Quantum Monte Carlo, keeping up with the HPC Evolution

Jeongnim Kim[1,2], Kenneth P Esler[1]
and David M Ceperley[1,2,3]

[1] National Center for Supercomputing Applications
[2] Materials Computation Center
[3] Department of Physics
University of Illinois at Urbana-Champaign

NCSA MCC

Acknowledgements

QMCPACK developers*

- Kenneth P. Esler (Stoneridge)
- Jeremy McMinis (UI)
- Miguel Morales (LLNL)
- Bryan Clark (Princeton)
- Luke Shulenburger (Sandia)
- Simone Chiesa (W&M)
- Kris Delaney (UCSB)
- Jaron Krogel (UI)

and more

*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)

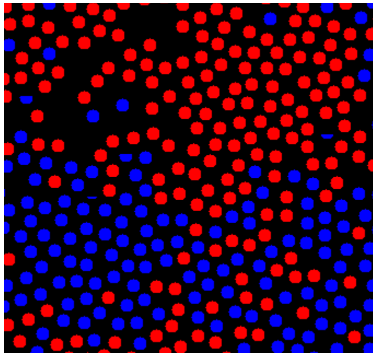
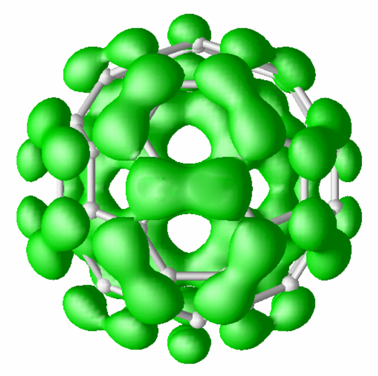
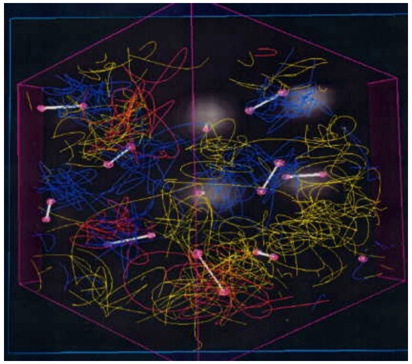
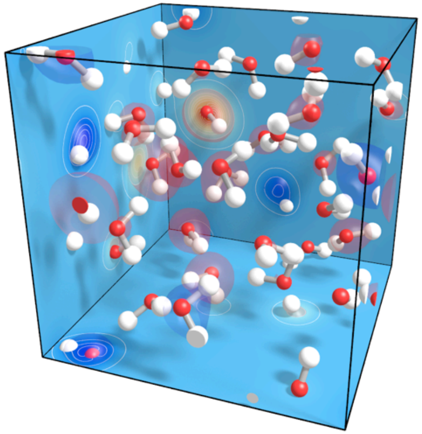
Special thanks to

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

Outline

- Quantum Monte Carlo Methods: accurate, robust and efficient solution for electronic structure calculations, especially for correlated systems
- QMC on clusters of multi-core and GPUs
  - OpenMP/MPI hybrid
  - CUDA/MPI hybrid
- Prospect of QMC algorithms on hybrid architectures
- Conclusions

Quest for Accurate Quantum Simulations: harnessing computing power

- Hard-core bosons on a CDC 6600 (1974)
- Electronic and structure properties of carbon/silicon clusters on HP 9000/715 cluster and Cray Y-MP (1995)
- Coupled Electron-Ion Monte Carlo simulations of dense hydrogen on Linux Clusters (2000)
- Diffusion Monte Carlo simulations of liquid water on multi-core SMP clusters (2009)

QMC advantages: accuracy and scalability

- 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

- Fundamental High-Pressure Calibration from **All-Electron** Quantum Monte Carlo Calculations, Esler *et al*, PRL (2010)
- Evidence for a first-order liquid-to-liquid transition in high-pressure hydrogen, Morales *et al*, PNAS (2010)

QMCPACK: QMC for HPC

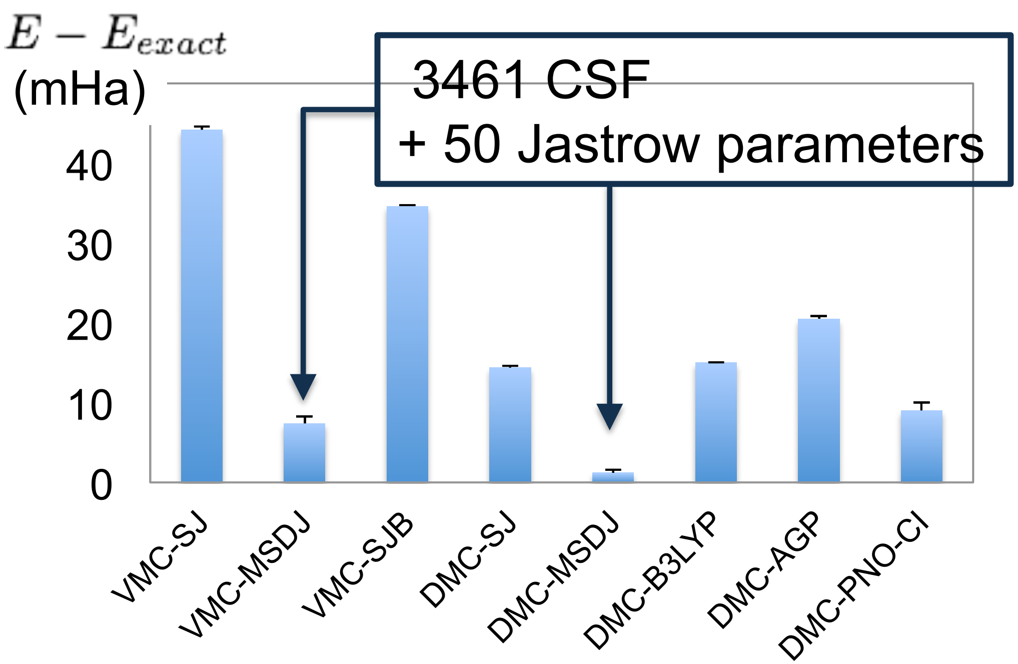
- Implements essential QMC algorithms and best practices developed over 20yrs+
- Designed for large-scale QMC simulations of molecules, solids and nanostructures on massively parallel machine
  - (OpenMP,CUDA)/MPI Hybrid parallelization
  - Object-oriented and generic programming in C++
- Apply software 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

