GPU Acceleration for the Impurity Solver in GW+DMFT Packages

Kwangmin Yu[1,a], Patrick Sémon[1], Nicholas D'Imperio[1,2]

1. Computational Science Lab, Brookhaven National Laboratory
2. Center for Design of Functional Materials and Theoretical Spectroscopy

a) kyu@bnl.gov

Center for Design of Functional Materials and Theoretical Spectroscopy

ABSTRACT

The combination of dynamical mean field theory (DMFT) and GW (or density functional theory) has become a powerful tool to study and predict properties of real materials with strongly correlated electrons, such as high temperature superconductors. At the core of this combined theory lies the solution of a quantum impurity model, and continuous-time quantum Monte Carlo (CT-QMC) has proven an indispensable algorithm in this respect. However, depending on the material, this algorithm is computationally very expensive, and enhancements are crucial for bringing new materials within reach of GW+DMFT. Based on a CPU implementation, GPU acceleration is added and two times speedup is achieved. New techniques are invented and implemented to deal with various GPU acceleration environment.

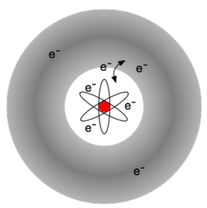
GW+Dynamical Mean Field Theory (DMFT)

- **Goal**: predict properties of strongly correlated materials
  - → **exponentially** hard problem in general
- Often only **local** correlations important
  - → capture correlations by Impurity Model

- Impurity Model simpler to solve, but not simple !

(CONTINUOUS-TIME) QUANTUM MONTE CARLO

- Solve



$$\langle O \rangle = \sum_c O(c)p(c) \quad \text{with} \quad \sum_c p(c) = 1$$

$$c := (\alpha_1, \alpha_2, \ldots, \alpha_{2k}) \qquad \mathrm{Tr}[\mathbf{F}_{\alpha_1}\mathbf{F}_{\alpha_2}\cdots\mathbf{F}_{\alpha_{2k}}] \times \mathrm{Det}\mathbf{M}(c)$$

$$\alpha_i \in A \qquad \mathbf{F}_{\alpha_i} \in \mathbb{R}^{M\times N} \qquad 1 \le N, M \le 300$$

- Monte Carlo sampling (to many c to sum over)
  - → Draw c with probability $p(c)$
  - Repeat M times

$$\langle O \rangle \approx \frac{1}{M}\sum_{i=1}^{M} O(c_i)$$

METROPOLIS-HASTING (M-H) ALGORITHM

- Draw $c = (\alpha_1, \alpha_2, \cdots, \alpha_{2k})$ with $p(c)$?
  - → $c_1 \to c_2 \to \cdots \to c_i \to c_{i+1} \to \cdots$
- One step $c_i \to c_{i+1}$?

Lets suppose $c_i = (\alpha_1, \alpha_2, \alpha_3, \alpha_4, \alpha_5, \alpha_6, \alpha_7, \alpha_8)$

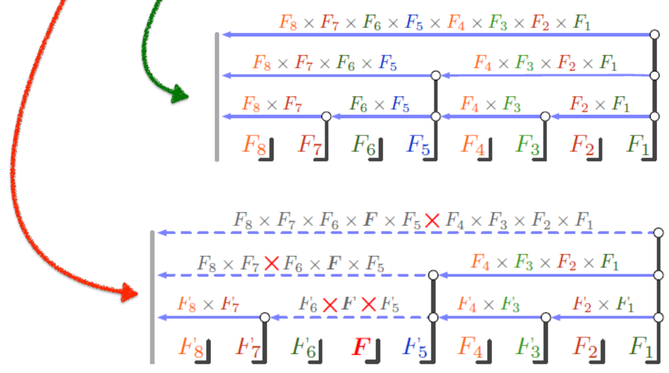
→ I) Propose trial $c' = (\alpha_1, \alpha_2, \alpha_3, \alpha, \alpha_4, \alpha_5, \alpha_6, \alpha', \alpha_7, \alpha_8)$

II) Set $c_{i+1} := \begin{cases} c' & \text{with } P = p(c')/p(c_i) \\ c_i & \text{otherwise} \end{cases}$

CPU IMPLEMENTATION

- MPI parallelized with one Monte-Carlo simulation per rank
- Skip List (or binary tree) to store sub-products

$$r := \frac{\mathrm{Tr}[\mathbf{F}_{\alpha_1}\cdots\mathbf{F}_{\alpha}\cdots\mathbf{F}_{\alpha'}\cdots\mathbf{F}_{\alpha_{2k}}]}{\mathrm{Tr}[\mathbf{F}_{\alpha_1}\cdots\mathbf{F}_{\alpha_{2k}}]} \times \frac{\mathrm{Det}\mathbf{M}(c')}{\mathrm{Det}\mathbf{M}(c_i)}$$



- Fast rejection for Metropolis-Hasting algorithm

I) Find cheap bound $B > P := p(c')/p(c_i)$ using matrix-norm
Draw $u \in [0, 1)$ uniformly

