Quick Start Guide

Incorporation of MICs in Application Execution

Goal - The compute nodes of Stampede incorporate 2 Sandy Bridge CPUs chips + (1 or 2) Xeon Phi chips. While the instruction sets of the CPU and Phi chips are both X86, the execution characteristics are quite different. The potential execution speed of the 61 cores of the Phi is about that of the two CPU chips. Additional the compilers and libraries provided by Intel make it straightforward to compile and execution applications on the Phi with little or no code modification. Thus, there is the potential for at least doubling the execution speed of an application by utilizing both the CPU and Phi chips. However, the Phis will perform well only if the application both vectorizes well and scales to many threads per chip. Additionally, the cost of data movement between the CPUs and Phi must be factored into the potential performance gain. The question is: *Can use of Phis alone or in combination with CPU outperform CPU-only execution?* This QuickStart Guide gives a simple way to determine the answer to this question.

The Phi can be used in three different ways: Native Mode – execute application solely on MIC chip, Off-Load Mode – split application code segments across CPUs and MICs and Symmetric Mode – execute entire application on both CPUs and MICs splitting data across the chips. Therefore the question becomes: *Which, if any, of these execution modes will outperformance execution on CPUs and which will perform best?*

Process – This QuickStart Guide assumes familiarity with Intel compilers and OpenMP directives. The steps in the process are:

1. **Optimize application on CPUs** Use PerfExpert with MACPO paying particular attention to vectorization and scaling and finding the optimal number of tasks/threads per node. Use of PerfExpert is very simple and requires no modification to your code. The link to the QuickStart Guide for PerfExpert [URL]. This step will be beneficial under any and all circumstances and should be done regardless of whether the application will use the Phis. It is also well established that an application which is well optimized for the SBs is usually well optimized for the Phis. Profile the execution of the optimized code to determine the execution time of each important function at the optimal scale. See Figure 1 at the end of this document for a flow chart of the rest of the process.
2. **Compile Application for Phi and Execute**. Add OpenMP pragmas for each loop nest in each important function and compile for the Phi. An example of a simple application with appropriate OpenMP pragmas is at [URL]. (More complete information on compiling and annotating applications to execute on Phis is given in the Quick Start Guide for Native Mode Execution. [URL]) Run optimized application in native mode on MICs for 60, 120 and 240 threads. Profile application using PerfExpert to get scalability of each function/procedure.
3. **Analyze Native Mode Executions.** If any of the runs in step 2 give significantly better performance than the optimized application on the CPU then choose native mode execution (or symmetric mode – see Step 4) and re-optimize application again on MICs. (New PerfExpert will be available on MICs in Fall, 2013.)

4. Analyze for Symmetric Mode. If none of the runs with different numbers of threads in step 2 are significantly faster than the CPU but at least one is about the same speed or faster, consider symmetric mode execution. Symmetric mode is implemented by splitting data across CPUs and MICs by creating MPI tasks and assigning subsets of MPI tasks to CPUs and Phis. Follow the simple example of an application prepared for symmetric mode execution at [URL] Symmetric mode is beneficial if the speedup from additional parallelism exceeds the cost of moving data between CPU and Phi memory. Follow the example at [URL] to estimate the gain from parallelism and the cost of data movement. (More complete information on compiling and annotating applications to execute on Phis is given in the Quick Start Guide for Symmetric Mode Execution. [URL]) If the cost of data movement is less than the gain from additional parallelism, then use symmetric mode.

5. Analyze for OffLoad Mode. If none of runs in step 2 are faster than execution on the CPUs, then analyze the scalability by procedure/function to see if there are functions with significant execution time which execute significantly faster on the MICs than on the CPUs. If so, evaluate off-load mode. Determine the time saved in execution of these functions on the MIC over the CPU by multiplying the ratios of execution speeds to the execution time on the CPU. Then follow the example code at [URL] to determine the cost of data movement. (More complete information on compiling and annotating applications to execute on Phis is given in the Quick Start Guide for Offload Mode Execution. [URL]) If the time saved in execution time exceeds the cost of data movement, then consider using offload mode.

6. References and Additional Information.

a) A complete example with results for each step can be found at [URL].

b) Every person who wants to consider use of MIC chips should have a copy of **Intel Xeon Phi Coprocessor High Performance Programming** by James Jeffers and James Reinders (Mar 1, 2013)

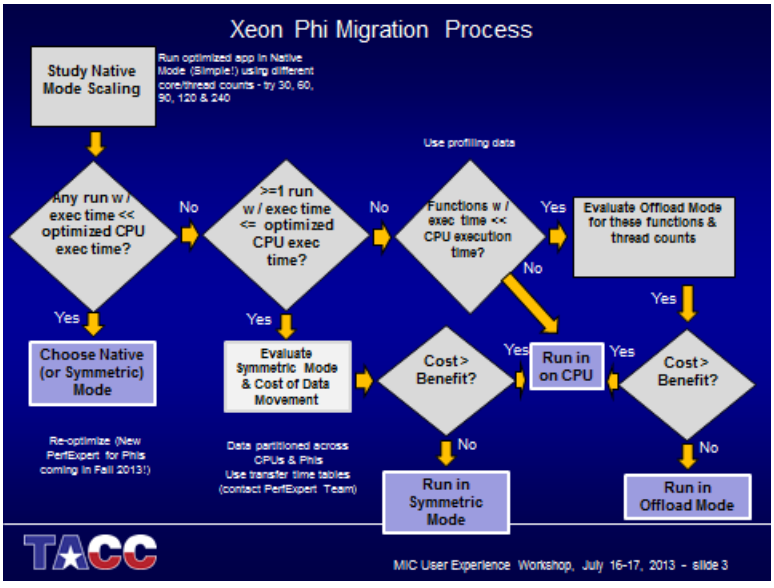

Figure 1 – Flow Chart for Steps 2 through 6.