More recent QMC development*

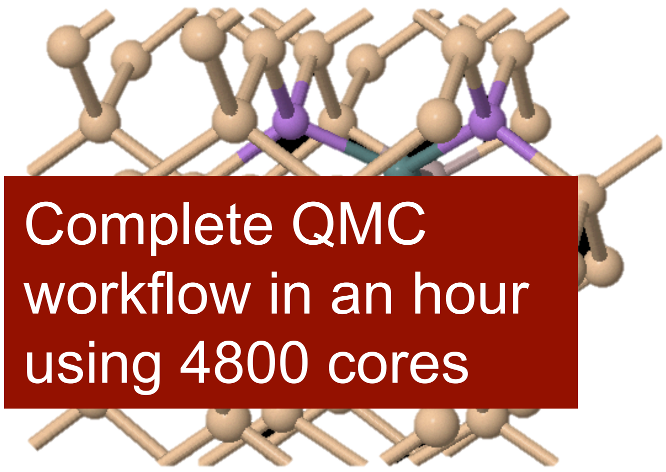
- Efficient and scalable QMC algorithms
- Fast algorithm for multi-determinant evaluation
- Improved energy minimization in VMC and DMC

Energy of H_2O



Formation energy of a native defect in Si

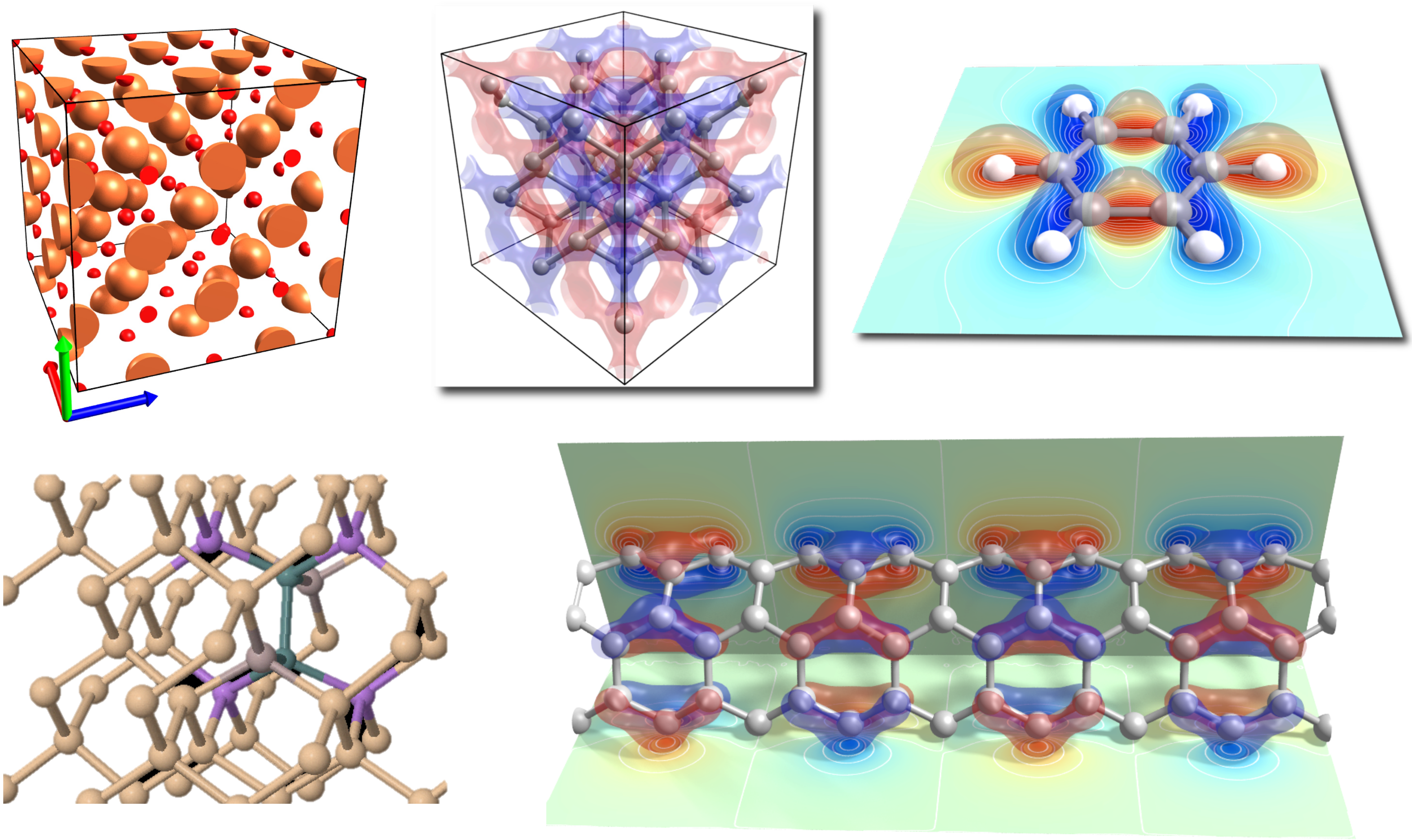
$E_f = 3.07\ (11)$ eV



Complete QMC workflow in an hour using 4800 cores

* By QMCPACK developers

QMC in Action



QMC keeping up with HPC evolution

- Increasing accuracy, computational complexity and problem size of QMC simulations with HPC evolution
  - Model Hamiltonian in 70s, e.g., hard-sphere and LJ potential
  - Homogeneous electron gas in 80s, seminal work by Ceperley and Alder laid the foundation of DFT
  - Atoms, molecules and bulk
  - Recently, routine QMC simulations of 1000s of electrons including disordered solids
- Shorter time-to-solution = More Science
- Can QMC continue?