II) Reject c' if $u > B$ without calculating $P := p(c')/p(c_i)$

CODE ANALYSIS

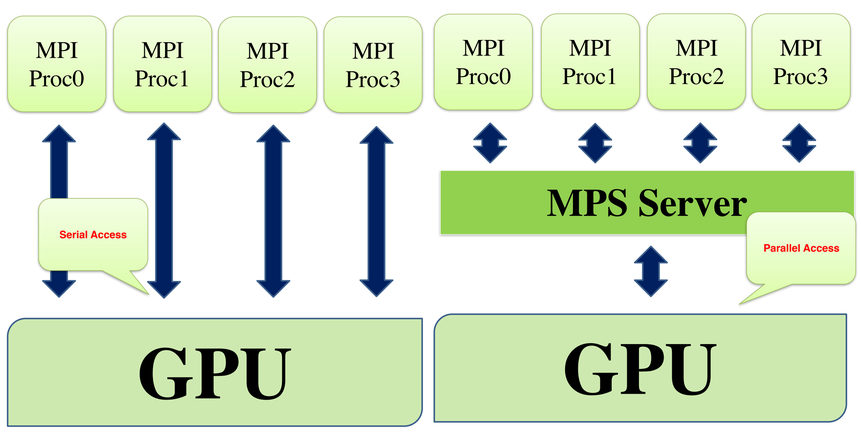
Profiling for Plutonium based compound

	Matrix Mult.	Norm	M-H (Matrix related)
Time (sec)	47577	8062	57707
Ratio (%)	81.8	13.9	**99.2**

$\mathrm{Tr}[\mathbf{F}_{\alpha_1}\mathbf{F}_{\alpha_2}\cdots\mathbf{F}_{\alpha_{2k}}] \times \mathrm{Det}\mathbf{M}(c)$ $\quad$ $B > P$

GPU ACCELERATION (I)

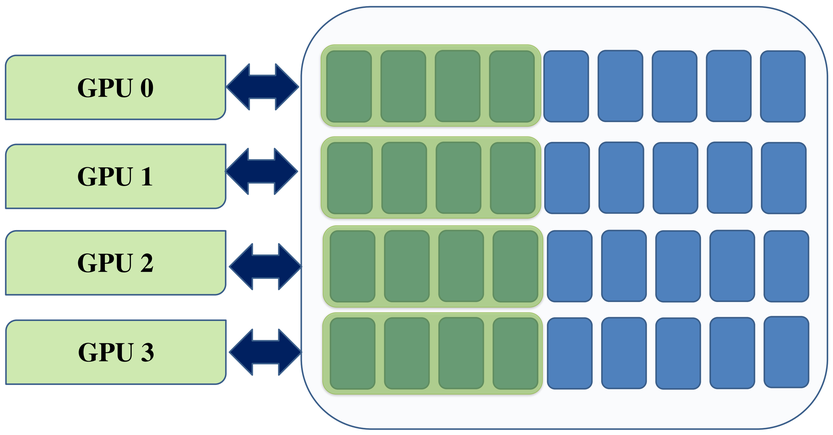
- $\mathbf{F}$-matrix related operations ported to GPU
  - → Matrix-Matrix multiplication : cuBLAS
  - → Frobenius norm : CUDA Kernel
- GPU underutilized when running one Monte-Carlo simulation
  - → **CUDA MPS (Multi Process Service)**



GPU ACCELERATION (II)

Concurrent Run of CPU & GPU

- ✓ Some part of MPI process in a node use (or share) one GPU device.
- ✓ Even though all MPI processes in a node share one GPU device, it would not be optimal.
- ✓ Rather than all MPI processes try to share the GPU resource, only some MPI processes utilize the GPU resource and others run on CPU.
- ✓ Since MPI processes accelerated by GPU would be faster and each MPI process has its own Markov-Chain, the accelerated processes take more Markov-Chain steps.
- ✓ Dynamical load balancing is possible and implemented.



RESULTS

HPC1 Cluster at BNL

- ✓ Intel(R) Xeon(R) CPU E5-2670 0 @ 2.60GHz (16 cores)
- ✓ GPU: one NVIDIA K40
- ✓ Plutonium (f-shell) with total 144k thermalization, 2880k measurement

MODE	CPU	GPU	GPU+CPU (8/16)
Running Time (sec)	4775.4	N/A	2071.1
Speedup over CPU	1	**N/A**	**2.31**

Institutional Cluster at BNL

- ✓ Intel(R) Xeon(R) CPU E5-2695 v4 @ 2.10GHz (36 cores)
- ✓ GPU: Two NVIDIA K80
- ✓ Plutonium (f-shell) with total 144k thermalization, 2880k measurement

MODE	CPU	GPU	GPU+CPU (6/9)
Running Time (sec)	1726.5	889.8	825.8
Speedup over CPU	1	**1.94**	**2.09**

Titan at ORNL

- ✓ AMD Opteron 6274 @ 2.20GHz (16 cores)
- ✓ GPU: one NVIDIA K20X
- ✓ Plutonium (f-shell) with total 144k thermalization, 2880k measurement

MODE	CPU	GPU	GPU+CPU (4/16)
Running Time (sec)	3660.3	N/A	1594.6
Speedup over CPU	1	N/A	**2.30**