Lattice QCD software for the IBM Blue Gene/P architecture

Stefan Krieg
<s.krieg@fz-juelich.de>

Jülich Supercomputing Center,
Forschungszentrum Jülich
and
FB C, Mathematik und Naturwissenschaften,
Bergische Universität Wuppertal

Joint Seminar on High Performance Computing
Trinity College, Dublin - Bergische Universität Wuppertal
Outline

Lattice QCD
  LQCD basics
  Simulation algorithm

Blue Gene/P hardware
  BGP system
  BGP functionality

Lattice QCD software for Blue Gene
  The Wilson kernel
  Communication
  Serial code

Performance results
  Kernel Performance
  Mixed prec. inverts

Conclusion and outlook
Outline

Lattice QCD
  LQCD basics
  Simulation algorithm

Blue Gene/P hardware
  BGP system
  BGP functionality

Lattice QCD software for Blue Gene
  The Wilson kernel
  Communication
  Serial code

Performance results
  Kernel Performance
  Mixed prec. inverters

Conclusion and outlook
Outline

Lattice QCD
  LQCD basics
    Simulation algorithm

Blue Gene/P hardware
  BGP system
  BGP functionality

Lattice QCD software for Blue Gene
  The Wilson kernel
  Communication
  Serial code

Performance results
  Kernel Performance
  Mixed prec. inverters

Conclusion and outlook
The lattice

Lattice QCD (LQCD) is defined on a 4 dim. periodic lattice; LQCD is a way to define QCD in a mathematically precise way.

Key ingredients are:

- *The Quarks* living on the lattice sites
- *The Gluons* living on the lattice links
- Typically the LQCD *action* connects only neighboring sites (plain Wilson)

Simulations of LQCD are the only available method to directly access the low energy regime of QCD.
QCD particle spectrum (Blue Gene, BMW coll.)

\[ \text{Science 322, 1224 (2008)} \]
Outline

Lattice QCD

  LQCD basics
  Simulation algorithm

Blue Gene/P hardware

  BGP system
  BGP functionality

Lattice QCD software for Blue Gene

  The Wilson kernel
  Communication
  Serial code

Performance results

  Kernel Performance
  Mixed prec. inverters

Conclusion and outlook
Importance Sampling

- After Wick-rotation, the path-integral becomes equivalent to a partition function of statistical mechanics

\[
\langle O \rangle_{QCD}^{F,G} = \int \mathcal{D}\overline{\psi}\mathcal{D}\psi\mathcal{D}A \, O(\overline{\psi}, \psi, A) \exp\left\{-S_{QCD}^{\mathcal{E}}(\overline{\psi}, \psi, A)\right\}
\]

\[
= \int \mathcal{D}A \det[D] \langle O(A) \rangle^{F} \exp\left\{-S_{E,G}^{QCD}(A)\right\}
\]

- The Boltzmann-weight can be interpreted as probability
- if \(A_i\) has the probability \(p \propto \exp\{-S_{E}^{QCD}(A_i)\}\) to appear in the ensemble of \(A\)’s then

\[
\langle O \rangle_{QCD}^{F,G} = \frac{1}{N} \sum_{i=0}^{N} \langle O(A_i) \rangle^{F},
\]

gives a statistical estimate of an operator expectation value
Simulation algorithm: HMC

The fermionic determinant

\[ \int \prod_i d\eta_i d\eta_i^\dagger \exp\left\{ -\sum_{ij} \eta_i^\dagger D_{ij} \eta_j \right\} = \text{det}[D] \]

is included via the pseudofermions (pos. definite $K$):

\[ \int \prod_i dc_i dc_i^\ast \exp\left\{ -\sum_{ij} c_i^\ast K_{ij} c_j \right\} = \frac{1}{\text{det}[K]} \]

The Hybrid Monte Carlo algorithm now proceeds as follows

1. Generate a pseudofermion vector
2. Molecular Dynamics evolution of the gauge links (⇒ Inversions)
3. Metropolis accept reject step (energy: $E = S + \frac{1}{2} \Pi^2$)
Outline

Lattice QCD
  LQCD basics
  Simulation algorithm

Blue Gene/P hardware
  BGP system
  BGP functionality

Lattice QCD software for Blue Gene
  The Wilson kernel
  Communication
  Serial code