High-performance computing in 2010s

- Petaflop machines have been around, e.g. Jaguar (OLCF)
- Sustainable petaflop machines are coming, *e.g.*, Blue Waters at NCSA in 2011

Clusters of Shared-memory Processors (SMP)

- Hierarchical memory and communication
- Fast interconnects & various inter-node topology
- Increasing number of cores per SMP node
  - 8-32 cores are common; more is expected.
- *Fixed* memory per core but *more aggregated* memory per node
- SIMD units: SSE on x86 and VSX on IBM Power 7(P7)
- Large number of threads: simultaneous multi-threading (a.k.a. hyperthreading), *e.g.*, 128 threads on IBM P7 32-core node

Basics of QMC

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

Many-body Hamiltonian

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

Find the solution $\hat{H}|\Psi >= E_0|\Psi >$ & $\langle \Psi | \hat{H} | \Psi \rangle = E_0$

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

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

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

$$E_{VMC} = \min_{\alpha}\langle\Psi_T(\mathbf{R};\alpha)|\hat{H}|\Psi_T(\mathbf{R};\alpha)\rangle \qquad |\Psi_T|^2$$

- Diffusion Monte Carlo (DMC)

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

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

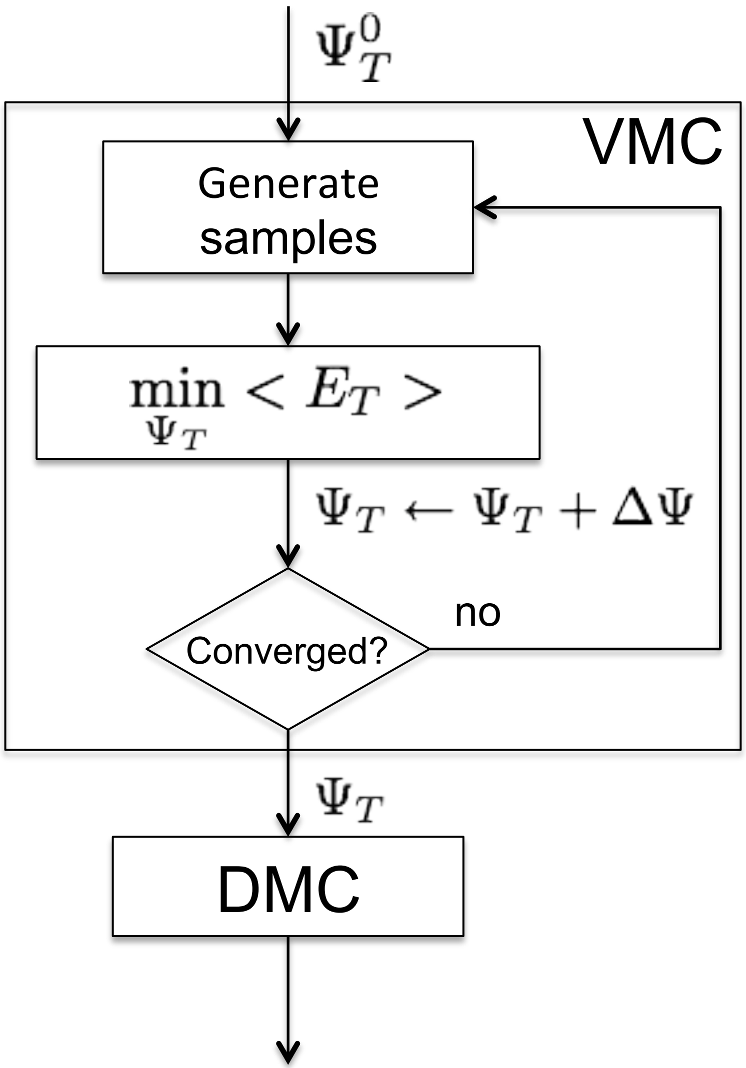
variance

- Minimize wall-clock time to reach a target error bar
- Efficiency of QMC simulations is high, when
  - Variance is small: $\sigma \rightarrow 0 \text{ as } \Psi_T \rightarrow \Psi$ (zero-variance)

    Physical insights & improved optimization
  - The rate of MC sample generation is high

    Parallelism, compact form of Ψ_T & optimized kernels

HowTo for QMC Calculations



- Initial guess Ψ_T^0
  - Compact, easy to evaluate, but close to true Ψ

$$\Psi_T(\mathbf{R}) = J(\boxed{\{\alpha\}}) \sum \boxed{C_i} D_i^{\uparrow}(\phi) D_i^{\downarrow}(\phi)$$

  - Single-particle orbitals $\boxed{\{\phi\}}$ e.g., KS or HF solution
- Find $\{\alpha\}$ & $\{C\}$ to optimize an object function: energy and variation minimization
- Projecting out the ground-state by applying a propagator $e^{-\tau \hat{H}}$

Diffusion Monte Carlo

for generation $= 1 \cdots N_{\text{MC}}$ **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 state of a walker
      end if
    end for{particle}
    Compute $E_L = \hat{H}\Psi_T(\mathbf{R})/\Psi_T(\mathbf{R})$
    Reweight and branch walkers
    Update E_T
  end for{walker}
end for{generation}

Drift & Diffusion

Branch

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 state of a walker
      end if
    end for{particle}
    Compute $E_L = \hat{H}\Psi_T(\mathbf{R})/\Psi_T(\mathbf{R})$
    Reweight and branch walkers
    Update E_T
  end for{walker}
end for{generation}

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

Characteristics of QMC

DMC pseudo code

for generation $= 1 \cdots N_{\text{MC}}$ **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 state of a walker
      end if
    end for{particle}
    Compute $E_L = \hat{H}\Psi_T(\mathbf{R})/\Psi_T(\mathbf{R})$
    Reweight and branch walkers
    Update E_T
  end for{walker}
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

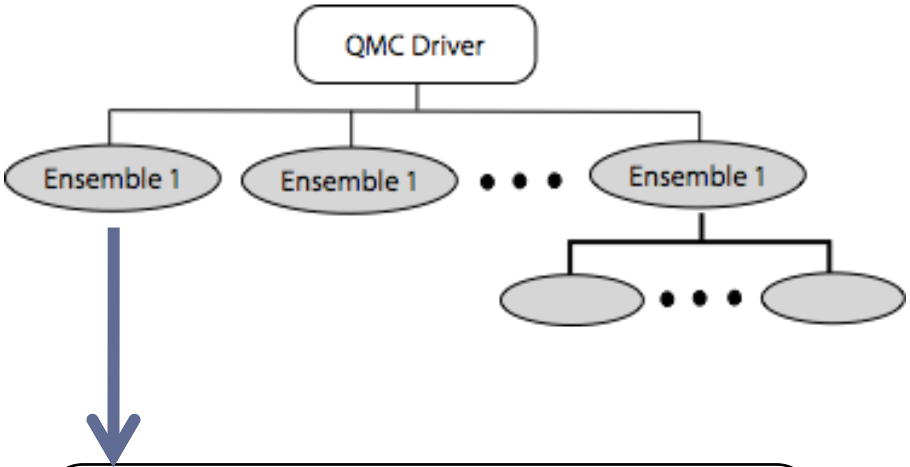
Characteristics of QMC

DMC pseudo code

for generation $= 1 \cdots N_{\mathrm{MC}}$ **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 state of a walker
      end if
    end for{particle}
    Compute $E_L = \hat{H}\Psi_T(\mathbf{R})/\Psi_T(\mathbf{R})$
    Reweight and branch walkers
    Update E_T
  end for{walker}
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

