Accelerating quantum Monte Carlo

Presented at

Electronic Structure Calculation Methods on Accelerators

Jeongnim Kim

R&D Staff

Materials Science and Technology Division

Computational Chemistry and Materials Division

Oak Ridge, TN

February 7, 2012

U.S. DEPARTMENT OF ENERGY

OAK RIDGE NATIONAL LABORATORY

MANAGED BY UT-BATTELLE FOR THE DEPARTMENT OF ENERGY

Acknowledgements

QMCPACK developers

- **Kenneth P. Esler (Stoneridge)**
- Jeremy McMinis (UI)
- Miguel Morales (LLNL)
- Bryan Clark (Princeton)
- Luke Shulenburger (Sandia)
- Jaron Krogel (UI)
- John Gergely (NY TU)
- Simone Chiesa (UT)
- Kris Delaney (UCSB)

http://qmcpack.cmscc.org

QMC Endstation

- David M Ceperley (UI)
- S. Zhang & H. Krakauer (W&M)
- P. Kent (ORNL)
- L. Mitas (NCSU)
- Umrigar & Hennig (Corrnell)
- A. Srinivasan (FSU)

Special thanks to

- Richard M. Martin (UI)
- T. C. Schulthess (ORNL, CSCS)
- John W. Wilkins (OSU)

Outline

- Introduction to quantum Monte Carlo
- Accelerating QMC
- Accelerating QMC on GPUs
- Conclusions

Solutions for Electronic Structure

Variational principle:

$$E_T = \frac{\int d^{3N}\mathbf{R}\ \Psi_T^*(\mathbf{R})\hat{H}\Psi_T(\mathbf{R})}{\int d^{3N}\mathbf{R}\ |\Psi_T(\mathbf{R})|^2}, \quad E_T \geq E_0$$

- Make an intelligent guess for the basis set for Ψ_T

  HF, MP2, CCSD(T) … Full CI (QC)

- Map *N*-body problem to *N* 1-body problem and use effective $H[n(\mathbf{r})]$ & one-body orbitals ϕ s and find $n(\mathbf{r})$

  Density Functional Theory & KS Scheme (DFT)

- Make a reasonable guess for Ψ_T and throw dice to *estimate* the properties

  Continuum Quantum Monte Carlo (QMC)

Basics of QMC for ES

For *N*-electron system $\{\mathbf{R}\} = (\mathbf{r}_1, \cdots, \mathbf{r}_N)$

Many-body Hamiltonian

$$\hat{H} = \sum_i \frac{1}{2m_e}\nabla^2 + \frac{1}{2}\sum_{i\neq j}\frac{e^2}{|\mathbf{r}_i - \mathbf{r}_j|} + \sum_i V_{ext}(\mathbf{r}_i)$$

Many-body trial wavefunction $\Psi_T(\mathbf{R})$

$$E_T = \frac{\int d^{3N}\mathbf{R}\ \Psi_T^*(\mathbf{R})\hat{H}\Psi_T(\mathbf{R})}{\int d^{3N}\mathbf{R}\ |\Psi_T(\mathbf{R})|^2}, \quad E_T \geq E_0$$

QMC

$$< E_T >= \frac{\sum_i^M w(\mathbf{R}_i)E_L(\mathbf{R}_i)}{\sum_i^M w(\mathbf{R}_i)}, \quad E_L = \frac{\hat{H}\Psi_T(\mathbf{R})}{\Psi_T(\mathbf{R})}$$

Essentials of QMC

Note that

$$E_T = < E_T > |_{M\to\infty}, \quad E_0 \leftarrow E_T|_{\Psi_T \to \Psi}$$

QMC methods employ

- $\Psi_T(\mathbf{R})$, compact, fast to compute, and accurate
- Efficient stochastic sampling to generate large *M*

Variational Monte Carlo (VMC)

Variational parameters

$$E_{VMC} = \min_{\alpha} \langle \Psi_T(\mathbf{R};\alpha)|\hat{H}|\Psi_T(\mathbf{R};\alpha)\rangle$$

w

$|\Psi_T|^2$

Diffusion Monte Carlo (DMC)

$$E_{DMC} = \langle \Phi_0|\hat{H}|\Psi_T\rangle, \quad \Phi_0 = \lim_{\beta\to\infty} \exp^{-\beta\hat{H}} \Psi_T$$

$\Phi_0 \Psi_T$

Efficiency of QMC

- QMC employs sampling to obtain

$$< E_T >= \frac{\sum_i^M w(\mathbf{R}_i) E_L(\mathbf{R}_i)}{\sum_i^M w(\mathbf{R}_i)}, \quad E_L = \frac{\hat{H}\Psi_T(\mathbf{R})}{\Psi_T(\mathbf{R})}$$

with an error bar $\delta = \frac{\sigma}{\sqrt{M}}$, $\boxed{\sigma^2 =< E_T^2 > - < E_T >^2}$

variance

- Efficiency of QMC simulations is high, when
  - Variance is small: $\sigma \to 0 \text{ as } \Psi_T \to \Psi$ (zero-variance)

    Physical insights & improved optimization
  - M/τ, the rate of MC sample generation is high

