



# I/O and the SciDAC Software API

---

Robert Edwards

U.S. SciDAC Software Coordinating Committee

May 2, 2003



# Cast of Characters

---

**Committee:** R.Brower, C.DeTar, R.Edwards,  
D.Holmgren, R.Mawhinney, C.Mendes, C.Watson

**Additional:** J.Chen, E.Gregory, J.Hetrick, C.Jung,  
J.Osborn, K.Petrov, A.Pochinsky, J.Simone

**IO:** C.DeTar, R.Edwards, J.Osborn, J.Simone,  
B.Joó



Scientific

---

Discovery

Through

Advanced

Computing

<http://www.lqcd.org/scidac>

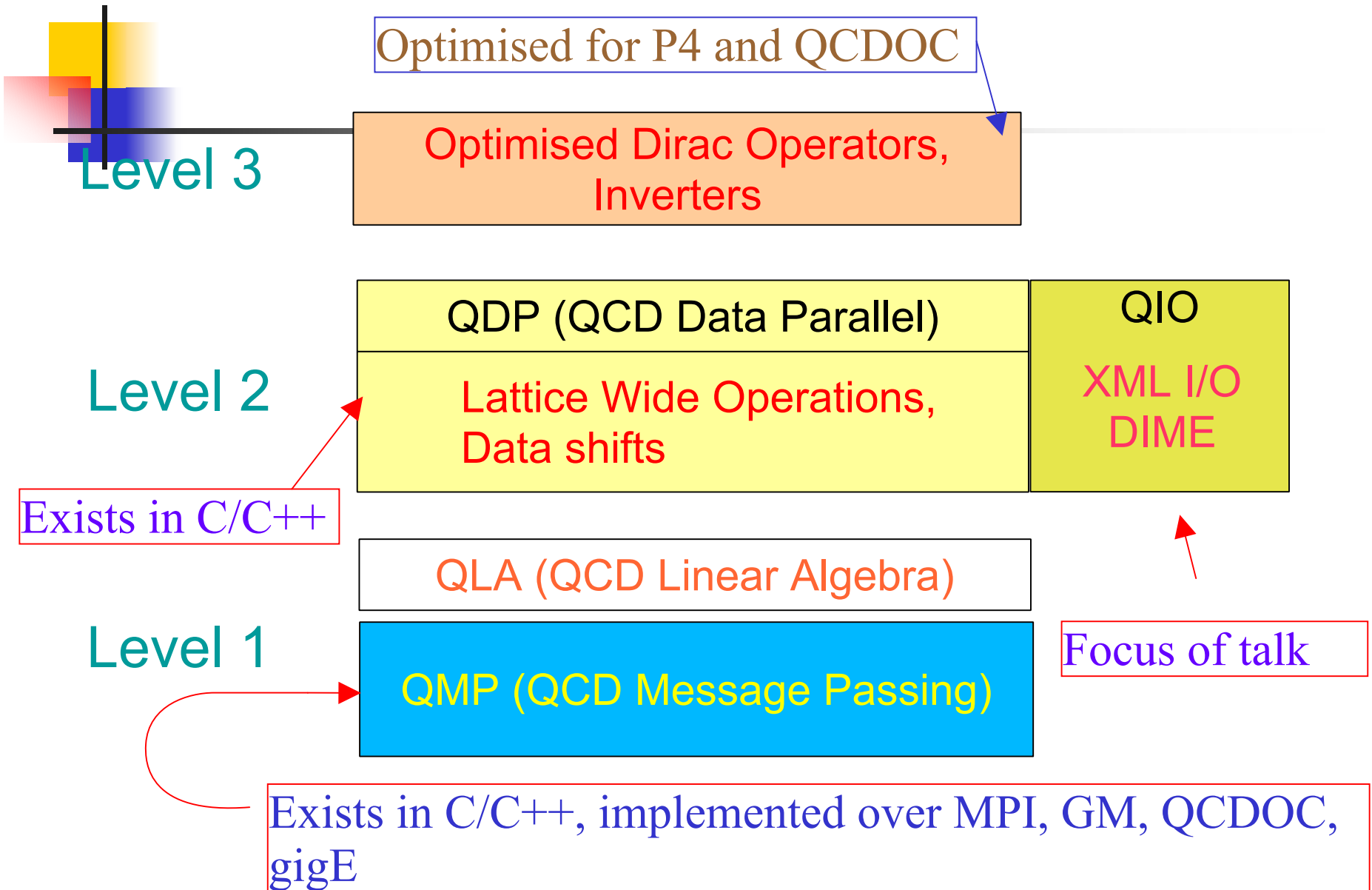


# SciDAC Project Goals

---

- Portable, scalable software
- High performance optimization on two target architectures **QCDOC** Clusters
- Exploitation and Optimization of existing application base
- Infrastructure for (US) national community
- Sharing of valuable lattice data, and data management **GRID (ILDG)**

# SciDAC Software Structure





# Data Parallel QDP/C,C++ API

---

- Hides architecture and layout
- Operates on lattice fields across sites
- Linear algebra tailored for QCD
- Shifts and permutation maps across sites
- Reductions
- Subsets



# Data-parallel Operations

---

- *Unary and binary:*

-a; a-b; ...

