Architectural Comparisons for a Quantum Monte Carlo Application

Akila Gothandaraman, Rick Weber, Gregory D. Peterson, Robert J. Hinde, and Robert J. Harrison

{akila, fweber1, gdp, rhinde, robert.harrison}@utk.edu

Symposium on Application Accelerators in High Performance Computing
July 30, 2009
Introduction

- Supercomputing has surpassed petaflop milestone
  - RoadRunner (at LANL) - 1.105 petaflop
  - Jaguar (at ORNL) - 1.059 petaflop

But …

- Ever-increasing demand for CPU cycles in scientific computing
- We need to scale applications to utilize the performance
- Increased power, space, and cooling requirements
Introduction

Emerging architectures in the landscape of HPC

- Multi-core Processors
- Reconfigurable Computing (RC) and High Performance Reconfigurable Computing (HPRC) - Field-Programmable Gate Arrays (FPGAs)
- Graphics processing units for general-purpose computing - GP-GPUs

Exploit parallelism at different levels of granularity

Next generation supercomputers likely have one or more accelerators

<table>
<thead>
<tr>
<th>Multi-core processors</th>
<th>Reconfigurable Computing</th>
<th>Graphics processors</th>
</tr>
</thead>
<tbody>
<tr>
<td>PCI to HPRC transition, Floating-point or customized precision</td>
<td>Hardware Performance and Programmability, Single- and double-precision</td>
<td></td>
</tr>
</tbody>
</table>
Target Accelerators

- Reconfigurable Computing (RC)
  - FPGA-based solutions - Cray (XD1, XT5 \textsubscript{h}/XR1), Nallatech, SRC Computers, DRC Computers, XtremeData, Maxwell
  - VHDL/Verilog, Mitrion-C, DIME-C, Impulse-C
  - Cray XD1 HPRC - VHDL Programming

- Graphics Processing Units (GPUs)
  - GPU solutions - NVIDIA, AMD/ATI
  - CUDA, Brook+, CAL, OpenCL
  - NVIDIA Tesla - Compute Unified Device Architecture (CUDA)
  - AMD/ATI Firestream - Brook+
Quantum Monte Carlo

“Computational Science is often thought of as the third leg of science along with experimental and theoretical science” -Cornell Theory Center

☐ Two simulation approaches in physics and physical chemistry
  ■ Molecular Dynamics (MD), Quantum Monte Carlo (QMC)

☐ Quantum Monte Carlo
  ■ Solve the many-body Schrödinger equation
  ■ Variational Monte Carlo - Trial wave function that when optimized best approximates the exact wave function
  ■ QMC is compute bound. With increased computational power
    ☐ Accelerate the computationally intensive kernels
    ☐ Length- and time- scales
Quantum Monte Carlo

Step 1: Select a reference configuration, $R(x, y, z)$ at random

REPEAT (for $I$ iterations)

REPEAT for all configurations

Step 2: Obtain a new configuration, $R'$, by adding a small random displacement to all the atoms in the configuration, $R$.

Step 3: Compute the ground-state properties (energy, wavefunction) of the atoms in the current configuration, $R'$

Step 4: Accept or reject the present configuration using the ratio of the wave function values,

$$ p = \frac{\psi_T(R')^2}{\psi_T(R)} $$

If $p \geq 1$, $R'$ is accepted. If $p < 1$, if $p < \text{rand}()$ $R'$ is rejected, $R$ is retained

UNTIL finished (for all configurations)

UNTIL finished (for all iterations)
CPU Implementation

- Consider the interaction matrix
  - Each row represents pair-wise interactions of particles in that row
  - Lower (or upper) triangular matrix, diagonal elements are zeros
  - Reduction of the matrix (in-place) to yield total energy or wave function
- C implementation uses double-precision
- Optimized using Intel MKL library yielded some speedup
FPGA Implementation

