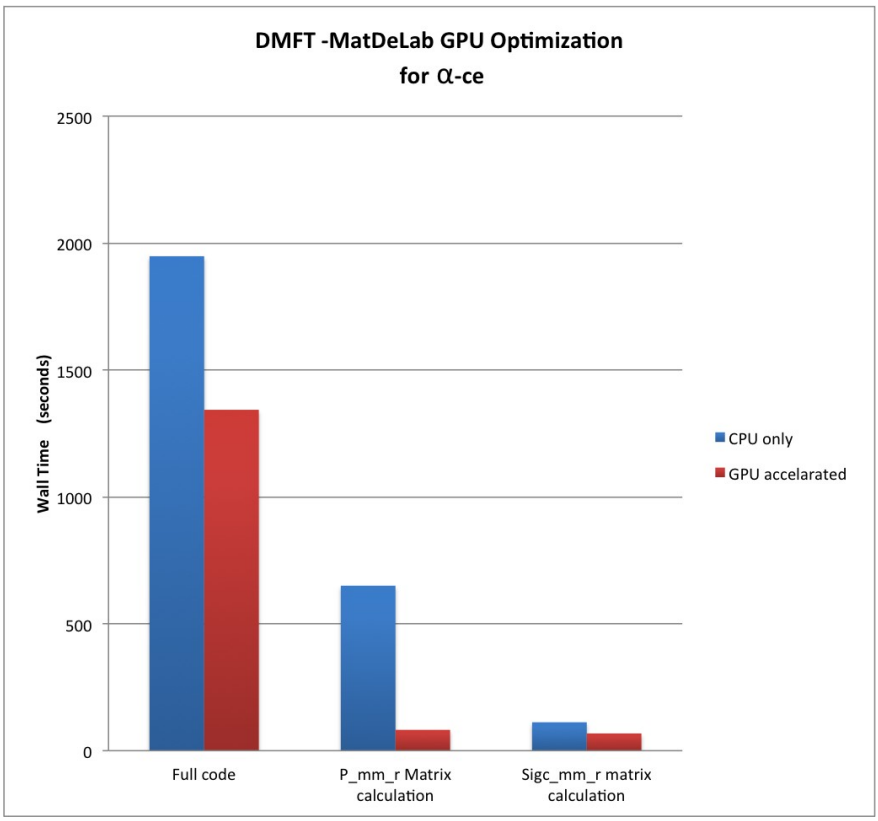
DMFT-MatDeLab Code Optimization on Titan

Technical Note

S. Choi, N. D'Imperio, Z. Dong, K. Yu

The Computational Science Laboratory at BNL has begun implementing GPU optimizations in the DMFT-MatDeLab code for running on the Leadership Class Facility computer, Titan, at Oak Ridge National Laboratory. Initial GPU coding has been implemented on two functions, p_mm_r for the polarlization matrix and sigc_mm_r for the self-energy matrix, and a brief timing analysis was done on Titan. Results show a significant speed up in p_mm_r and a moderate performance improvement in sigc_mm_r.

The DMFT-MatDeLab code was profiled using the "α-ce" job example with results showing significant hot spots in two functions, p_mm_r and sigc_mm_r. Runtime percentages were measured at approximately 33.4% and 5.8% respectively. These two functions were then targeted for optimization using GPGPUs. Both functions contain deeply nested loops that build index tables for matrix operations with a subsequent matrix multiplication operation. The matrix multiplications were originally implemented manually or by using the BLAS zGemm function. Two approaches were attempted. The first involved replacing all matrix multiplications with the corresponding Cuda BLAS versions while the second involved a full GPU port of the functions. The full GPU port was not entirely successful due to insufficient memory capacity on the Nvidia K-20X GPUs available on Titan.



Test Runs were conducted on Titan using 48 nodes, each with 16 MPI processes. Measurements were made of total wall clock running time. Total running time of the full code using the CPU-only code was 1948 seconds versus 1343 seconds for the GPU-enabled code for a 31% performance improvement. Though this increase seems relatively modest, the improvements are much more significant when examined by individual function. The p_mm_r function running time was decreased from 650 seconds for CPU-only code to 82 seconds for GPU-enabled code resulting in a speedup factor of 7.9x. The sigc_mm_r did not show as dramatic an increase in performance with CPU-only running time dropping from 112 seconds to a GPU optimized running time of 68 seconds for a speedup factor 1.65x.

Further analysis is underway to identify other hotspots in the code which can be ported or optimized for GPGPU. Also, study is beginning on algorithmic changes that may lead to greater performance enhancement.