SciDAC Software Infrastructure for Lattice Gauge Theory

Jefferson Laboratory --- February 1, 2002

Software Co-ordinating Committee

• Rich Brower --- Boston University
• Carleton DeTar --- University of Utah
• Robert Edwards --- Jefferson Laboratory
• Don Holmgren --- Fermi National Laboratory
• Bob Mawhinney --- Columbia University/BNL
• Celso Mendes --- University of Illinois
• Chip Watson --- Jefferson Laboratory
SciDAC Software Infrastructure Goals

- Create a unified programming environment that will enable the US lattice community to achieve very high efficiency on diverse multi-terascale hardware

Major Software Tasks

I. QCD API and Code Library
II. Optimize Network Communication
III. Optimize Lattice QCD Kernels
IV. Application Porting and Optimization
V. Data Management and Documentation
VI. Execution Environment
<table>
<thead>
<tr>
<th>Name</th>
<th>Institution</th>
<th>Percentage</th>
<th>Start Date</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bob Mawhinney</td>
<td>Columbia</td>
<td>----</td>
<td></td>
</tr>
<tr>
<td>Chulwoo Jung</td>
<td>BNL</td>
<td>100%</td>
<td>Sept 1, 2001</td>
</tr>
<tr>
<td>Chris Miller</td>
<td>BNL</td>
<td>100%</td>
<td></td>
</tr>
<tr>
<td>New hire</td>
<td>BNL</td>
<td>100%</td>
<td>(March 1, 2002)</td>
</tr>
<tr>
<td>Don Holmgren</td>
<td>FNAL</td>
<td>40%</td>
<td></td>
</tr>
<tr>
<td>Ron Rechenmacher</td>
<td>FNAL</td>
<td>25%</td>
<td></td>
</tr>
<tr>
<td>Jim Simone</td>
<td>FNAL</td>
<td>35%</td>
<td></td>
</tr>
<tr>
<td>Simon Epsteyn</td>
<td>FNAL</td>
<td>10%</td>
<td></td>
</tr>
<tr>
<td>New hire</td>
<td>FNAL</td>
<td>100%</td>
<td>(Feb 1, 2002)</td>
</tr>
<tr>
<td>Steve Gottlieb</td>
<td>FNAL</td>
<td>10%</td>
<td>Oct 15, 2001</td>
</tr>
<tr>
<td>Daniel A. Reed</td>
<td>Illinois</td>
<td>----</td>
<td></td>
</tr>
<tr>
<td>Celso L. Mendes</td>
<td>Illinois</td>
<td>35%</td>
<td>Oct 1, 2001</td>
</tr>
<tr>
<td>Name</td>
<td>Institution</td>
<td>Percentage</td>
<td>Date/Note</td>
</tr>
<tr>
<td>-----------------------</td>
<td>-------------</td>
<td>------------</td>
<td>----------------------------</td>
</tr>
<tr>
<td>Robert Edwards</td>
<td>Jlab</td>
<td>----</td>
<td></td>
</tr>
<tr>
<td>Chip Watson</td>
<td>Jlab</td>
<td>33%</td>
<td></td>
</tr>
<tr>
<td>Chris McClendon</td>
<td>Jlab</td>
<td>100%</td>
<td>(Summer 01)</td>
</tr>
<tr>
<td>Jie Chen</td>
<td>Jlab</td>
<td>100%</td>
<td>Jan 1, 2002</td>
</tr>
<tr>
<td>Patrick Dreher</td>
<td>MIT</td>
<td>----</td>
<td></td>
</tr>
<tr>
<td>Andrew Pochinsky</td>
<td>MIT</td>
<td>100%</td>
<td></td>
</tr>
<tr>
<td>Richard Brower</td>
<td>BU</td>
<td>30%</td>
<td></td>
</tr>
<tr>
<td>Massimo Di Pierro</td>
<td>BU</td>
<td>100%</td>
<td>(Oct 1, 2002)</td>
</tr>
<tr>
<td>Carleton DeTar</td>
<td>Utah</td>
<td>----</td>
<td></td>
</tr>
<tr>
<td>James Osborn</td>
<td>Utah</td>
<td>50%</td>
<td>Sept 1, 2001</td>
</tr>
<tr>
<td>Doug Toussaint</td>
<td>Arizona</td>
<td>----</td>
<td></td>
</tr>
<tr>
<td>Eric Gregory</td>
<td>Arizona</td>
<td>50%</td>
<td>Oct 15, 2001</td>
</tr>
</tbody>
</table>
Lattice QCD – extremely uniform

Dirac operator

\[ D\psi(x) = \sum_{\mu} \left( \partial_\mu + igA_\mu(x) \right)\psi(x) \]