    Parallelism, compact form of Ψ_T & optimized kernels

Accelerating QMC

- Better Ψ_T

$$\Psi_T(\mathbf{R}) = e^{J_1+J_2+\cdots} \sum C_i D_i^{\uparrow}(\phi) D_i^{\downarrow}(\phi)$$

$$N = N^{\uparrow} + N^{\downarrow}$$

Anti-symmetric function
(Pauli principle)

$$D_i^{\uparrow} = \det \begin{vmatrix} \phi_1(\mathbf{r}_1) & \cdots & \phi_1(\mathbf{r}_{N^{\uparrow}}) \\ \vdots & \vdots & \vdots \\ \phi_{N^{\uparrow}}(\mathbf{r}_1) & \cdots & \phi_{N^{\uparrow}}(\mathbf{r}_{N^{\uparrow}}) \end{vmatrix}$$

Single-particle orbitals

Correlation (Jastrow)

$$J_1 = \sum_i^N \sum_I^{ions} u_1(|\mathbf{r}_i - \mathbf{r}_I|)$$

$$J_2 = \sum_{i \neq j}^N u_2(|\mathbf{r}_i - \mathbf{r}_j|)$$

SPOs $\phi_i = \sum_k^{k=N_b} c_k^i \Phi_k$

Basis sets: molecular orbitals, plane-wave and grid-based orbitals

Accelerating QMC

- Better Ψ_T

- Improved algorithms

$$\phi_i = \sum_{k}^{k=N_b} c_k^i \Phi_k$$

$$\sum_{i}^{M\to\infty} C_i D_i^{\uparrow} D_i^{\downarrow}$$

Optimization of Ψ_T

- einspline library, http://einspline.svn.sourceforge.net/
- Clark et al., JCP **135** 244105 (2011)
- Umrigar, *et. al.*, PRL **98**,110201 (2007)

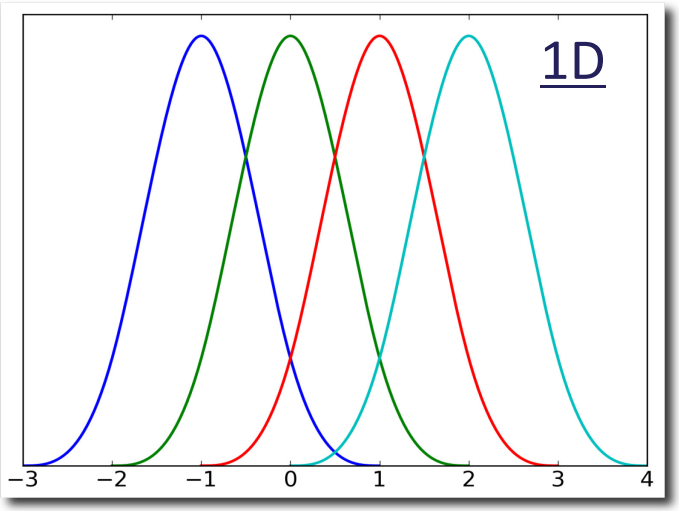
Evaluating single-particle orbitals

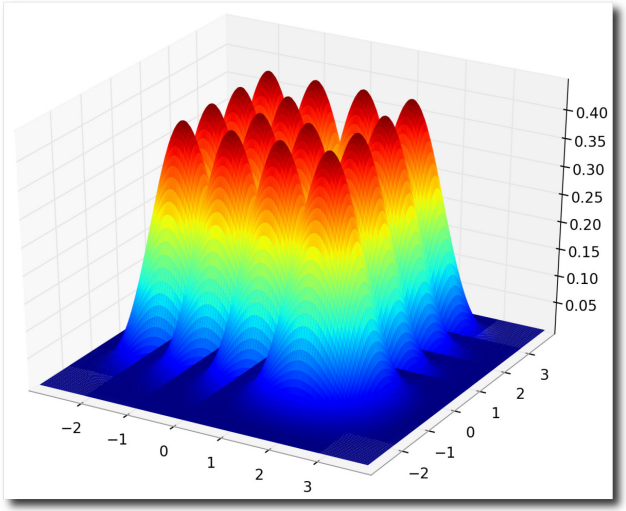
- In QMC, we evaluate orbitals at a specific point in space
  - Most other methods evaluate integrals over all space directly
- We find 3D cubic B-spline basis is fastest for large systems
  - Basis elements are strictly local
  - Only 64 elements are nonzero at each point

$$B_{ijk}(x, y, z) = a_i(x) b_j(y) c_k(z)$$

a,b,c piecewise cubic polynomials

  - Per-orbital evaluation time is independent of N
  - Table is constructed only once and read-only





Accelerating QMC

- Better Ψ_T

$$\phi_i = \sum_{k}^{k=N_b} c_k^i \Phi_k$$

- Improved algorithms

$$\sum_{i}^{M\to\infty} C_i D_i^{\uparrow} D_i^{\downarrow}$$

Optimization of Ψ_T

- einspline library, http://einspline.svn.sourceforge.net/
- Clark et al., JCP **135** 244105 (2011)
- Umrigar, *et. al.*, PRL **98**,110201 (2007)