Hierarchical Parallelization of QMC



for generation $= 1 \cdots N_{\mathrm{MC}}$ **do**
  for walker $= 1 \cdots N_w$ **do**

A walker in cache

  Reweight and branch walkers
  Update E_T
  end for{walker}
end for{generation}

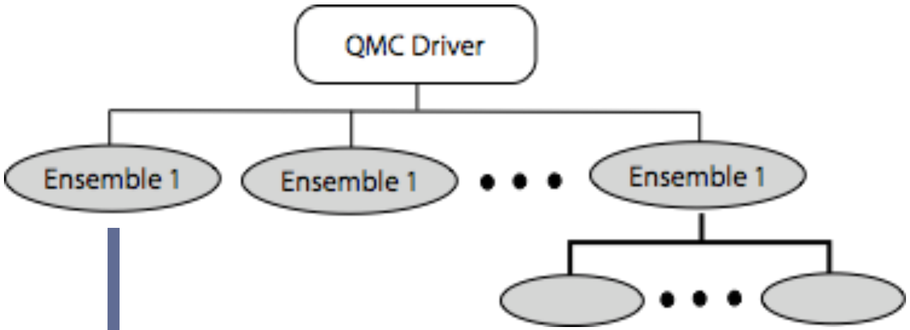
For a given *N*-electron system

1. Multiple instances of correlated configurations: any
2. Multiple k-points : 1-100

   Critical to remove finite-size effects

3. Walker parallelization: $N_w = 10^4 - 10^6$ Multi-core

Hierarchical Parallelization of QMC



for generation $= 1 \cdots N_{\mathrm{MC}}$ **do**
  for walker $= 1 \cdots N_w$ **do**
    Reweight and branch walkers
    Update E_T
  end for{walker}
end for{generation}

For a given *N*-electron system

1. Multiple instances of correlated configurations: any
2. Multiple k-points : 1-100
   Critical to remove finite-size effects