Performance results
  Kernel Performance
  Mixed prec. inverters

Conclusion and outlook
Outline

Lattice QCD
  LQCD basics
  Simulation algorithm

Blue Gene/P hardware
  BGP system
  BGP functionality

Lattice QCD software for Blue Gene
  The Wilson kernel
  Communication
  Serial code

Performance results
  Kernel Performance
  Mixed prec. inverters

Conclusion and outlook
Blue Gene/P system structure

System
72 Racks, 72x32x32
1 PF/s, 144 TB

Rack
32 Node Cards
Cabled 8x8x16
13.9 TF/s, 2 TB

Node Card
(32 chips 4x4x2)
32 compute, 0-1 IO cards
435 GF/s, 64 GB

Chip
4 processors
13.6 GF/s

Compute Card
1 chip, 13.6 GF/s
2.0 GB DDR2
(4.0GB optional)
Blue Gene/P networks

Blue Gene/P main networks are:

- The 3 dimensional torus network
  - Bandwidth (node): 6*2*425MB/s = 5.1GB/s
  - HW latency (1 hop): 100ns (32B), 800ns (256B packet)
  - HW latency (worst): 3.2μs
  - DMA controlled: overlapping communication and computation

- The tree network for collectives
  - Bandwidth (node): 2*850MB/s = 1.7GB/s
  - HW latency (worst): 3.5μs

- Global barrier network
Outline

Lattice QCD
  LQCD basics
  Simulation algorithm

Blue Gene/P hardware
  BGP system
  BGP functionality

Lattice QCD software for Blue Gene
  The Wilson kernel
  Communication
  Serial code

Performance results
  Kernel Performance
  Mixed prec. inverters

Conclusion and outlook
DMA communications

DMA is capable of:

- Direct-put: put data into memory on destination node
- MemFifo comms: put data into reception fifo on destination node
- Remote-get: put a descriptor into injection fifo on destination node
- Prefetch-only: prefetch data into L3 (no transfers)
- Destination node can be the node itself (local transfers)

DMA “directly” programmable: SPI
“Double Hummer” FPU

The “Double Hummer” FPU features:

- Instructions optimized for complex arithmetic: only 2 instructions required for complex multiplication
- 32 primary + 32 secondary registers
- Capability to load 16 Byte quadword
- 5 stage pipeline
Complex multiplication with “Double Hummer” FPU

Complex multiplication \((A \times B = C)\)

\[
\begin{align*}
\text{Re}(C) &= (\text{Re}(A)\text{Re}(B) - \text{Im}(A)\text{Im}(B)) \\
\text{Im}(C) &= (\text{Im}(A)\text{Re}(B) + \text{Re}(A)\text{Im}(B))
\end{align*}
\]

Required:

- a cross copy primary multiply (2+1 register operands)
- a cross mixed negative secondary multiply-add (3+1 register operands)
Outline

Lattice QCD
- LQCD basics
- Simulation algorithm

Blue Gene/P hardware
- BGP system
- BGP functionality

Lattice QCD software for Blue Gene
- The Wilson kernel
- Communication
- Serial code

Performance results
- Kernel Performance
- Mixed prec. inverters

Conclusion and outlook

Stefan Krieg
30.01.2009
BlueGene software for LQCD
Outline

Lattice QCD
  LQCD basics
  Simulation algorithm

Blue Gene/P hardware
  BGP system
  BGP functionality

Lattice QCD software for Blue Gene
  The Wilson kernel
  Communication
  Serial code

Performance results
  Kernel Performance
  Mixed prec. inverters

Conclusion and outlook
Wilson kernel properties

- Wilson kernel is sparse matrix vector multiplication
- Sparse: memory footprint of kernel scales linearly with $N$
- Wilson kernel is a first order derivative connecting NN
- However “NN coefficients” are random matrices

Blue Gene/P implementation strategy:

1. (scalar) Spin project forward
2. (comm) Start communication forward
3. (scalar) Spin project backward and SU(3) multiply
4. (comm) Wait forward; Start communication backward
5. (scalar) SU(3) multiply fwd. and sum up
6. (comm) Wait backward
7. (scalar) Add backward

⇒ Calculations and communications overlap.
The Wilson kernel

Wilson kernel communication pattern

Match 4 dim. periodic physics lattice to communication hardware:

- Put 3 dimensions along torus directions
- Use local transfers for 4th dimension:
  