- Two-region approach
- Transformation and binning schemes in each region
- Example, for potential energy - Exponential transform in Region I and logarithmic binning scheme in Region II
- Quadratic interpolation on the transformed function

Capabilities of the Framework
- Model system of helium atoms
- General-purpose and user-friendly
- Evaluate helium-helium wave function
- Allows us to calculate different forms of potential energy

\[ 0 \leq r_{ij}^2 < \sigma^2 \]
\[ r_{ij}^2 \geq \sigma^2 \]

Region I: \[ 0 \leq r_{ij}^2 < \sigma^2 \]
Region II: \[ r_{ij}^2 \geq \sigma^2 \]

Potential Energy vs. Distance
Reconfigurable Architecture

QMC Interface

Block RAM Interface

Register Interface

RT Client

Opteron

Block RAM interface

O(N)

QMC interface

PE C (CalcFunc

PE memory

Memory

Position memory

O(N)

O(N^2)

Req

Resp

Req

Resp

Reg

Reg

Reg

Decoder

Reg

access state

1

2

3

4

QMC application on Opteron
CUDA Partitioning

- $O(N)$ atom positions copied from host memory to device memory - CUDA runtime functions
- Execution Configuration: 1D grid of blocks and block of threads
- Subset of rows of the matrix ($N/T$) to each thread block
  - Each thread calculates the interactions between its atom and all other atoms
CUDA Implementation

- Different levels of memory hierarchy on the GPU
- 16 KB shared memory per block
  - A thread block loads positions of $T$ atoms from device memory to shared memory
- In-place reductions of row-wise function values on GPU to produce $O(N)$ results
- Final reductions on the CPU to produce $O(I)$ potential energy or wave function
Brook+ Implementation

Naïve
- Each partial result (for potential energy or wave function) stored in a matrix
- Reduction of matrices
- $O(N^2)$ memory, $O(N^2)$ parallelism

Optimized
- Row-wise energies and wave functions are reduced in-place and partial results stored in a 1D stream
- $O(N)$ memory and $O(N)$ parallelism
Target Platform Details

- **Multi-core:** Eight core Intel Xeon Clovertown 2.66 GHz (8MB L2 cache)

- **Target HPRC:** Pacific Cray XD1
  - Single chassis system with 6 nodes
  - Each node - 2.2 GHz AMD Opteron dual-core dual-processor
  - Xilinx Virtex-II Pro (XC2VP50) or Xilinx Virtex-4 (XC4VLX160) FPGA
  - Processor and FPGA communication - RapidArray interconnect (1.6 GB/sec)
  - Development language and tools - VHDL, Xilinx 8.1 ISE/EDK

- **Target GPUs:**
  - AMD/ATI Firestream 9170 (320 cores, ~500 GFlops/sec - single)
  - NVIDIA C1060 Tesla (240 cores, ~933 Gflops/sec - single)
Results

- **Baseline Software**: QMC, C (*double-precision*), Eight cores of Intel Xeon Clovertown 2.66 GHz, gcc, -03 optimization

![Graph showing speedup vs. number of atoms for different implementations: Brook+-single, CUDA-double, CUDA-mixed, FPGA-fixed, CUDA-double. The x-axis represents the number of atoms, and the y-axis represents speedup. The graph shows Brook+-single with the highest speedup and FPGA-fixed with the lowest speedup.](image-url)
Results

Note:

- CUDA double-precision - best error performance (1.6555x10^{-12})
- CUDA single-precision - worst error performance (1.34x10^{-5})
Future Work

- Tuning wave function parameters using low-precision and compute exact energies using a higher precision

- Optimizing the Intel MKL QMC implementation and OpenCL implementation

- Run multiple walkers in float4 type on AMD card
  - Try doing the same thing in MKL to improve x86 implementation

- Performance modeling for efficient mapping on hybrid-computing platforms
Conclusions

- Explore individual computing platforms - Field-Programmable Gate Arrays (FPGAs), Graphics Processing Units (GPUs)