3. Walker parallelization: $N_w = 10^4 - 10^6$
4. *N*-particle : $N - N^3$

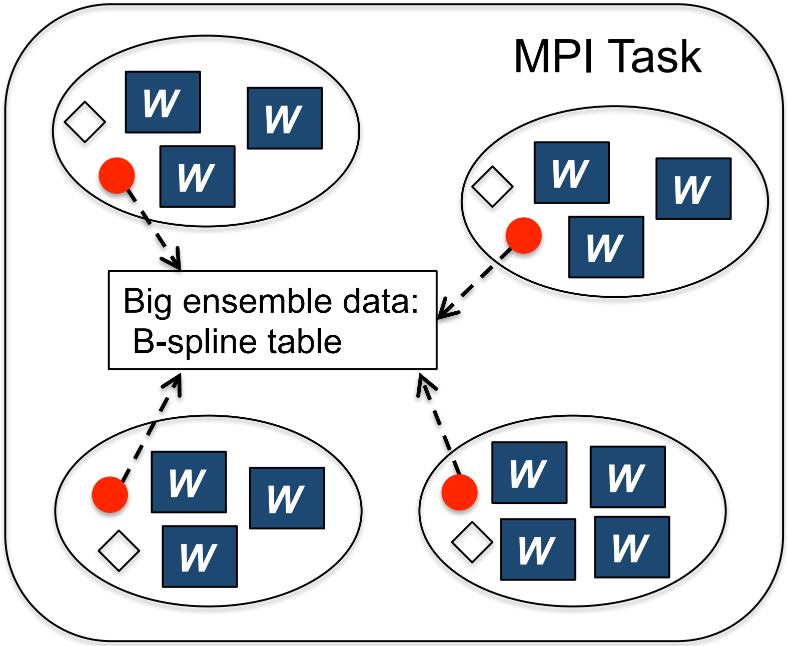
GPU

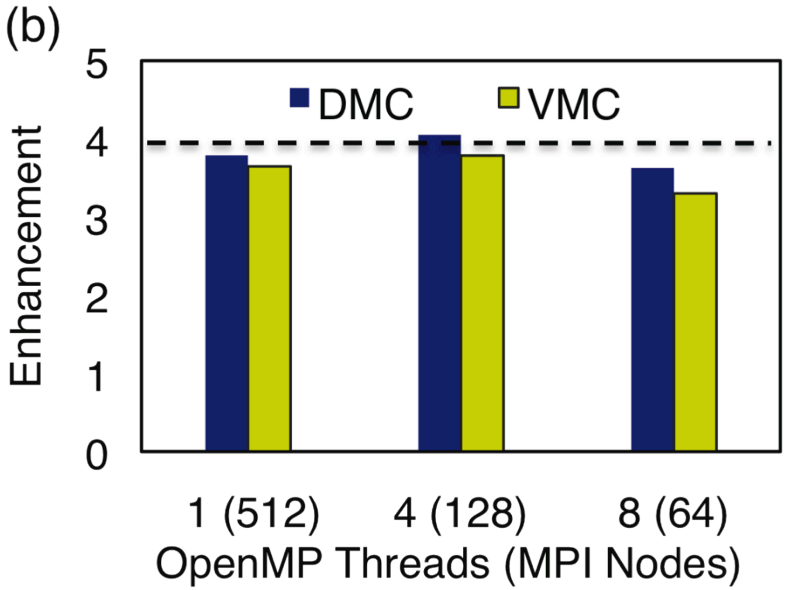
And, more parallelism can exposed

$$\Psi_T(\mathbf{R}) = \Pi_i \Psi_i, \hat{H} = \sum_i \hat{h}_i$$

Hybrid scheme on SMP

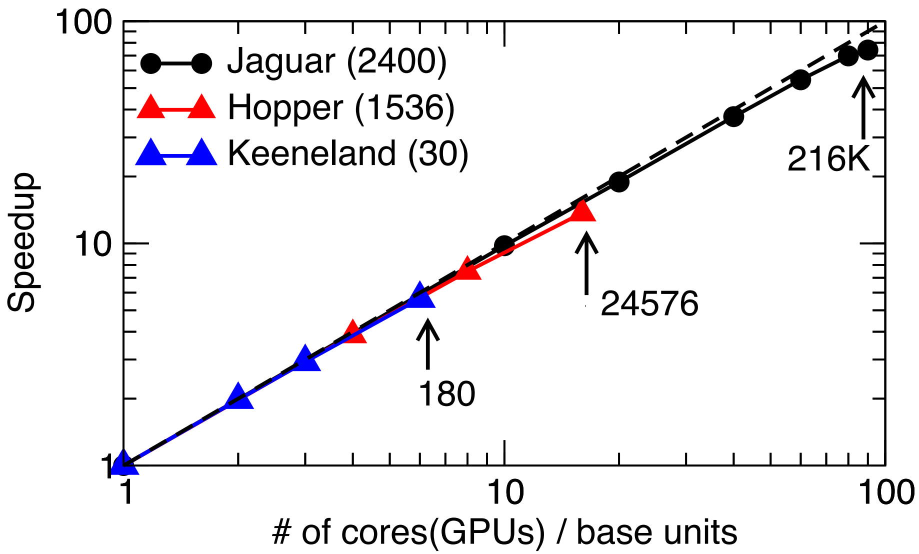
- Maximize performance and reduce the time-to-solution
  - MPI task per SMP, better per NUMA node
  - Multiple walkers per threads
  - Use all the hardware threads available

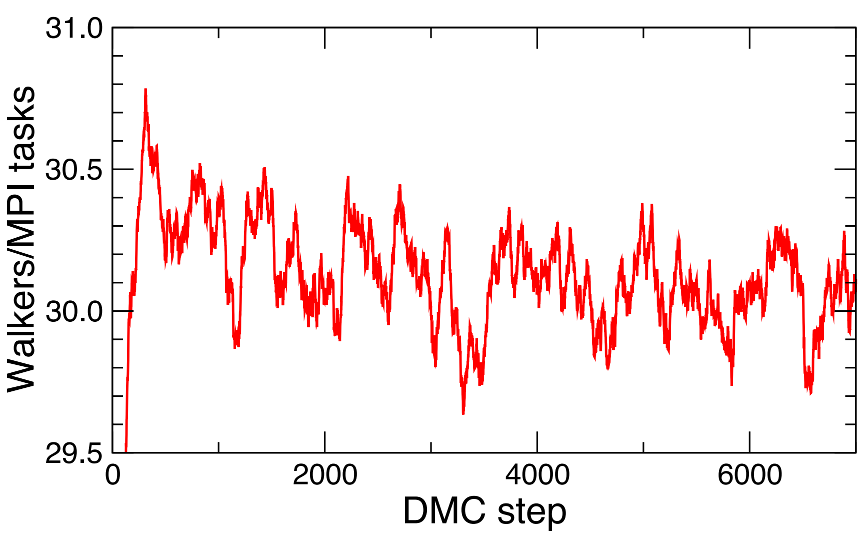




Performance of Hybrid QMC

- DMC scaling is almost perfect , > 90% efficiency
  - Limited by collectives for $E_T, \; N_p^w - < N^w >$
- Open/MPI hybrid helps more than memory footprint
  - Collectives scale O(P^2) *or O(P ln P)* for *P* tasks
  - Large average number of walkers per MPI task, thus small fluctuations : easy to balance walkers per node





NCSA MCC

MCPACK

QMC on Clusters of SMPs

- Compute-heavy and communication-light nature makes QMC an easier parallel problem than other problems
- But, as the parallelism increases > 10^4, many issues arise
  - Limited memory per core
  - MPI performance : collectives
  - I/O : initialization and checkpoint
- MPI/OpenMP provides QMC with simple but effective solutions
  - Standards of both commercial and HPC : rely on steady improvement of the HP infrastructure, compilers and libraries
  - Can exploit hierarchy of memory and communication
  - Large-shared memory per node : minimize data replications, while taking advantage of increasing hardware threads

QMC on GPU

- Why GPU?
  - Many threads, high floating-point performance, and bandwidth
  - Tera- and peta-scale workstations
  - A candidate for the future HPC architecture

- GPU port of QMCPACK*
  - Restructure the algorithm and data structure to exploit parallelism
  - MPI for load balancing & reductions : high parallel efficiency

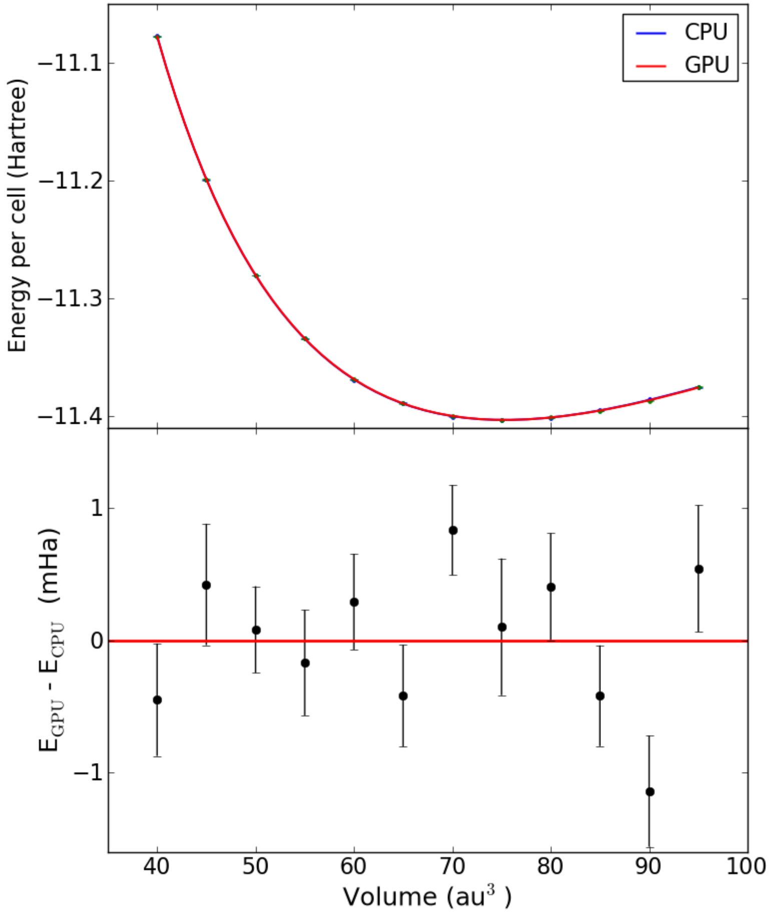
for walker $= 1 \cdots N_w$ **do**
  let $\mathbf{R} = \{\mathbf{r}_1 \ldots \mathbf{r}_N\}$
  for particle $i = 1 \cdots N$ **do**
    set $\mathbf{r}_i' = \mathbf{r}_i + \delta$
    let $\mathbf{R}' = \{\mathbf{r}_1 \ldots \mathbf{r}_i' \ldots \mathbf{r}_N\}$
    ratio $\rho = \boxed{\Psi_T(\mathbf{R}')/\Psi_T(\mathbf{R})}$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

