Code development for lattice QCD and beyond

James C. Osborn

Argonne National Laboratory

Lattice Meets Experiment 2011:
Beyond the Standard Model

October 14-15, 2011, FNAL

US Lattice QCD software development effort

- Formal effort organized about 10 years ago
- Funded by US DOE SciDAC program, national labs and other DOE/NSF grants
- Over 30 people at 16 institutions (on weekly calls)
  - ANL, Boston U., BNL, Columbia U., DePaul U., FNAL, Harvard, Jlab, LLNL, MIT, RENCI, U. Arizona, UC Santa Barbara, U. Indiana, U. Pacific, U. Utah
- Plus many other researchers who also also contribute code and algorithms

US Lattice QCD SciDAC libraries

layered approach to software

level 1:

- QMP message passing
- QLA linear algebra

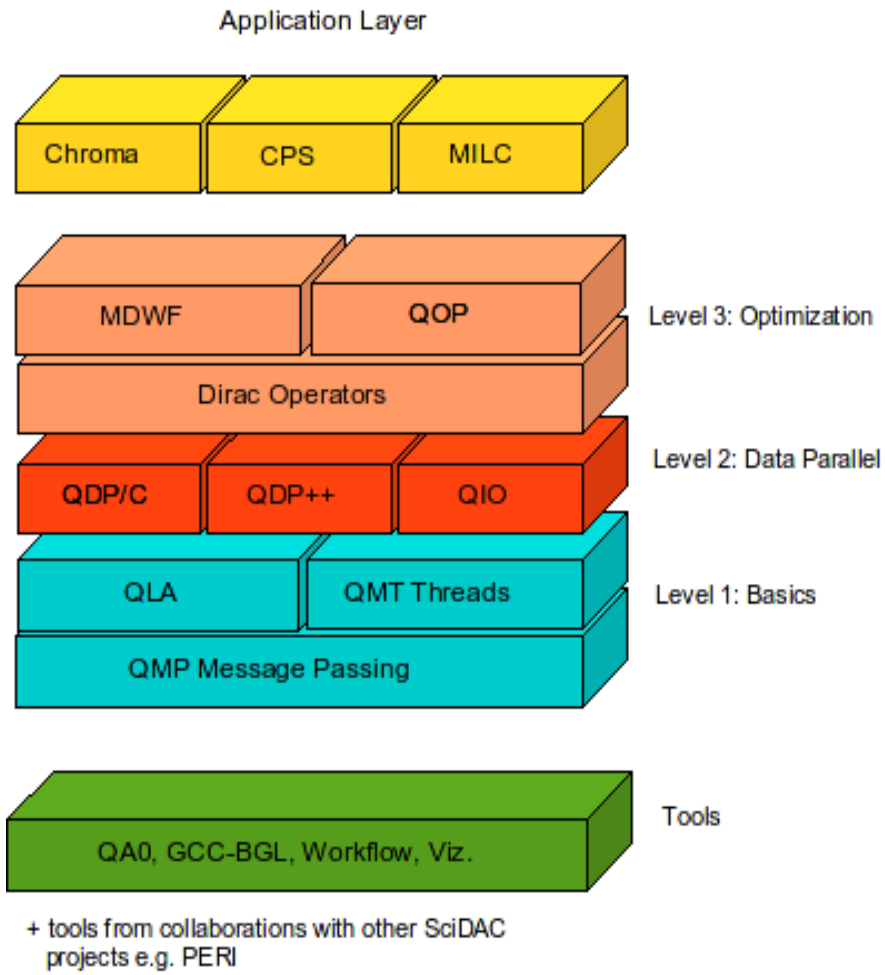
level 2:

- QIO I/O
- QDP data parallel

level 3:

