CPU, GPU and MIC performance: a comparison of modern reconstruction hardware

Eric A. Borisch[1], Paul T. Weavers[1], and Stephen J. Riederer[1]
[1] *Mayo Clinic, Rochester, MN, United States*

BACKGROUND: Many recently developed reconstruction techniques require high computational throughput (processing workload completed per processing time) due to either large overall processing workload, such as iterative reconstructions used in many compressed sensing applications, or short permissible processing time, such as real-time or "patient on-the-table" applications requiring execution to complete within a preset time window. While modern host CPUs (Central Processing Units) have continued to grow in core and hardware thread counts, their single-thread performance growth rate year over year (tightly coupled within an architecture to clock rate) has reduced significantly. Faced with this constraint, developers of reconstruction systems (comprising hardware and software) seeking higher computational throughput must use more of the resources which are continuing to grow (core count and hardware thread count) with processor generations, and may augment the CPU's resources with an accelerator. Popular for its prevalence and low entry cost, GPUs (Graphical Processing Units) used in a GPGPU (General Purpose GPU) fashion are frequently explored for reconstruction time reductions. GPGPUs can provide a cost-effective alternative for workloads that can be effectively parallelized, which many image processing tasks are by their nature. Another type of accelerator that has recently become available from Intel is referred to as a MIC (Many Integrated Core) coprocessor, currently available as the Xeon Phi family of PCIe (Peripheral Component Interconnect Express) cards.

The MIC is fundamentally different from the GPUs available from nVidia and AMD/ATI. GPGPUs use their own discrete instruction sets, and must be programmed for explicitly (typically through either CUDA or OpenCL) to be used efficiently. The MIC, which is based on the x86 instruction set, attempts to retain much more source-level compatibility with the application it is being used to accelerate. In its current implementation, the MIC exists as a computer-within-a-computer, running Linux within the coprocessor card on a 60 core, 240 hardware thread processor. In addition, Intel markets the programmer time used to tune software performance for the MIC architecture as providing a "dual-tuning benefit" [1] by also improving the software's performance on modern multi-core CPUs. The MIC supports MPI, OpenCL, OpenMP, and Pthreads designs, and can operate in concert with or independently of the host CPUs.

PURPOSE: This work investigates the process of accelerating the reconstruction performance of a processing time limited reconstruction. The reconstruction considered is used for investigating the noise enhancement penalties over a range of potential accelerations for 3D accelerated MRA acquisitions [2] given patient-specific coil placement and body habitus, enabling patient-specific acceleration prescription during the examination process without delaying the workflow. The reconstruction performance enhancement process began with distributed (cluster) computing, progressed through both GPGPU and MIC-based implementations, and concluded with evaluating the now highly tuned software again on modern CPUs. As the cluster and GPGPU implementation steps have been described previously [3], focus will be primarily on the MIC and (modern) CPU development process, as well a discussion of the resulting performance and implications.

METHODS: The primary calculation under consideration is that of g-factor penalty, as defined in Pruessmann [4], $g_p = \sqrt{[(S^H\Psi^{-1}S)^{-1}]_{p,p}(S^H\Psi^{-1}S)_{p,p}}$, where p is the number of aliases; S is complex with a size of $n_{coils} \times p$; and the inverse of the noise correlation matrix, Ψ^{-1}, if used, is complex and $n_{coils} \times n_{coils}$ in size. For each of the many potential accelerations considered, this calculation is evaluated for every set of aliasing voxels (with masked out voxels excluded) within the volume. This leads to hundreds of millions of small (edge sizes typically less than 16) matrix-matrix calculations and matrix inversions that need to be calculated, where each individual set of matrix calculations is independent and thus highly parallelizable. The input data for the calculation (the sensitivity profile, which provides S) is transferred at the beginning of the computing process. The additional memory transfer overhead imposed with each iteration is minimal, as only the new acceleration to be interrogated and the resulting single value penalty are communicated for each cycle.