Loops

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

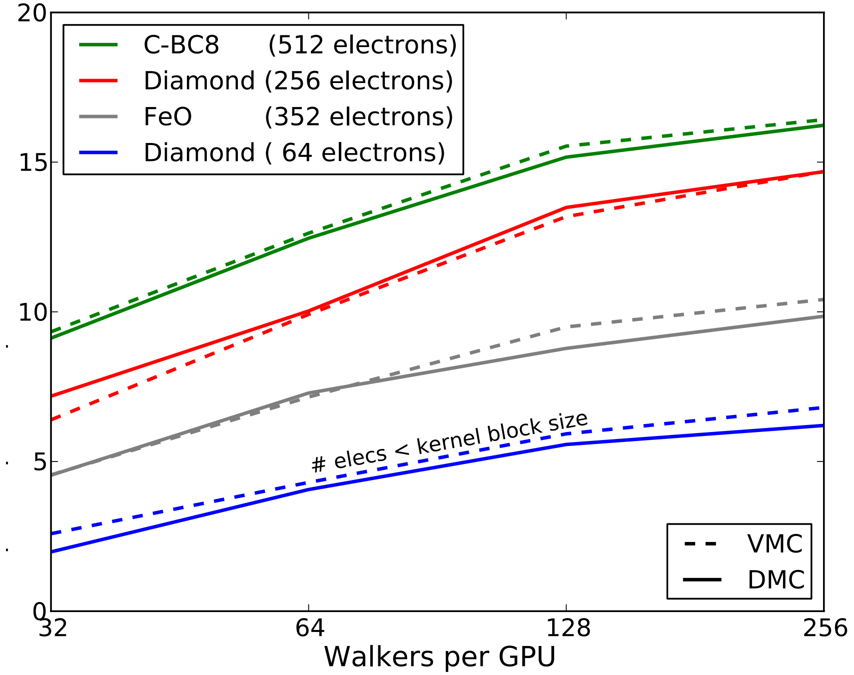
QMC on GPU

Impact of single precision



CPU: double
GPU: mixed, main kernels in single

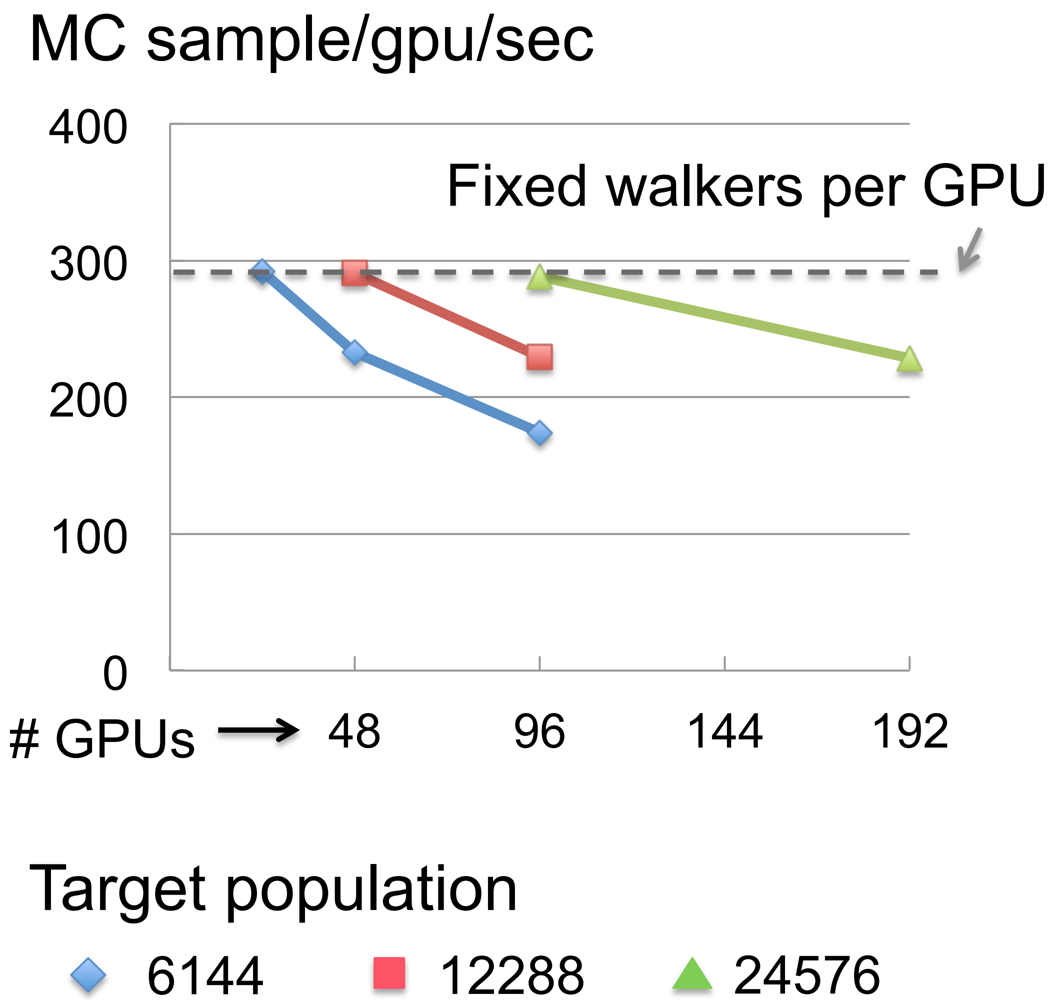
Speedup: 1 GPU/ 4 cores

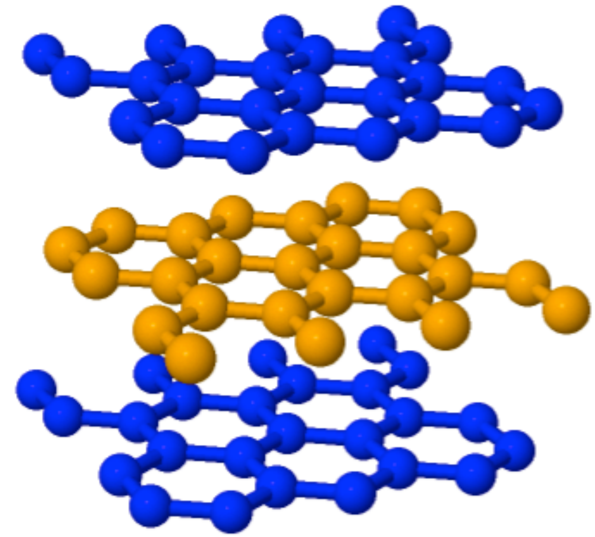


Performance data on NCSA Lincoln cluster

- nVidia G200 GPUs
- Intel Xeon (Harpertown)

Scaling on multiple GPUs

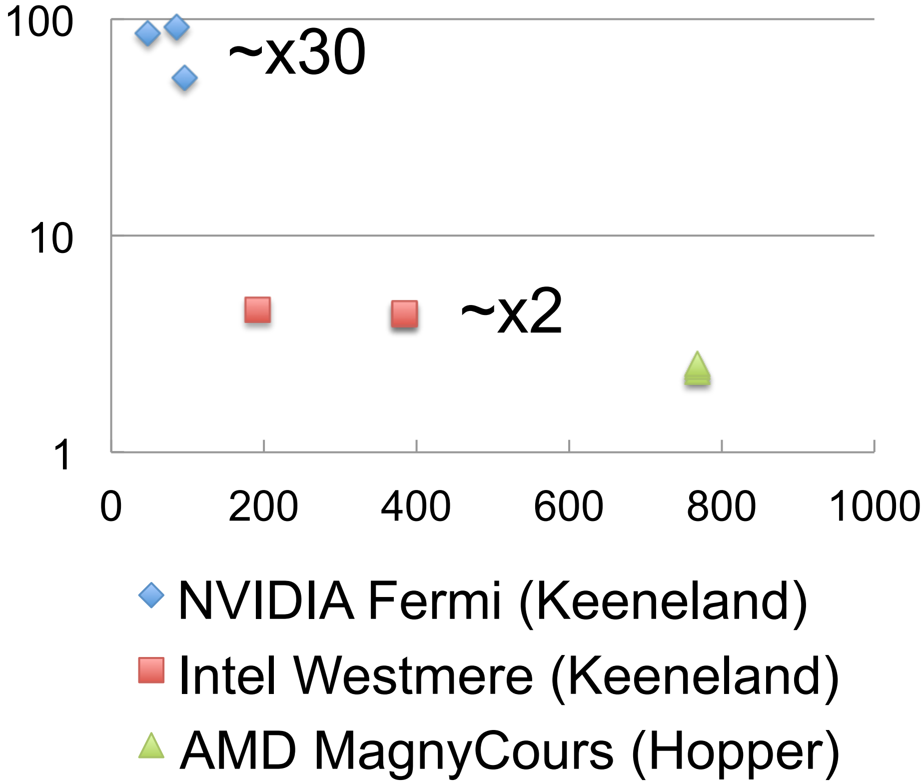




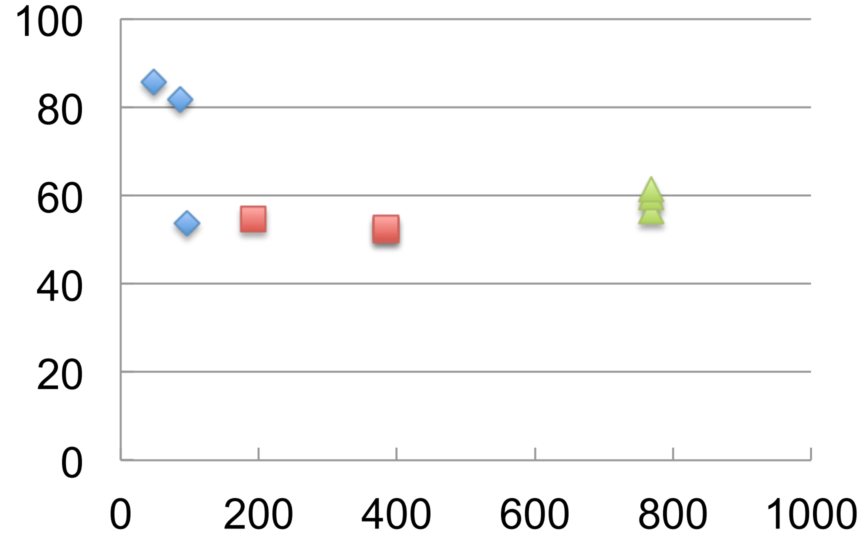
- 3x3x1 Graphite
  - 36 Carbon atoms
  - 144 electrons
- On Keeneland at NICS, each node has
  - Dual Hex-core X5560
  - 3 NVIDIA Fermi

Performance update

MC samples/(GPU,core)/sec



MC samples/(GPU,Node)/sec



MC samples/sec
= figure of merit for QMC

*4x4x1 graphite, 256 electrons

Computational challenges for QMC

QMC positioned to harness the increasing computing powers of current and next generation of HPC

- Sufficient parallelism over walkers on current HPC systems
  - Petaflop multi-core systems
  - Teraflop GPU systems
- A lot of new sciences on petaflop heterogeneous systems, including Titan

Reduce time per walker per DMC step: $O(N^2)$-$O(N^3)$

- Fine-level parallelisms: light-weight threads, nested tasks
- Optimizations on multi-core chips: random-access of read-only data, private/shared cache reuse on NUMA systems
- Utilizing all the power of heterogeneous nodes