  Core0 → Core1
  
  Core2 → Core3

⇒ 4 dimensional torus
Outline

Lattice QCD
- LQCD basics
- Simulation algorithm

Blue Gene/P hardware
- BGP system
- BGP functionality

Lattice QCD software for Blue Gene
- The Wilson kernel
- Communication
- Serial code

Performance results
- Kernel Performance
- Mixed prec. inverters

Conclusion and outlook
Using the DMA

Fifo is a contiguous chunk of DDR memory

The Fifo has
- Starting address
- Ending address
- Head
- Tail

Message descriptors are injected into injection fifo

DMA “executes” a descriptor, then updates fifo head

New descriptor is injected at fifo tail, tail is then updated

Tail is wrapped around
Counters

Two types of counters:

- **Reception (rDMA) counters**
  - *Base address*: associated virtual address in main memory; incoming data is stored relative to this address
  - *Max address*: associated virtual address in main memory; defines “memory reception window”
  - *Counter value*: is decreased by DMA when receiving a message associated to the counter

- **Injection (iDMA) counters**
  - *Base address*: associated virtual address in main memory; data is send relative to this address
  - *Counter value*: is decreased by DMA when sending a message associated to the counter
Descriptors

A message descriptor contains (direct-put)

- Coordinates of the destination node (nonlocal comm.)
- Send offset relative to inj. counter base address
- Injection counter group and ID
- Reception offset relative to rec. counter base address (on destination node)
- Reception counter group and ID (on destination node)
- Message length
SPI software layer

SPI contains low level function calls to

- access and manipulate fifos, counters etc.
- create descriptors, inject descriptors
- use the global interrupt and collective network
Counters and descriptors

For e.g. a “direct-put“ using SPI one could proceed as follows:

1. Allocate 1 injection fifo, 1 reception and 1 injection counter (if you do not have them already)
2. Set the injection counter base address e.g. to the smallest virtual address of the data that you want to communicate
3. Set the injection counter value to the number of bytes you intend to send
4. (destination node) send the reception counter base address to the smallest virtual address where you want to store received data
5. Set the reception counter max address
6. Set the reception counter value to the number of bytes send

For each continuous chunk of data, inject 1 descriptor into the inj. fifo

1. Calculate the address offset relative to the injection counter base
2. Calculate the address offset relative to the reception counter base
3. Give the message size
Summary: kernel persistent communication

Setting up the persistent communication for the kernel with SPI

1. Allocate 6+2 injection fifos, 2 reception and 2 injection counters
2. Set the reception counter base and max and the injection counter base addresses (same on every node)
3. Deactivate fifos and counters.
4. Calculate the offsets and message sizes etc. (block-strided-move) and inject the descriptors
5. Move each fifo head to the fifo tail
6. Activate fifos and counters

To start the communication

1. Set the reception and injection counter values
2. Move the fifo heads to the fifo start

And the DMA does its work in the background.
Poll counters to make sure communication is completed.
Outline

Lattice QCD
- LQCD basics
- Simulation algorithm

Blue Gene/P hardware
- BGP system
- BGP functionality

Lattice QCD software for Blue Gene
- The Wilson kernel
- Communication
- Serial code

Performance results
- Kernel Performance
- Mixed prec. inverters

Conclusion and outlook
Problems

Making efficient use of the “Double Hummer” FPU is not easy:

- Secondary FPU and secondary registers only accessible through oedipus instructions
- Oedipus instructions always work on 16 Byte aligned 16 Byte quadwords = 2 double precision floating point numbers → User has to make sure data is aligned
- Compilers frequently fail to rearrange code properly → “Simdization” fails.

⇒ To get maximum performance one needs to go low level
Fast and easy: XL intrinsics

- IBM XL compilers provide built-in functions (intrinsics) that map to (e.g. floating point) assembly instructions, e.g. `__lfpd`, `__stfpd`, `__fpmadd` and `__dcbt`.
- Intrinsics operate on "double _Complex" variables that map to registers.
- Number of variables is not limited (like registers).
- Scheduling will be done by compiler.

⇒ Comparatively easy to use.

- LQCD code has large parts optimize using intrinsics.
- Use a set of macros that implement basic mathematical operations.
- With macros optimization is almost trivial.
Example: daxpy

![Graph showing performance of daxpy with vector length in KBytes on Blue Gene/P.](image-url)
Example: caxpy
Faster and annoying: assembly

When using the intrinsics, the compiler has great influence on the performance.