Discretize on lattice – finite difference

\[ D\psi(x) = \frac{1}{2a} \sum_{\mu} \left[ U(x)\psi(x+\hat{\mu}) - U^\dagger(x-\hat{\mu})\psi(x-\hat{\mu}) \right] \]

- Periodic or very simple boundary conditions
- SPMD: Identical sublattices per processor
# QCD-API Level Structure

<table>
<thead>
<tr>
<th>Level 3</th>
</tr>
</thead>
<tbody>
<tr>
<td>Dirac Operators, CG Routines etc.</td>
</tr>
<tr>
<td>C, C++, etc.</td>
</tr>
<tr>
<td>(Organized by MILC or SZIN or CPS etc.)</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Level 2</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>QDP_XXX</strong></td>
</tr>
<tr>
<td>Data Parallel QCD Lattice Operations (overlapping Algebra and Messaging)</td>
</tr>
<tr>
<td>A = SHIFT(B, mu) * C; Global sums, etc</td>
</tr>
<tr>
<td>Lattice Wide Linear Algebra (No Communication)</td>
</tr>
<tr>
<td>e.g. A = B * C</td>
</tr>
<tr>
<td>Lattice Wide Data Movement (Pure Communication, non-blocking)</td>
</tr>
<tr>
<td>e.g Atemp = SHIFT(A, mu)</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Level 1</th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>QLA_XXX</strong></td>
<td>Single Site Linear Algebra API</td>
<td><strong>QMP_XXX</strong></td>
</tr>
<tr>
<td>SU(3), gamma algebra etc.</td>
<td>Message Passing API (Know about mapping of Lattice onto Network Geometry)</td>
<td></td>
</tr>
</tbody>
</table>
I. Design & Documentation of QCD-API

- **Major Focus** of Software Co-ordinating Committee
  - See agenda and minutes on http://physics.bu.edu/~brower/SciDAC
  - Published documents to appear on http://www.lqcd.org

- **First design workshop**, Jlab: Nov. 8-9, 2001

- **Second design workshop**, Jlab: Feb 2, noon-midnight, 2002
  - finalize level 1 API and resolve outstanding questions for level 2 data parallel framework.

- **Goal:**
  - Distribute documentation and a C implementation for community review by Lattice 2002 in Boston, MA.
  - Foster “Linux style” contributions to level 3 API library functions?
Data-Parallel Programming Model

- Basic uniform operations across lattice: \( C(x) = A(x) \times B(x) \)
- Map problem grid onto machine grid.
- API should hide subgrid layout and subgrid faces communicated between nodes.
- Like Fortran 90 – place Data Parallel paradigm on top of message passing.
- Implement API without writing a compiler.
API Design Criteria

- **Routines are extern C functions** callable from C and Fortran: *extern functions <=> C++ methods.*
- **Overlapping** of computation and communications.
- **Hide data layout:** Constructor, destructors. Query routines to support limited number of deftypes.
- **Support for multi-process** or multi-threaded computations hidden from user control.
- **Functions** do not (by default) make conversions of arguments from one layout into another layout. An error is generated if arguments are in incompatible.
II. Level 1 MP-API implementation

- **Definition of MP interface** (Edwards, Watson)
  - Bindings for C, C++ and eventually Fortran.
  - see doc http://www.jlab.org/~watson/lqcd/MessageAPI.html

- **Implementation of MP-API over MPI subset** (Edwards)

- **Implementation of C++ MP-API for QCDOC** (Jung)

- **Myrinet optimization using GM** (Jie Chen)

- **Port of MILC code to level 1 MP-API** (DeTar, Osborn)
  - Status 80% complete pending finalization on Feb 2
Performance Considerations for Level 2

- Predefined channels are opened between neighboring nodes of a d dimensional grid.

- Overlapping communications and computations, e.g. $C(x) = A(x) \times \text{shift}(B, \mu)$:
  - The face of a subgrid is sent non-blocking to a neighboring node, e.g. in the forward direction.
  - The neighboring node, in the backward direction, sends its face into a preallocated buffer.
  - While this is going on, the specified operation is performed on the interior sites.
  - A “wait” is issued and the operation is performed on the face.

Data layout over processors
Lazy Evaluation for Overlapping Comm/Comp

Consider the equation \( \text{dest}(x) = \text{src1}(x) \times \text{src2}(x+\text{nu}) \); (for all \( x \))

or decomposed as

\[
\text{tmp}(x) = \text{src2}(x+\mu);
\]

\[
\text{dest}(x) = \text{src1}(x) \times \text{tmp}(x)
\]

**Implementation 1:** This can be implemented as two functions as follows:

\[
\text{Shift}(\text{tmp}, \text{src2}, \mu, \text{plus});
\]

\[
\text{Multiply}(\text{dest}, \text{src1}, \text{tmp});
\]