- *Unary functions:*

adj(a), cos(a), sin(a), ...

- *Random numbers:*

// platform independent

random(a), gaussian(a)

- *Comparisons (booleans)*

a <= b, ...

- *Broadcasts:*

a = 0, ...

- *Reductions:*

sum(a), ...



# QDP Expressions

---

- Can create expressions

$$c_{\alpha}^i(x) = U_{\mu}^{ij}(x) b_{\alpha}^i(x + \mu) + 2 d_{\alpha}^i(x) \quad \forall x$$

- QDP/C++ code

```
multild<LatticeColorMatrix>  u(Nd);  
LatticeDiracFermion  b, c, d;  
int mu;  
c = u[mu] * shift(b,mu) + 2 * d;
```

- **PETE**: Portable Expression Template Engine
  - Temporaries eliminated, expressions optimised



# Generic QDP Binary File Formats

---

- Composed of 1 or more application records
- Single application record has 1 **QDP** field or an array of fields
- Binary data in lexicographic site major order
- Physics metadata for file and for each record
- Using **DIME** to package



# Metadata

---

- Use **XML** for file and record metadata
- File and record metadata managed at user convenience
- No agreed minimum standard
- Use **binX** to describe binary
- **binX not** in record metadata – provides serialization info



# Gauge Fields

---

- For **published** data – still converging on metadata
  - Considering adopting ILDG-like schema
  - Need more cases, like Asqtad, domain wall
- Likely that **non-published/private** configs will use reduced set of metadata
- Write arrays of fields as one record – all 3 rows
- Site major order – slowest varying
- Will adopt single format and byte ordering



# File Format

---

- File physics metadata
- Application record 1
  - Physics metadata
  - binX description
  - Binary data [may have array indices within sites]
  - Checksum
- Record 2 - possible additional records
  - Physics metadata
  - binX description
  - Binary data
  - Checksum
- Record 3....



# Data Hierarchy

---

- **Project** built from datasets (e.g. gauge fields and propagators)
- **Dataset** built from files (e.g. gauge fields)
- **File** built from records (e.g. eigenvectors)
- **Record** = **QDP** field and **metadata**



# Direct Internet Message Encapsulation (DIME)

- Data written to (read from) a list of records
- Each record has
  - DIME Type (required)
    - URL or like MIME type
  - DIME Id (optional URL)
- Maximum record size is 2Gb
- Data larger than 2Gb can be split into successive record “chunks”
- Chunking easy, file size > 2Gb a problem



# QIO: Grid Friendly I/O

---

- Metadata & Physics data Reader / Writer API
  - Read/Write simple XML documents
  - Not using data binding
  - Metadata used like a buffer, physics data like a stream
- QDP IO (QIO)
  - **Serial** – all nodes stream through one node
  - **Parallel** – if available, many nodes to parallel filesystem

```
MetaWriter file_xml, record_xml;  
SerialFileWriter out(file_xml, "foo.dat");  
LatticeDiracFermion psi;  
out.write(record_xml, psi);
```

# MetaReader

*File xml:*

```
struct foo_t foo;
struct bar_t bar;
double kappa;
MetaReader in;
char *key="/foo/bar/kappa";

in.get<foo_t>(foo, "/foo")
in.get<double>(kappa, key);
```

```
<foo>
  <bar>
    <kappa>
      0.120
    </kappa>
  </bar>
</foo>
```

- XML Reader/Writer supports recursive serialization
  - To/From buffers (strings - **MetaData**)
  - To/From files (**PhysicsData**)
- Intended to drive codes rather than **DataGrid**
- C,C++ versions



# Current Status

---

- Releases and documentation

<http://www.lqcd.org/scidac>

- QMP, QDP/C,C++ in first release
- Performance improvements/testing underway
- Porting & development efforts of physics codes over QDP on-going
- QIO near completion
- DIME completed (Bálint Joó)
- XML Reader/Writer in development