Accelerating QMC

- Better Ψ_T
- Improved algorithms
- Faster computers
- Bigger computers

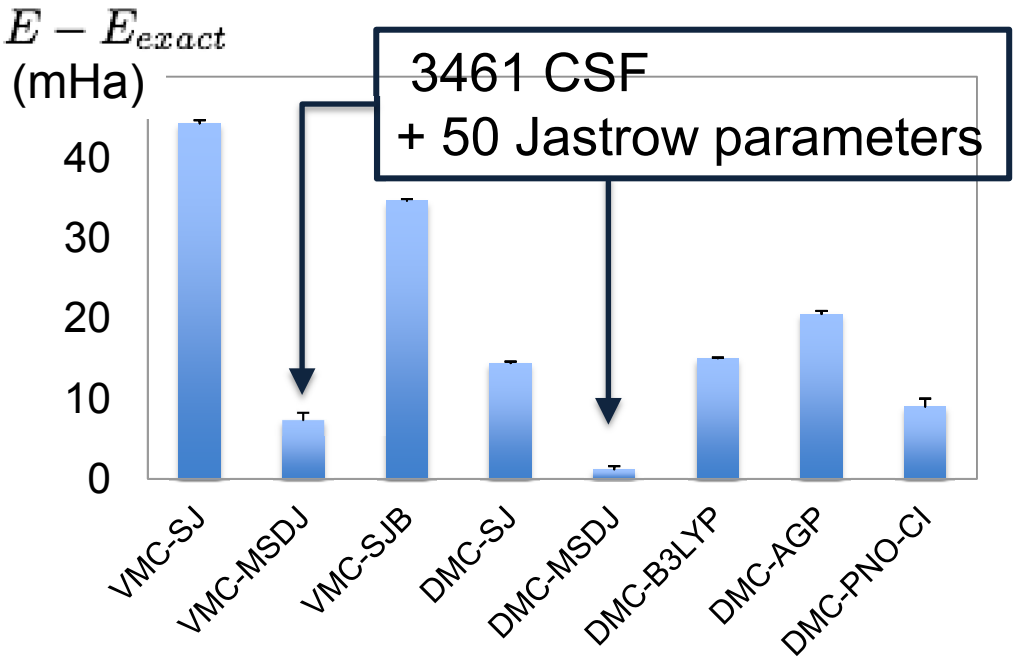
Increase the QMC efficiency
Minimize time-to-solution (wall-clock time) to reach a target error bar

More science

State-of-art QMC

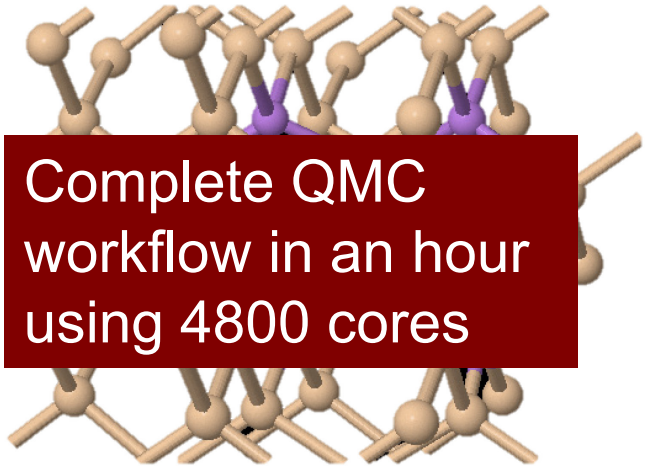
- Fast algorithm for multi-determinant evaluation
- Improved energy minimization in VMC
- QMCPACK: efficient and scalable QMC for large clusters of multi-core and GPUs

Energy of H_2O



Formation energy of a native defect in Si

E_f = 3.07 (11) eV



Complete QMC workflow in an hour using 4800 cores

Characteristics of QMC

DMC pseudo code

for generation $= 1 \cdots N_{\mathrm{MC}}$ **do**
  for walker $= 1 \cdots N_w$ **do**
    let $\mathbf{R} = \{\mathbf{r}_1 \ldots \mathbf{r}_N\}$
    for particle $i = 1 \cdots N$ **do**
      set $\mathbf{r}_i^{'} = \mathbf{r}_i + \delta$
      let $\mathbf{R}^{'} = \{\mathbf{r}_1 \ldots \mathbf{r}_i^{'} \ldots \mathbf{r}_N\}$
      ratio $\rho = \Psi_T(\mathbf{R}^{'})/\Psi_T(\mathbf{R})$
      if $\mathbf{r} \rightarrow \mathbf{r}'$ is accepted **then**
        update the state of a walker
      end if
    end for{particle}
    Compute $E_L = \hat{H}\Psi_T(\mathbf{R})/\Psi_T(\mathbf{R})$
  end for{walker}
  Reweight and branch walkers
  Update E_T and collect properties
end for{generation}

- Ample opportunity for parallelism
  - Configurations
  - K-point
  - Walker parallelization

Characteristics of QMC

DMC pseudo code

for generation $= 1 \cdots N_{\mathrm{MC}}$ **do**
  for walker $= 1 \cdots N_w$ **do**
    let $\mathbf{R} = \{\mathbf{r}_1 \ldots \mathbf{r}_N\}$
    for particle $i = 1 \cdots N$ **do**
      set $\mathbf{r}_i^{'} = \mathbf{r}_i + \delta$
      let $\mathbf{R}^{'} = \{\mathbf{r}_1 \ldots \mathbf{r}_i^{'} \ldots \mathbf{r}_N\}$
      ratio $\rho = \Psi_T(\mathbf{R}^{'})/\Psi_T(\mathbf{R})$
      if $\mathbf{r} \rightarrow \mathbf{r}'$ is accepted **then**
        update the state of a walker
      end if
    end for{particle}
    Compute $E_L = \hat{H}\Psi_T(\mathbf{R})/\Psi_T(\mathbf{R})$
  end for{walker}
  Reweight and branch walkers
  Update E_T and collect properties
end for{generation}