- FPGA Implementation
  - Fixed-point representation for all calculations on FPGA
  - 4x over optimized QMC software on eight cores of Intel Xeon 2.66 GHz (for a cluster with 4096 atoms)

- NVIDIA vs. AMD/ATI Implementations
  - Experimented with single-, double- and mixed- precisions on the Tesla GPUs
  - 19x and 16x speedup over optimized QMC software on eight cores of Intel Xeon 2.66 GHz (for a cluster with 4096 atoms)

- Choice of a platform - tradeoffs (performance, numerical accuracy)
  - Other Factors: cost, power, programming effort
Thank You
Backup slides
GPU vs. FPGA (*HFDB Potential*)

- CUDA double-precision has the best error performance
- FPGA fixed-point, and GPU mixed-precision are comparable (relative to CPU double-precision)

- Virtex-4 is also an older FPGA compared to C1060
- FPGA speedup limited by capacity

**Note:** FPGAs operate at clock frequencies an order of magnitude lower!

Relative error, pair potential ($x10^{-8}$) vs. number of atoms

**Speedup vs. number of atoms**

- CUDA double-precision has the best error performance
- FPGA fixed-point, and GPU mixed-precision are comparable (relative to CPU double-precision)

**Note:** FPGAs operate at clock frequencies an order of magnitude lower!
Error analysis

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Formats (s-signed, u-unsigned)</th>
</tr>
</thead>
<tbody>
<tr>
<td>(x, y, z) co-ordinate positions</td>
<td>s12.20</td>
</tr>
<tr>
<td>Squared distances</td>
<td>u27.26</td>
</tr>
<tr>
<td>Potential energy, wavefunction</td>
<td>s0.51</td>
</tr>
<tr>
<td>Interpolation coefficients</td>
<td>s0.51</td>
</tr>
</tbody>
</table>

- Fixed-point representation accurately reproduces the floating-point results almost over the entire range.
- Worst error performance when $r^2 > 2^{16}$
- Distances are confined to be within $2^{16}$ for our application.

No compromise in accuracy for our chemistry application.

Absolute error of potential

\[
\log_2(r^2) \quad \text{and} \quad \log_2(\text{Absolute error})
\]
Results - FPGA Implementation

- Target HPRC Platform: *Pacific* Cray XD1
  - Single chassis system with 6 nodes
  - Each node - 2.2 GHz AMD Opteron dual-core dual-processor
  - Xilinx Virtex-II Pro (XC2VP50) or Xilinx Virtex-4 (XC4VLX160) FPGA
  - Processor and FPGA communication - RapidArray interconnect (1.6 GB/sec)
- Development language and tools - VHDL, Xilinx 8.1 ISE/EDK

<table>
<thead>
<tr>
<th>Kernel / Resource type</th>
<th>PE</th>
<th>PE and WF</th>
</tr>
</thead>
<tbody>
<tr>
<td>SLICEs (67,854)</td>
<td>17,984 (26%)</td>
<td>30,432 (45%)</td>
</tr>
<tr>
<td>Block RAMs (288)</td>
<td>79 (27%)</td>
<td>147 (51%)</td>
</tr>
<tr>
<td>DSP48s (96)</td>
<td>25 (26%)</td>
<td>50 (52%)</td>
</tr>
</tbody>
</table>

PE - Potential Energy
WF - Wave Function

PE and WF on the same FPGA
Pipelined Reconfigurable Architecture

Datapath of CalcDist module

- **CalcDist**: 32-bit fixed-point format for positions and a 53-bit squared distance per clock cycle
- **CalcFunc**: Output a final 52-bit fixed-point potential energy/wavefunction every clock cycle
- **AccFunc**: Accumulates PE values as running products in region I and running sums in region II, WF values as running products in region I and II
- **Host processor reconstructs the floating-point values of potential energy and wavefunction**