**Implementation 2:** If the Shift is changed to also return its result, this can be rewritten as:

\[
\text{Multiply}(\text{dest}, \text{src1}, \text{Shift}(\text{src2}, \mu, \text{plus}));
\]
Communications-Computations Overlap

\[ T = \text{shift}(A, \mu) \quad \text{sending face in } \mu \text{ & receive in } -\mu \]
\[ C = T \times B \quad \text{calls wait on send/receive} \]

\textbf{Type} *QDPF\_shift\_T(Type *a, int mu)
{
    for(I=0; I < size\_of\_face; ++I)
        QDPF\_copy\_T1T2\_repl(msgmem\_f[mu][I], a[offset[I]])
    QMP\_start(msgh\_a[mu]) \quad \text{// start receive}
    QMP\_start(msgh\_f[mu]) \quad \text{// start send}
}

void QCDF\_mult\_T3T1T2\_op3(Type3 *c, const Type1 *t,
                                      const Type2 *b)
{
    thread\_dispatch(&interior, interior\_volume, c, t, b)
    QMP\_wait(msgh\_t) \quad \text{// Wait on receive}
    thread\_dispatch(&face, face\_size, c, t, b)
    QMP\_wait(msgh\_f) \quad \text{// Wait on send}
}

void interior(int start, int end, va\_list \ args) \ // similar for face
{
    for(I=start; I < end; ++I)
        QDPF\_mult\_T3T1T2\_op3\_S(c[off3[I]], t[off1[I]], b[off2[I]])
}
Data Types

• Fields have various types (indices):

  | Color: $U^{ij}(x)$, Spin: $\Gamma_{\alpha\beta}$, $\psi^{i}_{\alpha}(x)$, $Q^{ij}_{\alpha\beta}(x)$ |

• Index type (i.e. the "fiber" over "base" lattice site)
  - Gauge: Product(Matrix(Nc),Scalar)
  - Dirac: Product(Vector(Nc),Vector(Ns))
  - Scalars: Scalar
  - Propagators: Product(Matrix(Nc),Matrix(Ns))?

• Support Red/Black sublattices & other subsets (Mask ?)

• Support compatible operations on types:

  $U^{ij}(x) * \Gamma_{\alpha\beta} * \psi^{i}_{\alpha}(x) \rightarrow \text{Matrix(color)}*\text{Matrix(\text{spin})}*\text{Vector(color,spin)}$
C Interface for Level 2

• void QCDF_mult_\_T3T1T2\_op3(Type3 *r, const Type1 *a, const Type2 *b)

• T3, T1, T2 are short for the type Type1, Type2 and Type3:
  - LatticeGaugeF, LatticeDiracFermionF, LatticeHalfFermionF, LatticePropagatorF

• op3 are options like:

<table>
<thead>
<tr>
<th>op</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>nnr</td>
<td>( r = a \ast b )</td>
</tr>
<tr>
<td>ncr</td>
<td>( r = a \ast \text{conj}(b) )</td>
</tr>
<tr>
<td>cnr</td>
<td>( r = \text{conj}(a) \ast b )</td>
</tr>
<tr>
<td>ccr</td>
<td>( r = \text{conj}(a) \ast \text{conj}(b) )</td>
</tr>
<tr>
<td>nna</td>
<td>( r = r + a \ast b )</td>
</tr>
<tr>
<td>nca</td>
<td>( r = r + a \ast \text{conj}(b) )</td>
</tr>
<tr>
<td>cna</td>
<td>( r = r + \text{conj}(a) \ast b )</td>
</tr>
<tr>
<td>cca</td>
<td>( r = r + \text{conj}(a) \ast \text{conj}(b) )</td>
</tr>
<tr>
<td>nnn</td>
<td>( r = -a \ast b )</td>
</tr>
<tr>
<td>ncn</td>
<td>( r = -a \ast \text{conj}(b) )</td>
</tr>
<tr>
<td>cnn</td>
<td>( r = -\text{conj}(a) \ast b )</td>
</tr>
<tr>
<td>ccn</td>
<td>( r = -\text{conj}(a) \ast \text{conj}(b) )</td>
</tr>
<tr>
<td>nns</td>
<td>( r = r - a \ast b )</td>
</tr>
<tr>
<td>ncs</td>
<td>( r = r - a \ast \text{conj}(b) )</td>
</tr>
<tr>
<td>cns</td>
<td>( r = r - \text{conj}(a) \ast b )</td>
</tr>
<tr>
<td>ccs</td>
<td>( r = r - \text{conj}(a) \ast \text{conj}(b) )</td>
</tr>
</tbody>
</table>
C++ Interface for Level 2

**Unary operations:** operate on one source into a target

Lattice_Field Shift(Lattice_field source, enum sign, int direction);
void Copy(Lattice_Field dest, Lattice_Field source, enum option);
void Trace(double dest, Lattice_Field source, enum option);