Room for improvement

$\Psi_T(\mathbf{R}) = \Pi_k \Psi_k$

$\hat{H} = \sum_k \hat{h}_k$

for generation $= 1 \cdots N_{\mathrm{MC}}$ **do**
  for walker $= 1 \cdots N_w$ **do**
    let $\mathbf{R} = \{\mathbf{r}_1 \ldots \mathbf{r}_N\}$

node

    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 state of a walker
      end if
    end for{particle}
    Compute $E_L = \hat{H}\Psi_T(\mathbf{R})/\Psi_T(\mathbf{R})$
    Reweight and branch walkers
    Update E_T
  end for{walker}
end for{generation}

```
T Psi<T>::ratio(int i)
{
  T r(1.0);
  for (int k=0; k<Z.size(); ++k)
    r *= Z[k]->ratio(P,i);
  return r;
}
```

```
T Hamiltonian<T>::evaluate()
{
  T eloc=0.0;
  for(int k=0; k<H.size(); ++k)
    eloc += H[k]->evaluate(P);
  return eloc
}
```

Core Computations

For each walker,

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 state of a walker
  end if
end for{particle}
Compute $E_L = \hat{H}\Psi_T(\mathbf{R})/\Psi_T(\mathbf{R})$

All about Ψ_T

$$\delta = r + \tau \nabla_i \ln \Psi_T$$

Quantum force

$$\frac{\Psi_T(\mathbf{r}_1 \cdots \mathbf{r}_i^{'} \cdots \mathbf{r}_N)}{\Psi_T(\mathbf{r}_1 \cdots \mathbf{r}_i \cdots \mathbf{r}_N)}$$

$$\Psi_T \leftarrow \Psi_T(\mathbf{r}_1 \cdots \mathbf{r}_i^{'} \cdots \mathbf{r}_N)$$

$$f(\{\mathbf{R}\}, \nabla \ln \Psi_T, \nabla^2 \ln \Psi_T)$$

Use $\Psi_T = \Pi_i \Psi_i \longrightarrow \ln \Psi_T = \sum_i \ln \Psi_i$

Slater-Jastrow for Electrons

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

Correlation (Jastrow)

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

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

Anti-symmetric function (Pauli principle)

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

Single-particle orbitals

- Computational complexity per MC step
  - Evaluation $\{\phi\}$ — $\mathcal{O}(N^2 N_{spo})$
  - Determinant evaluation — $\mathcal{O}(N^3)$
  - Jastrow evaluation — $\mathcal{O}(N) - \mathcal{O}(N^3)$

Single-particle orbitals

- Linear combinations of basis functions

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

$$N_{spo} \propto N_b Op(\Phi)$$

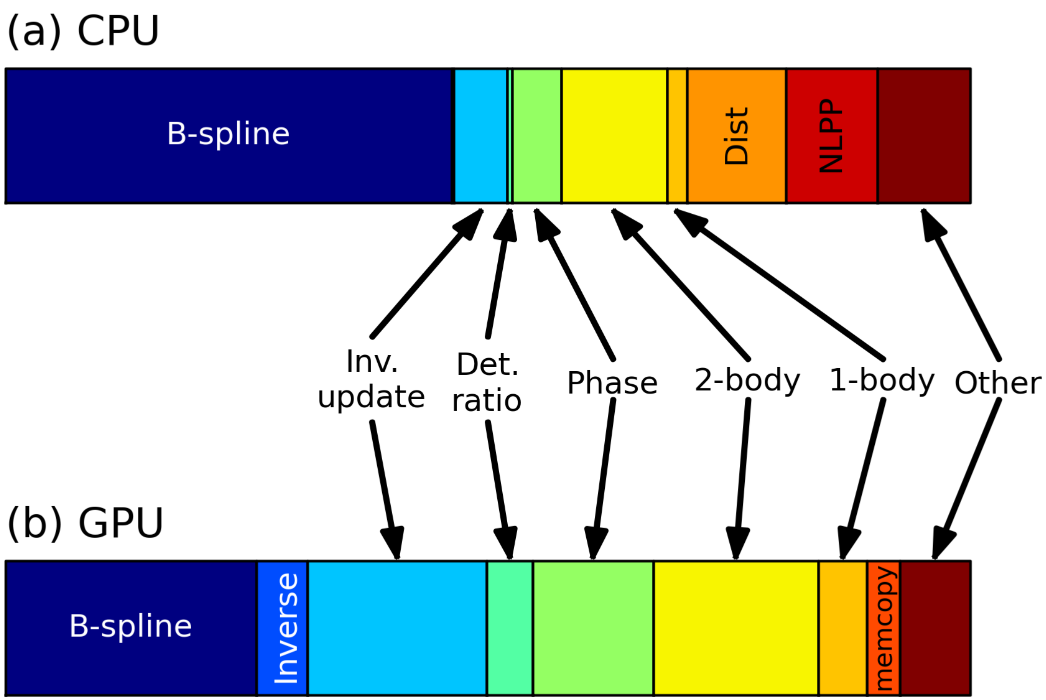
- Typically the solutions of simpler theories, i.e. $C's \ \& \{\Phi\}$ from Hartree-Fock or DFT calculations

- SPO can take various forms

SPO Type	N_b	$Op(\Phi)$	Memory Use
Molecular orbitals	$\mathcal{O}(N)$	Medium-High	Low
Plane waves	$\mathcal{O}(N)$	High	Medium
B-spline	Fixed	Low	High

Best solution for large-scale QMC on SMPs

Analysis on current CPU & GPU



Breakup of compute kernels

- QMCPACK achieves high efficiency by amortizing threads & memory
- As the system size and complexity grows, each kernel takes longer
- Can afford overhead for task-based parallelism
- But, difficult to balance the load among tasks: device and problem dependent

Strategy to further accelerate QMC

- Task-based parallelism with smart allocators on heterogeneous nodes
- Exploit generic programming
  - Specialization on devices: allocators, containers, algorithms
  - Hide low-level programming but optimize the kernels with the best option(s) available
  - Auto-tuning of SIMD kernels
- Stick to standards: C++, OpenMP, Pthreads and MPI
  - Heavy lifting by the compilers
  - Vendor optimized communication and numerical libraries
- Cope with the changes

Conclusions

- QMC has kept up with the HPC evolution and will continue improving predictive powers in physics, materials and chemistry
  - ✓ Clusters of multi- and many-core SMP
  - ✓ Clusters of GPU
  - 😰 Clusters of hybrid
  - 😱 What is next
- More to be done improve science productivity
  – Reduce impacts of application-level, software and hardware faults: Algorithms for robust and fault-tolerant simulations
  – Faster off-node communication and I/O

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)
- 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)