debugging for the Xeon Phi?**

Participants had anywhere from < 40 hours to 3 years.

### **How would you rate the experience developing and debugging for the Xeon Phi? Why did you give it this rating?**

“Hard to get speedup compared to Xeon”, “Easy but low initial performance and hard to improve it compared to Xeon”, “Better than GPU. Initial set up was easy, but MKL/OpenMP has been buggy”.

### **What was the biggest surprise you encountered developing and debugging for the Xeon Phi? What did you learn?**

“How much effect placement has on performance”, “Necessity of and process of propagating environment settings to the Xeon Phi - for OpenMP affinity settings or settings required for Vtune Amplifier XE.”, “To get good performance in the hotspot kernels, we have to consider major structural changes to calling methods to enable more vectorization and threading”, “running all cores may degrade performance”, “the benefits of SIMD on the performance. I learned to avoid local IO”

### **What could be done to improve the experience of developing and debugging on the Xeon Phi?**

“Have a toolchain / profiler for to analyze offload with memory movement”, “Make env setting propagation easier”, “Put a graphical debugger on stampede”, “improved debugging tools. More realistic examples; too many toy problems in the literature.”

### **What piece of Advice do you give to other people thinking of developing and/or debugging on the Xeon Phi?**

“Start with native mode”, “get the code parallelized and multithreaded correctly on Xeon first, with a decent numbers of threads, then the port is straightforward to the Xeon Phi”, “most applications are not adapted to the phi (only to classical CPUs).

Only embarrassingly parallel (GPU friendly) applications will benefit from MIC. More MPI/OMP overhead with 120/260 threads/processes than with 8/16.”

**What was the best thing about your experience of developing and/or debugging on the Xeon Phi? The worst?**

“Best: Port of the Xeon Phi was basically only a recompile. Worst: Lots of different configurations”, “Best: easy compilation, optimizations benefit host code as well. Worst: OpenMP/MKL bugs”, “Best: I got 6x speedup for one app one MIC comparing with the original code and 4x for Sandybridge (16 cores). Worst: Memory limitations per device”, “Best: it motivated me to understand processors better, and now I can also program standard Xeon processors more effectively. Worst: Managing expectations (mine and others)”

The slides presented to accompany the 5 presentations are available at

<https://www.dropbox.com/sh/qbazmwoo95em9td/OtRPRbCbVK>

Transcriptions of all the survey responses are also available.

<https://www.dropbox.com/s/vetvzot2tdrk3rk/SC-Xeon-phi-bof-survey-transcribed.pdf>