**Binary operations:** operate on two sources into a target

void Multiply(Lattice_Field dest, Lattice_Field src1, Lattice_Field src2, enum option);
void Compare(Lattice_Bool dest, Lattice_Field src1, Lattice_Field src2, enum compare_func);

**Broadcasts:** broadcast throughout lattice

void Fill(Lattice_Field dest, float val);

**Reductions:** reduce through the lattice

void Sum(double dest, Lattice_Field source);
III. Linear Algebra: QCD Kernels

• First draft of Level 1 Linear Algebra API (DeTar, Edwards, McClendon)
  – [http://www.jlab.org/~edwards/qcdapi/LinAlg1API_0_1.htm](http://www.jlab.org/~edwards/qcdapi/LinAlg1API_0_1.htm)

• Optimize on Pentium 4 SSE & SSE2 code:
  – for MILC (Holgren, Simone, Gottlieb)
  – for SZIN l (Edwards, McClendon)

• Vertical slice for QCD API (Pochinsky)
  – API conformant example of Dirac CG
  – MILC implementation (Osborn)
Pentium I V with Streaming SIMD Extensions (SSE)

• **Streaming SIMD** (Single Instruction Multiple Data) Extensions which are length 4 vector instructions.

• **In Pentium III**, SSE has eight 128 bit registers used in single prec., so each vector register holds and operates on four 32 bit registers.

• **In Pentium IV**, SSE2 introduced which allows use of same registers in 64 bit (double prec.) mode.

• **Pentium IV** capable of four 32 bit operations per clock (e.g., two mult-adds) or two 64 bit operations (e.g., one mult-add) per clock.
The Performance in Megaflops of 32-bit SU(3) Linear Algebra routines as a function of lattice size and operand size.

- Staggered Spinor
- Halfspinor
- SU(3) Matrix
- Spinor
The Performance in Megaflops of 64-bit SU(3) Linear Algebra routines as a function of lattice size and operand size.
Performance in Megaflops for single-node Dirac operator, 1.7 GHz Pentium 4 Xeon

![Graph showing performance in Megaflops for single-node Dirac operator, 1.7 GHz Pentium 4 Xeon. The graph compares performance for different lattice sizes and different bit sizes (32bit C, 32bit SSE, 64bit C, 64bit SSE).]
Parallel Version on Single Node

Performance in Megaflops for Parallel Dirac Operator with no communications (upper bound performance per processor for cluster), 1.7 GHz Pentium 4 Xeon

![Graph showing performance in Megaflops for different lattice sizes and processor types.](image-url)
IV. Application Porting & Opt.

• **MILC**: (revision version 6_15oct01)
  – QCDOC ASIC simulation of MILC (Calin, Christan, Toussaint, Gregory)
  – Prefetching Strategies (Holgren, Simone, Gottlieb)

• **SZIN**: (new documentation and revision) (Edwards)
  – Implementation on top of level 1 API (Edwards, Pochinsky)
  – Status: efficient code for P4 by Spring 2002

• **CPS (Columbia Physics System)**
  – Software Testing environment running on QCDSP (Miller)
  – Native OS & fabric for MP-API (Jung)
V. Data Archives and Data Grid

- **File formats and header**
  - Build on successful example of NERSC QCD archive
  - Extend to include lattice sets, propagators, etc.

- **Consider XML for ascii headers**
  - Control input and output data files
  - Search user data using SQL to find locations.

- **Lattice Portal**
  - Replicate data (multi-site), global tree structure.
  - SQL-like data base for storing data and retrieving

- **Web based computing**
  - batch system and uniform scripting tool.
VI. New Performance and Coding Tools

• Performance Analysis Tool:
  – SvPABLO instrumentation of MILC (Celso)
  – Extension through PAPI interface to P4 architecture (Dongarra)

• FNAL Tools:
  – Trace Tools extension to Pentium 4 and instrumentation of MILC (Rechenmacher, Holmgen, Matsumura)
  – FNAL “rgang” (parallel command dispatcher)
  – FermiQCD (DiPierro)

• Developing Q system (Pochinsky)
  – High level code generator of opgtimized API library
  – Current version generates ANSI C code
SvPablo Instrumentation of MILC
QCD-API Status and Schedule

- **Level 1 Message Passing** API (MP-API) design completed
  - Implementation in MPI completed (Sept 2001)
  - C++ MP-API for QCDOC (Started)
  - Application port of MILC/SZIN to MP-API (Started)
  - Myrinet optimization in GM (Feb 2002)

- **Level 1 Linear Algebra** API 1st draft completed (Feb 2002)

- **Vertical Slice** prototype of level 2 (Feb 2002)
  - Implementation for MILC (started)

- **Pentium IV** optimization of Linear Algebra (March 2002)