- Ample opportunity for parallelism
  - Configurations
  - K-point
  - Walker parallelization
- Freedom in Ψ_T
  - Compute vs Memory
- Computationally demanding
  - Ratio, update & Local energy
  - Random access

Characteristics of QMC

DMC pseudo code

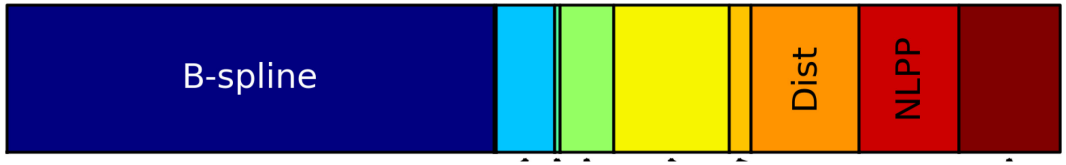
```
for generation = 1 ··· N_MC do
  for walker = 1 ··· N_w do
    let R = {r_1 ... r_N}
    for particle i = 1 ··· N do
      set r'_i = r_i + δ
      let R' = {r_1 ... r'_i ... r_N}
      ratio ρ = Ψ_T(R')/Ψ_T(R)
      if r → r' is accepted then
        update the state of a walker
      end if
    end for{particle}
    Compute E_L = Ĥ Ψ_T(R)/Ψ_T(R)
  end for{walker}
  Reweight and branch walkers
  Update E_T and collect properties
end for{generation}
```

- Ample opportunity for parallelism
  - Configurations
  - K-point
  - Walker parallelization
- Freedom in Ψ_T
  - Compute vs Memory
- Computationally demanding
  - Ratio, update & Local energy
  - Random access
- Communication light but need to
  - Global sum
  - Load balance

Why (not) QMC on GPUs?

- Profile on Xeon 5400 (Harpertown)

% of Kernel



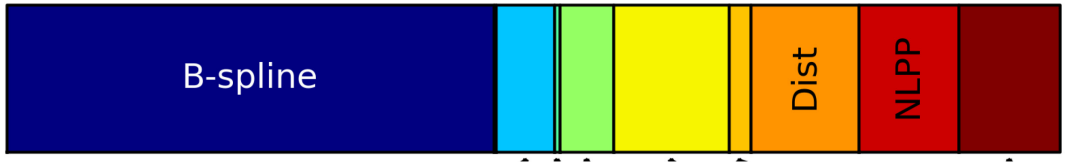
Observations

- B-spline 30-50 % of total run time
  - Random access
  - Few flops per load, memory bandwidth limited
  - Memory use grows with the problem size N^2
- No other dominant kernels
  - Many kernels and various code paths depending on a problem and physical representations at run time
  - No library routines
- For a walker, the parallelism is limited by N

Why (not) QMC on GPUs?

- Profile on Xeon 5400 (Harpertown)

% of Kernel



Observations

- B-spline 30-50 % of total run time
  - Random access
  - **Few flops per load, memory bandwidth limited**
  - Memory use grows with the problem size N^2
- No other dominant kernels
  - Many kernels and various code paths depending on a problem and physical representations at run time
  - No library routines
- For a walker, the parallelism is limited by $N \sim 10\text{-}100$

QMC on GPU

for walker $= 1 \cdots N_w$ **do**
  let $\mathbf{R} = \{\mathbf{r}_1 \ldots \mathbf{r}_N\}$
  for particle $i = 1 \cdots N$ **do**
    set $\mathbf{r}_i^{'} = \mathbf{r}_i + \delta$
    let $\mathbf{R}^{'} = \{\mathbf{r}_1 \ldots \mathbf{r}_i^{'} \ldots \mathbf{r}_N\}$
    ratio $\rho = \Psi_T(\mathbf{R}^{'})/\Psi_T(\mathbf{R})$
    if $\mathbf{r} \rightarrow \mathbf{r}'$ is accepted **then**
      update the state of a walker
    end if
  end for{particle}
  Compute $E_L = \hat{H}\Psi_T(\mathbf{R})/\Psi_T(\mathbf{R})$
end for{walker}
Reweight and branch walkers
Update E_T and collect properties

Loops

- Restructure the algorithm and data structure to expose & exploit parallelisms

multiple walkers per kernels

* Esler, Kim, Shulenburger & Ceperley, CISE (2010)

QMC on GPU

for walker $= 1 \cdots N_w$ **do**
  let $\mathbf{R} = \{\mathbf{r}_1 \ldots \mathbf{r}_N\}$
  for particle $i = 1 \cdots N$ **do**
    set $\mathbf{r}_i^{'} = \mathbf{r}_i + \delta$
    let $\mathbf{R}^{'} = \{\mathbf{r}_1 \ldots \mathbf{r}_i^{'} \ldots \mathbf{r}_N\}$
    ratio $\rho = \Psi_T(\mathbf{R}^{'})/\Psi_T(\mathbf{R})$
    if $\mathbf{r} \rightarrow \mathbf{r}'$ is accepted **then**
      update the state of a walker
    end if
  end for{particle}
  Compute $E_L = \hat{H}\Psi_T(\mathbf{R})/\Psi_T(\mathbf{R})$
end for{walker}
Reweight and branch walkers
Update E_T and collect properties