Datapath of CalcFunc module

- **CalcDist**: 32-bit fixed-point format for positions and a 53-bit squared distance per clock cycle
- **CalcFunc**: Output a final 52-bit fixed-point potential energy/wavefunction every clock cycle
- **AccFunc**: Accumulates PE values as running products in region I and running sums in region II, WF values as running products in region I and II
- **Host processor reconstructs the floating-point values of potential energy and wavefunction**
**Exponential transform in Region I**

N-atom potential energy, \( V_{total} \approx \sum_{i<j}^N V(r_{ij}) \) \hspace{1cm} (1)

\[ V_{total} = V_I + V_{II} = \sum_{(i<j)\in I} V_I(r_{ij}) + \sum_{(i<j)\in II} V_{II}(r_{ij}) \] \hspace{1cm} (2)

\[ V'_I(r_{ij}^2) = e^{-V_I(r_{ij}^2)} \] \hspace{1cm} (3) \hspace{1cm} \boxed{\text{Transformed potential, } V'_I \text{ is restricted to take values between zero and 1}}

\[ V'_I = \prod_{(i<j)\in I} V'_I(r_{ij}) \] \hspace{1cm} (4) \hspace{1cm} \boxed{\text{First term in Eq. 2 is now expressed as product of transformed pair-wise potentials (Eq. 4)}}

\[ -\ln V'_I = V_I = \sum_{(i<j)\in I} V_I(r_{ij}) \] \hspace{1cm} (5) \hspace{1cm} \boxed{\text{Final transformation, to obtain the natural logarithm can be delegated to the host processor}}
Binning schemes

**Region I – Bin lookup**

- Region I divided into 256 bins
- Region II divided into 21 sub-regions or regimes
- Each sub-region accommodates 64 bins
- Total of \(256 + 21 \times 64\) \times 3 interpolation coefficients → fa, fb, fc

**Region II – pseudo logarithmic binning**

**Region II – 1\textsuperscript{st} stage Bin lookup**

**Region II – 2\textsuperscript{nd} stage Bin lookup**
Variational Principle

The expectation value of a Hamiltonian, $\hat{H}$, calculated using a trial wave function, $\psi_T$, is never lower than the true ground-state energy, $\varepsilon_o$, which is the expectation value of $\hat{H}$ calculated using the true ground state wave function, $\psi_o$.

- Upper bound for ground-state energy
- Trial wave function as a “guess” of the true ground state wave function to be used as input wave function

\[
< E > = \int dR \psi H \psi / \int dR \psi^2
\]

\[
< E >= \int dR \frac{1}{P(R)} H \psi
\]

\[
P(R) = \frac{\psi^2}{\int dR \psi^2}
\]

\[
H \psi = E \psi
\]

\[
\psi_i \sum_{i=1}^{N} \nabla_i^2 + \sum_{i<j} V(r_{ij})
\]
Target platform

- Pacific Cray XD1 HPRC platform - Single chassis system with 6 nodes
  - Each node - 2.2 GHz AMD Opteron dual-core dual-processor, Xilinx Virtex-II Pro (XC2VP50) or Xilinx Virtex-4 (XC4VLX160) FPGA
  - Processor and FPGA communication - RapidArray interconnect

Development language and tools

- VHDL, Synplicity Synplify tools, Xilinx 8.1 ISE/EDK

Resource usage of QMC kernels (Virtex-4 FPGA)

<table>
<thead>
<tr>
<th>Kernel / Resource type</th>
<th>PE</th>
<th>PE and WF</th>
</tr>
</thead>
<tbody>
<tr>
<td>SLICEs (67,854)</td>
<td>17,984 (26%)</td>
<td>30,432 (45%)</td>
</tr>
<tr>
<td>Block RAMs (288)</td>
<td>79 (27%)</td>
<td>147 (51%)</td>
</tr>
<tr>
<td>DSP48s (96)</td>
<td>25 (26%)</td>
<td>50 (52%)</td>
</tr>
</tbody>
</table>

PE and WF on the same FPGA

PE - Potential Energy
WF - Wave Function