- Optimized versions of common routines

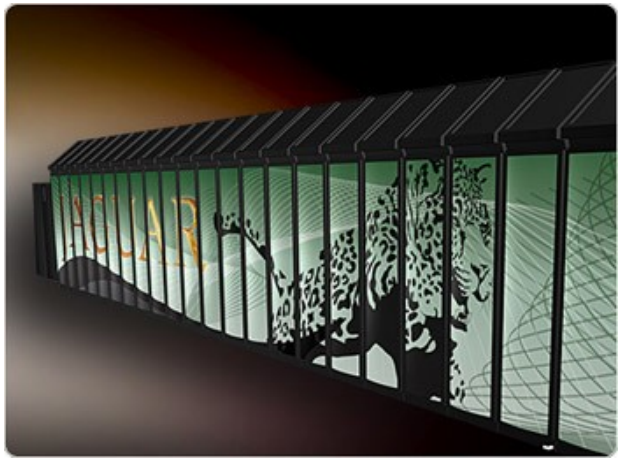
Application suites can use components at any level



Targeted machines

IBM Blue Gene

Cray XT



x86 clusters

GPU clusters

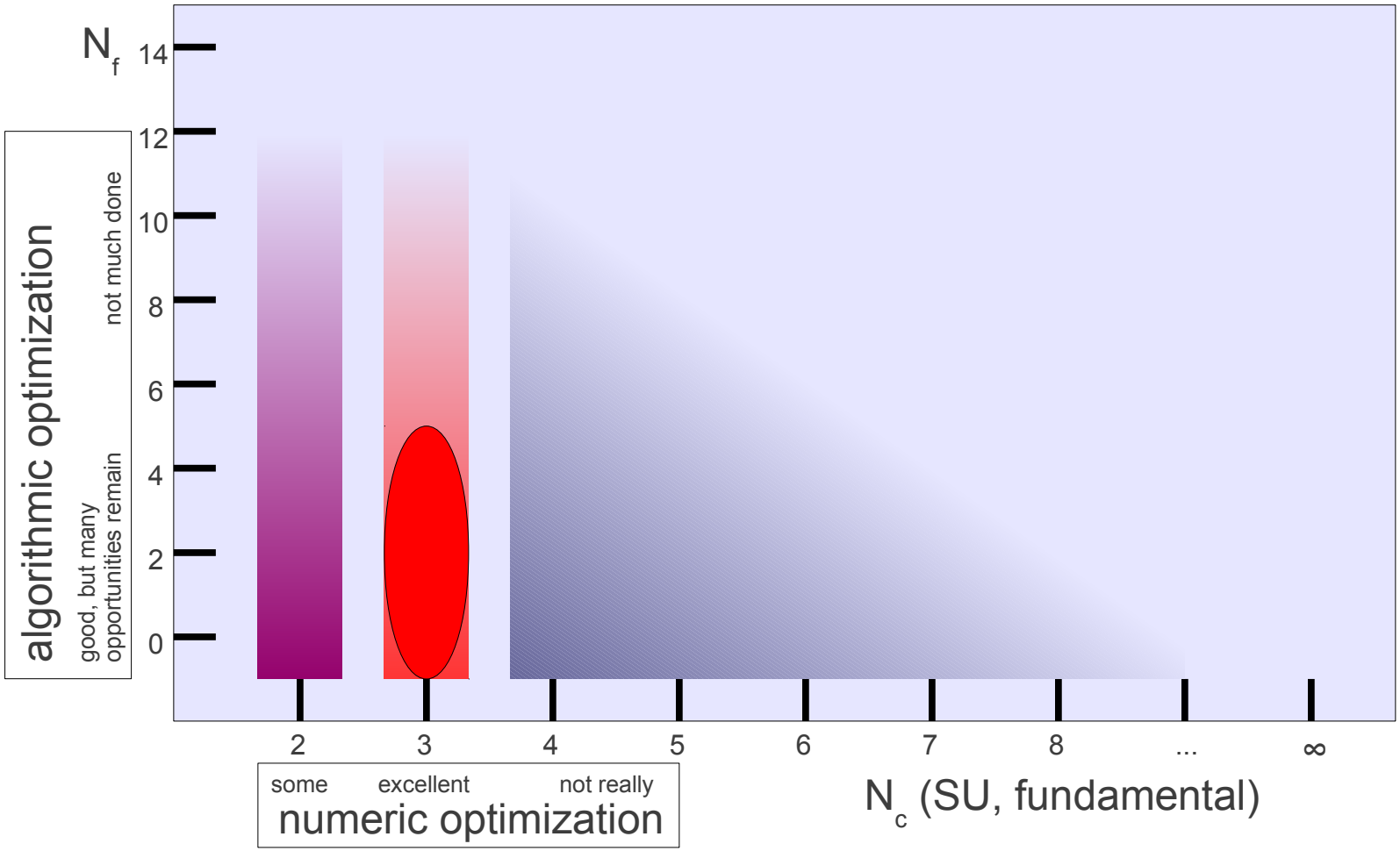
Total US community software effort

- SciDAC program for about 10 years, some application codes are much older
- Libraries (not complete list):
  - QMP: 8k lines in MPI version + optimized versions for other architectures
  - QLA: 20k lines (generates 180k lines)
  - QIO: 21k lines
  - QDP: 13k lines (generates 52k lines)
  - QDP++: 187k lines (some of it generated)
  - QOPQDP: 23k lines
  - QUDA: 45k lines

  Total: 317k lines (549k including generated code)
- Application suites:
  - Chroma: 461k lines
  - CPS: 220k lines
  - MILC: 201k lines

  Total: 882k lines

Current community lattice software



What's next?

- Wish list:
  - improved large N_f support
  - improved $N_c \neq 3$ support
  - other fermion representations (adjoint, 2-index symmetric, …)
  - multiple gauge and fermion actions
  - other gauge groups?
  - optimized for all major computer architectures (current and near future)
- Total effort required: >20x total effort for QCD
  - finish by 2100 (SciDAC-20)?
- Can't do everything at once
  - start simple
  - reuse as much as possible

Community software for Beyond Standard Model lattice simulations

- Now working to come up with plan for US community Beyond Standard Model lattice software
- Have an initial working prototype of one approach
- Not necessarily only or final community plan

Divide and conquer

- 2 main steps to lattice field theory calculations
  - gauge configuration generation
  - analysis

- Focus on gauge configuration generation
  - logical starting point
  - relatively well defined problem
  - needs only a few basic pieces (action, Dirac solver, force terms)