Loops

- Restructure the algorithm and data structure to expose & exploit parallelisms

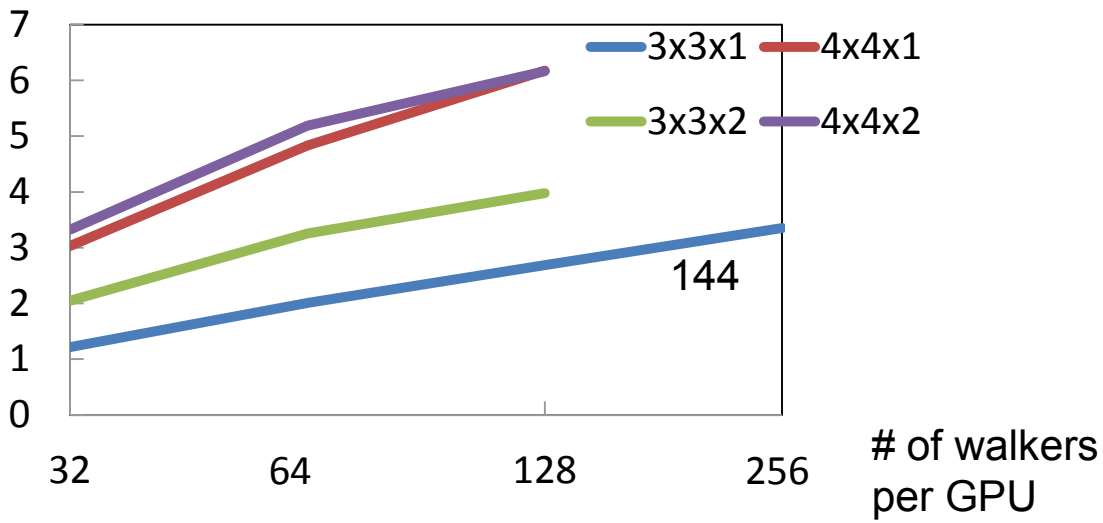
  multiple walkers per kernels

- Mixed precision sufficient for the target accuracy
- Higher-level implementation intact
- MPI for load balancing & reductions : sustain high parallel efficiency

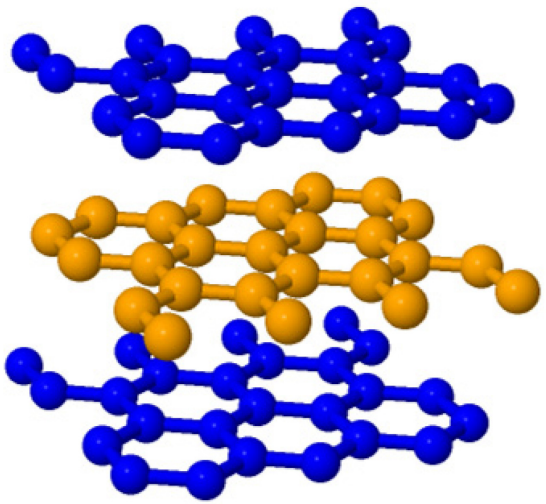
* Esler, Kim, Shulenburger & Ceperley, CISE (2010)