RESULTS AND DISCUSSION: With the implementation of this calculation on the Intel MIC, we can consider every possible CAIPIRINHA [5] acceleration pattern for a 420×336×132, 16-channel volume in less than 5 seconds, averaging less than 50ms per pattern. Similar performance over a range of SENSE [4] acceleration factors are shown in Figure 1. The MIC-based implementation of this calculation outperforms the OpenCL-based GPGPU (nVidia C2050) as well as the modern CPU-based (dual Intel E5-2670) version. Additionally, the MIC implementation uses ~300 lines of custom (not shared with other implementations) code, while the OpenCL version uses ~1000 algorithm-specific lines of code, as well as an additional ~1500 lines of code that comprises our OpenCL management interface (which was developed for this activity but is reusable for further OpenCL work.) This is worth consideration when comparing the performance results, as these values provide insight into the time required to move an algorithm onto a given accelerator technology. Also of interest is the performance of the modern CPU system: it is nearly that of the MIC implementation, while both implementations are faster than the GPU version. The CPU implementation has benefited (as predicted by Intel) from the tuning / algorithm modifications performed to improve performance on the MIC.

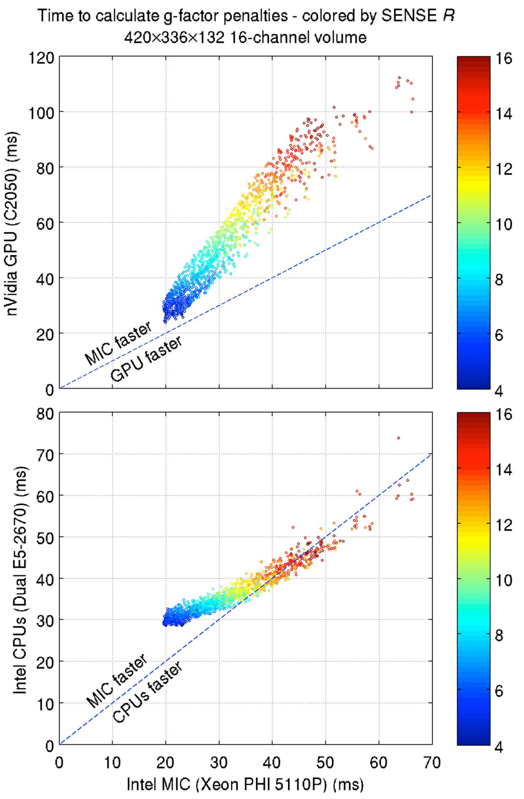

Figure 1: Comparison of time to calculate g-factor penalty volume for various SENSE R (acceleration) factors for GPU, CPU, and MIC implementations. Each point is one acceleration case.

CONCLUSIONS: The process of improving the performance of this algorithm has provided several interesting results. The initial GPGPU-powered OpenCL implementation performed well. Unfortunately, nVidia has stopped supporting new OpenCL features on their latest generation (K20) processors, requiring a re-write (OpenCL to CUDA) to fully exploit their performance. With Intel deploying its MIC technology, we chose to investigate the offerings of this new platform. Following a less intensive development process, we have achieved even better (compared to the C2050) performance on the MIC. As an added benefit, many of the optimizations performed for the MIC have benefitted all of our x86-based systems. In addition, the latest Xeon CPUs, when effectively threaded and paired with AVX vector instructions, are extremely capable in their own right. If a reconstruction system being developed is not replicated — deployed in multiple instances — the higher cost per throughput of more capable traditional CPUs may be smaller than the higher cost of development incurred by re-implementing existing software for an accelerator.

REFERENCES: [1] http://download.intel.com/newsroom/kits/xeon/phi/pdfs/overview-programming-intel-xeon-intel-xeon-phi-coprocessors.pdf [2] Weavers, MRM, DOI: 10.1002/mrm.24700 [3] Borisch, ISMRM 2013, #2640 [4] Pruessmann, MRM, 42:952 (1999) [5] Breuer, MRM, 53:684 (2005)