- For more control use (gcc inline) assembly
- Kernel serial code written in assembly
- Uses explicit prefetches (dcbt)
- All scheduling and register allocation done by hand
- No problem if all that is already there (BGL kernel)

⇒ Performance typically another 10% better compared to intrinsics

⇒ Code generation typically 10 times slower compared to intrinsics
Outline

Lattice QCD
  LQCD basics
  Simulation algorithm

Blue Gene/P hardware
  BGP system
  BGP functionality

Lattice QCD software for Blue Gene
  The Wilson kernel
  Communication
  Serial code

Performance results
  Kernel Performance
  Mixed prec. inverters

Conclusion and outlook
Outline

Lattice QCD
  LQCD basics
  Simulation algorithm

Blue Gene/P hardware
  BGP system
  BGP functionality

Lattice QCD software for Blue Gene
  The Wilson kernel
  Communication
  Serial code

Performance results
  Kernel Performance
  Mixed prec. inverters

Conclusion and outlook
Wilson kernel performance (strong scaling)

Wilson kernel shows:
- almost perfect strong scaling,
- a large scaling range,
- perfect weak scaling,
- and reaches 37.5% of absolute peak
Performance comparison BGL/BGP

- Optimized comm. + calc.
- Large scaling region

Stefan Krieg
30.01.2009
BlueGene software for LQCD
FZ Jülich, BU Wuppertal
Outline

Lattice QCD
  LQCD basics
  Simulation algorithm

Blue Gene/P hardware
  BGP system
  BGP functionality

Lattice QCD software for Blue Gene
  The Wilson kernel
  Communication
  Serial code

Performance results
  Kernel Performance
  Mixed prec. inverters

Conclusion and outlook
Mixed precision via iterative refinement

- Single precision kernel has best performance and largest scaling region
- Valence and sea sector calculations require double precision accuracy
- Solution: Mixed precision inverters
  Use a less precise $A^{-1} = a^{-1}$:
  1. Compute $b - Ax_0 = r$
  2. Compute $a^{-1}r = A^{-1}r + r_s$, with $|r_s| = \epsilon |r|$
  3. Update $x$: $x_1 = x_0 + a^{-1}r$
  \[ |b - Ax_1| = |b - Ax_0 - Aa^{-1}r| = |r - AA^{-1}r - r_s| = \epsilon |r| \]
- This can be used to speed up a generic CG
- Precision of $a^{-1}$ can be tuned
Mixed precision GMRESR

\{computes $x$ with $\|Ax - b\| \leq \epsilon \cdot \|b\|$ via relaxed GMRESR\}

$x = 0$; \quad \{\text{initial value}\}
$r = b$; \quad \{\text{empty matrix}\}
while $\|r\| > \epsilon \cdot \|b\|$ do
    solve $Au = r$ to relative accuracy $\xi$ \{preconditioner\}
    compute $c$ with $\|Au - c\| \leq \epsilon \cdot \|b\| \cdot \|u\|/\|r\|$
    for $i=1:$size$(C,2)$ do
        $\beta = C[:, i]^\dagger \cdot c$
        $c = c - \beta \cdot C[:, i]$
        $u = u - \beta \cdot U[:, i]$
    end for
    $c = c/\|c\|; \quad u = u/\|c\|$
    $C = [C, c]; \quad U = [U, u]$
    $\alpha = c^\dagger \cdot r$
    $x = x + \alpha \cdot u$
    $r = r - \alpha \cdot c$
end while
Mixed precision inverter performance

- **Wilson CG-64 vs. CG-mix**
- **overlap CG-64 vs. GMRES-mix**

Graphs showing performance over time with different scales for error and time.
Outline

Lattice QCD
  LQCD basics
  Simulation algorithm

Blue Gene/P hardware
  BGP system
  BGP functionality

Lattice QCD software for Blue Gene
  The Wilson kernel
  Communication
  Serial code

Performance results
  Kernel Performance
  Mixed prec. inverters

Conclusion and outlook
Conclusion and Outlook: LQCD software

Blue Gene/L and Blue Gene/P low level software layer
► is the core of all the simulation codes used within BMW projects on Blue Gene
► contains all performance critical routines
► extends, combined with mixed action inverters, the scaling region of the code to over ten thousands of CPUs
► delivers up to 37.5% sustained performance
► has been successfully used within the LQCD simulations to test and verify the Blue Gene’s at Jülich

In the future
► further optimize software layer
► include support for CELL BE
Excuse me? THEORETIC? Surely you must be joking... Boeing!