Performance Results

Speedup wrt hex-core X5560



Graphite using 24x24x4

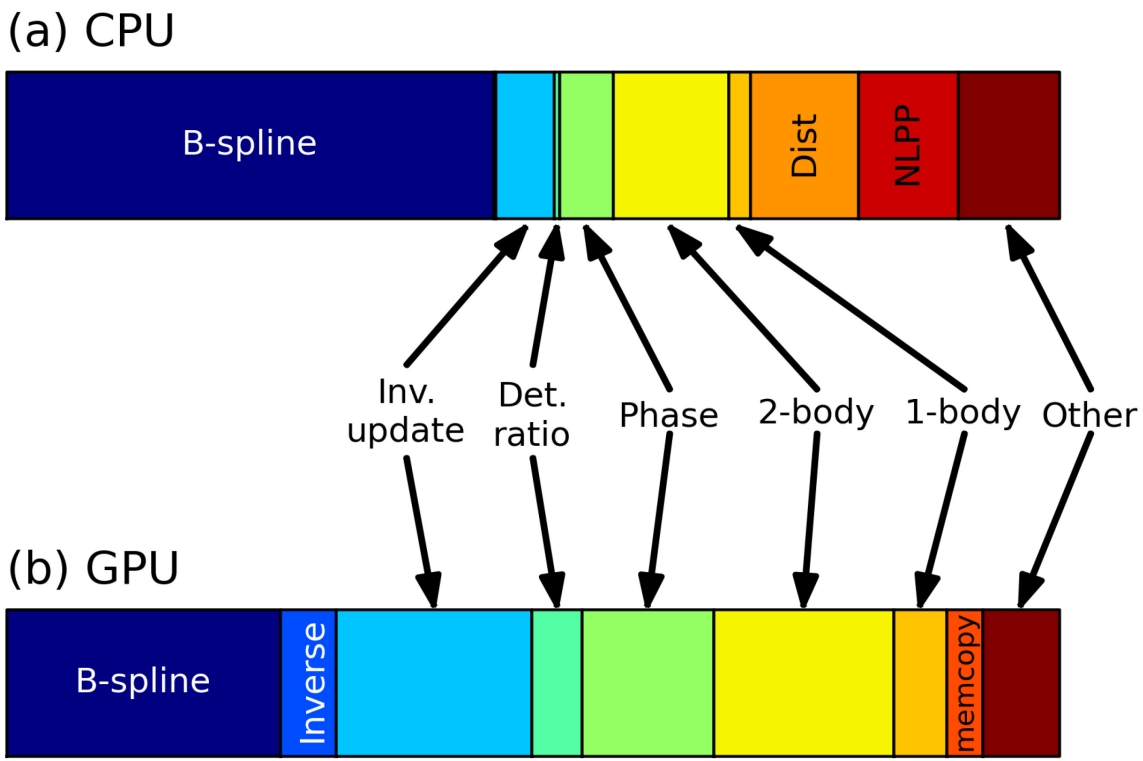


- CPU is in double precision, GPU is in mixed precision
- Optimal throughput with large system, many walkers
- With large system, speed is above 5 NUMA nodes
- Limited by memory but workaround possible

Data taken on Keeneland at NICS, 2012

Performance Analysis

% of Kernel



- B-spline: bandwidth limited on CPU & GPU
- Inverse update : bandwidth limited on GPU
- Inverse on GPU: periodic re-computation for accuracy
- Others: compute limited, C++ compiler limited on CPU

Inverse matrices

- Determinant ratios and derivatives use inverse of the determinant matrix
- Only one row changes at a time: use an update formula

$$\left[A + e_k \delta^T\right]^{-1} = A^{-1} - \frac{\boxed{A^{-1} e_k}\,\boxed{\delta^T A^{-1}}}{1 + \delta^T A^{-1} e_k},$$

matvec

outer product

- Requires matrix-vector product, then adding an outer product to A^{-1}
- Dominant computation only 1 FLOP per load/store → bandwidth limited
- Matrices fit in CPU cache: GPU speedup is relatively poor

Moving forward

- Obtained the speedup as expected by the bandwidth and peak FLOPS

- Enable larger systems to be simulated
  - Use GPU peer-to-peer to allow distribution of read-only orbital dataset between GPUs on same node
- Reduce the time-to-solution, fewer walkers/GPU
  - Launch several kernels in parallel
  - Fast atomics to communicate between blocks
  - Facilitated by newer GPUs and CUDA
- Utilize CPU as well
- Breakthrough science

Challenges

- New methods and algorithms
  - E.g., current algorithms and data structure on GPUs are not ideal for the fast algorithm for massive multi-determinant expansions

$$\sum_{i}^{M\to\infty} C_i D_i^{\uparrow} D_i^{\downarrow}$$

10-100 times more expensive than M=1

- Programming models and software environments for productivity: need portable and efficient solutions on multiple platforms, now and future

Acknowledgements

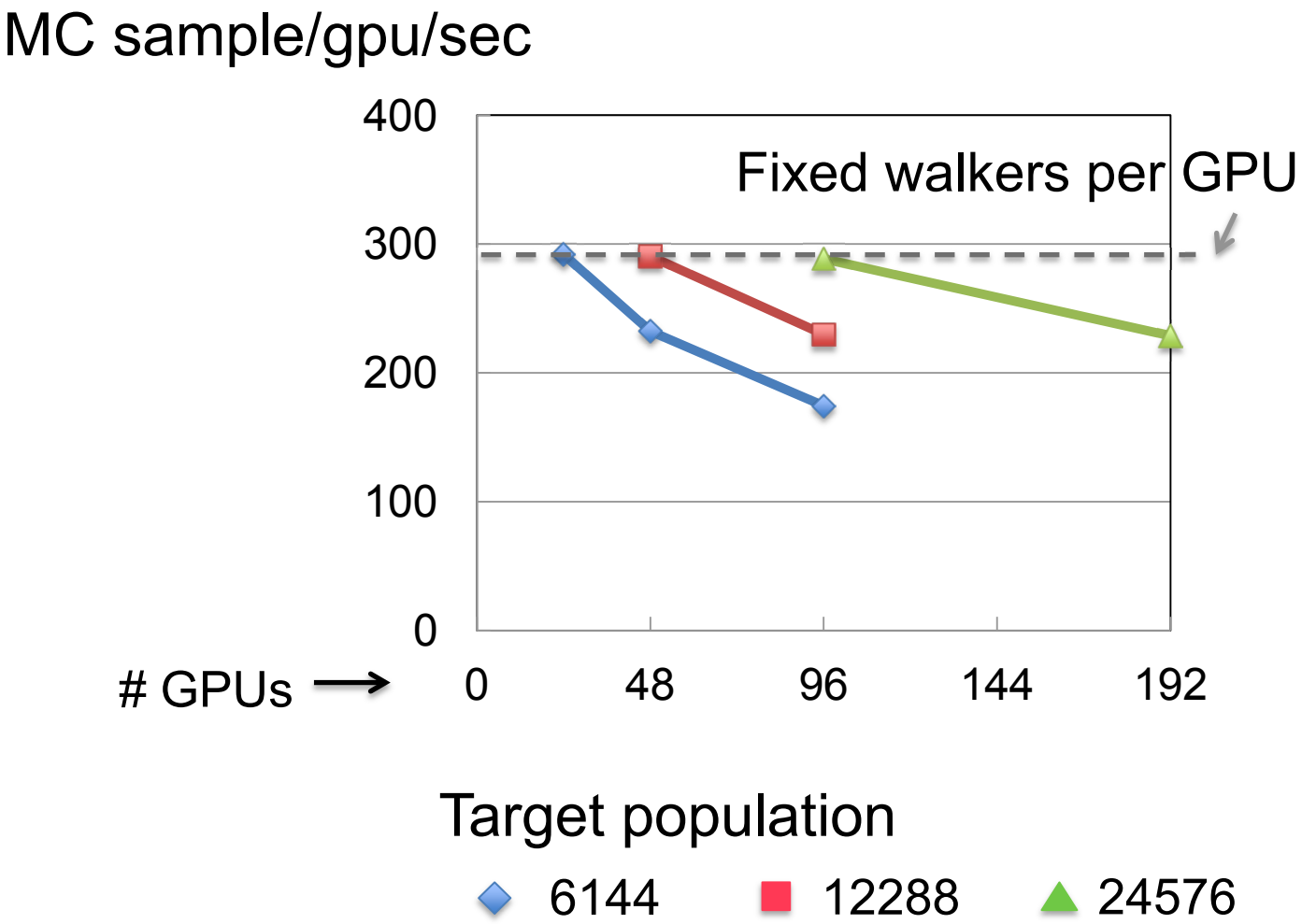
Supported by

- QMC Endstation (DOE-ASCR)
- PetaApps (NSF-DMR, OCI)
- Materials Computation Center, University of Illinois (NSF-DMR)
- Center for Defect Physics, ORNL (DOE-BES)
- ORNL LDRD (DOE-ASCR)
- National Center for Supercomputing Applications (NSF)

Computing resources provided by

- Oak Ridge Leadership Computing Facility (OLCF)
- NSF Teragrid facilities at NCSA, NICS, PSC and TACC
- National Energy Research Scientific Computing Center (NERSC)
- Argon Leadership Computing Facility (ALCF)

Parallel Scaling on multiple GPUs

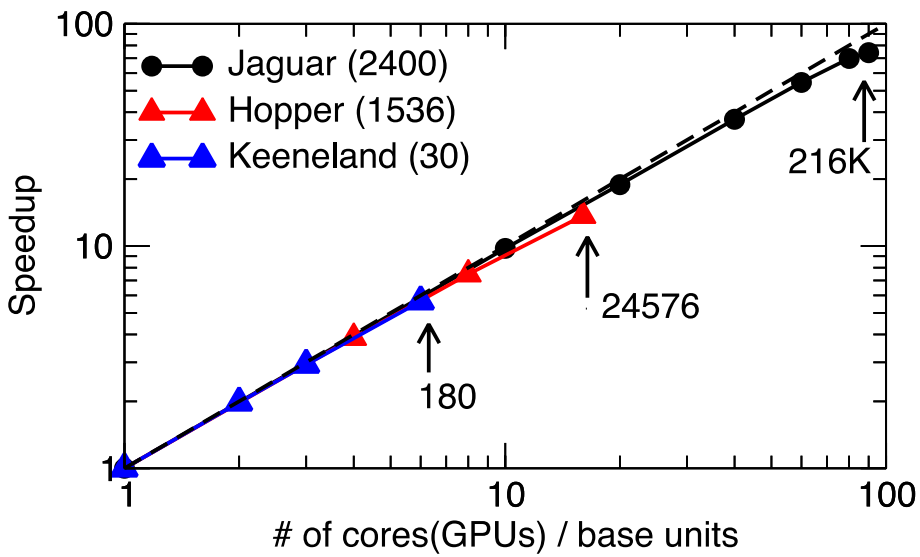


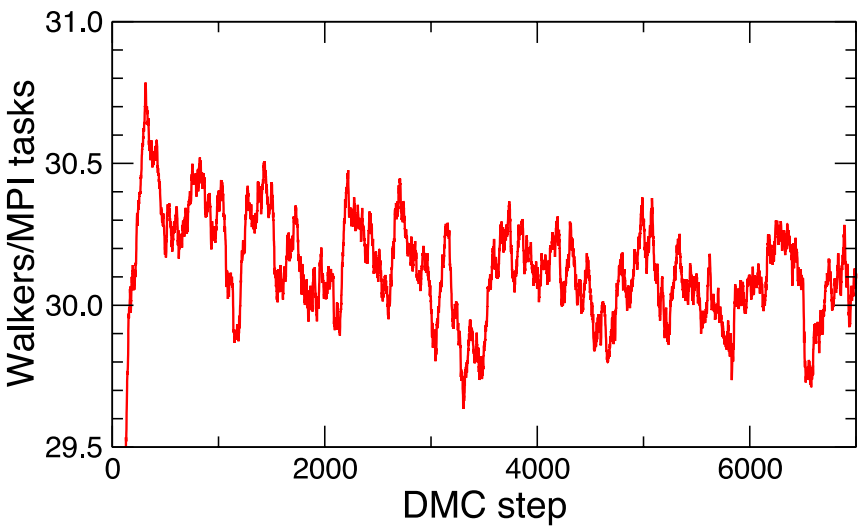
3x3x1 Graphite: N=144

Data taken on Keeneland at NICS

Parallel Performance

- DMC scaling is almost perfect , > 90% efficiency
  - Limited by collectives for $E_T,\ N_p^w - < N^w >$
- 1 MPI to 1 GPU mapping
  - Acceleration allows more walkers per MPI
  - Large average number of walkers per MPI task, thus small fluctuations : easy to balance walkers per node





QMCPACK: QMC for HPC

- Implements essential QMC algorithms and best practices developed over 20yrs+
- Designed for large-scale QMC simulations of molecules, solids and nanostructures on massively parallel machine
  - (OpenMP,CUDA)/MPI Hybrid parallelization
  - Object-oriented and generic programming
- Apply software engineering
  - Reusable and extensible solution for new development
  - Standard open-source libraries and utilities for development, compilation and execution
  - Portable and scalable I/O with XML/HDF5

http://qmcpack.cmscc.org

QMC advantages: accuracy and scalability

- Address high-dimensionality through stochastic sampling
- Exact for bosons, very accurate for fermions
- Applicable to a wide range of problems
  - Any boundary conditions: molecular and solid-state systems
  - Dimensionality: 1D, 2D, and 3D
  - Representation: atomistic to model Hamiltonians
- Scale with a few powers in system size: $O(N^3)$-$O(N^4)$
  - Routine calculations of 100s-1000s electrons
- Ample opportunities of parallelism

QMC has enabled accurate predictions of correlated electronic systems: plasmas to molecules to solids; insulators to highly correlated metals

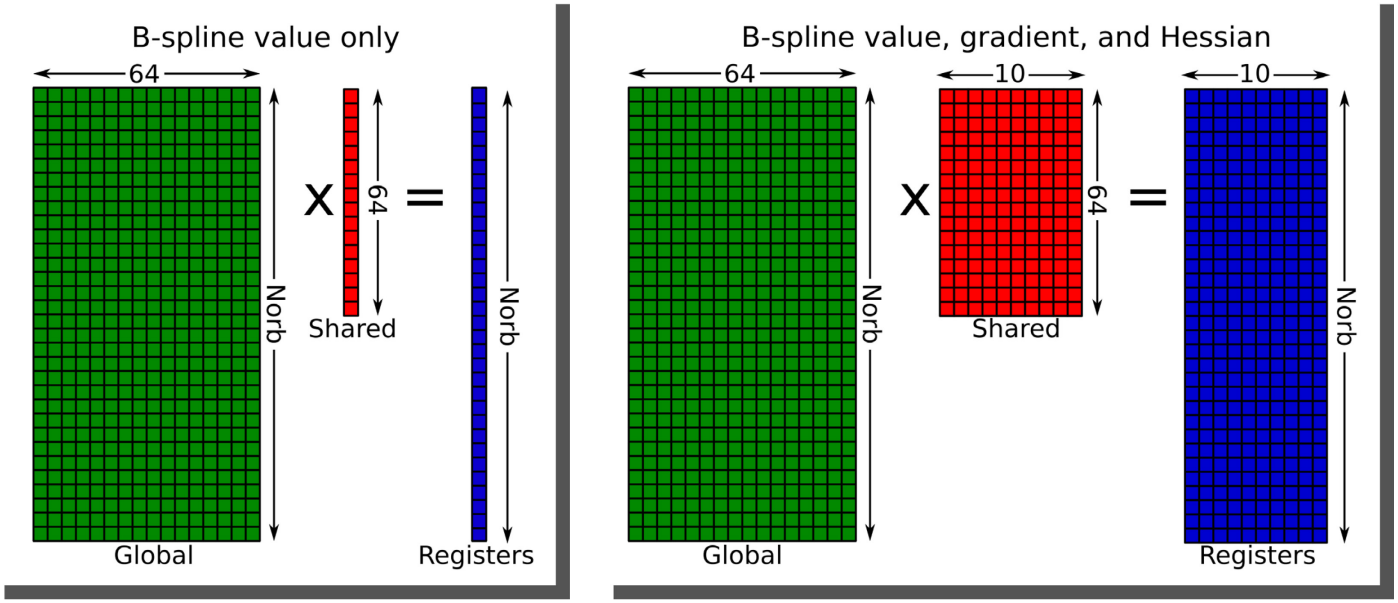
Evaluating 3D B-splines

- First, evaluate 64 basis elements

$$B_{ijk}(x,y,z) = a_i(x)b_j(y)c_k(z)$$

  - 1D functions are piecewise cubic polynomials
- Then, do matrix-vector product with orbital index fastest
- Value: 2 FLOPS per load: bandwidth limited, 75 GB/s on T10
- Value, gradient, and Hessian: 20 FLOPS per load 216 GFLOPS (SP)

~256 walkers
x



Lessons learned

- Many kernels to port
  - No single major kernel
  - Multiple forms of Ψ_T
  - Subset of functionality
  - ~100 CUDA kernels (10K lines of the code)
- Algorithms and data restructuring
  - Optimize memory use
  - Allocations and anonymous buffers for bulk transfer
- Debugging stochastic method is hard but OO programming helps
- No software infrastructure in 2010
  - Much has improved but code re-use is questionable

Recent QMC Development*

- Advances in trial wave functions
  - Backflow, geminals (BCS-type) and Pfaffians
  - *Improved* $\{\phi\}$ from various electronic structure methods
  - Efficient representation of $\{\phi\}$: PWs, MOs, bspline, blips
- Advances in optimizations
  - Energy minimization in VMC [1]
  - Node optimization in DMC [2]
- Finite-size corrections in solids
- Efficient and scalable QMC algorithms and codes

* Review in SciDac 2008, QMC Endstation team; [1] Umrigar, *et. al.*, PRL **98**,110201 (2007); [2] Reboredo, *et. al.*, PRB **79**, 195117 (2009)