  - can add simple analysis later as needed

Gauge configuration generation framework requirements

- High level layer focused on gauge configuration generation
  - algorithmic abstraction: generation algorithm independent of gauge group, action, etc.
  - easy to write new high-level algorithms, tune parameters
  - easy to plug in new routines
  - new routines can be written in any other language/framework
  - perfect for scripting language

- Scripting language requirements
  - Small
  - Easy to use
  - Easy to port
  - Easy to embed and interface with existing or new libraries

Lua

- Small, simple, fast and powerful scripting language
- Developed at Computer Graphics Technology Group (Tecgraf) at the Pontifical Catholic University of Rio de Janeiro (PUC-Rio), in Brazil
- Name means “moon” in Portuguese
- About 17k lines of ANSI C (easily ported)
- Designed to be embedded and easily interface with C libraries
- Liberal MIT license

Initial prototype

- Initially using SciDAC QDP/C and QOPQDP libraries to provide needed routines
- Supports SU(3) lattice generation with HISQ (highly improved staggered quarks)
  - using RHMC (Rational Hybrid Monte Carlo) algorithm
  - also supports HMC with mass precondition (suitable for $N_f = 4,8,12,...$)
- Matches conventions of current MILC code (and can parse same input files)
- Size:
  - QOPQDP/Lua interface: 3k lines
  - (R)HMC/bookkeeping in Lua: 1k lines

Hybrid Monte Carlo example

```
function hmcstep(fields, params)
  fields:save()
  local Sold = fields:action()

  local intparams =
        setupint(fields, params)
  integrate(fields, intparams)

  local Snew = fields:action()
```

```
  local ds = Snew - Sold
  local p = math.exp(-ds)
  local r = globalRand()
  if( r > p ) then -- reject
    fields:reject()
  else
    fields:accept()
  end
end
```

Integration example

- Integration also abstracted, e.g. leapfrog (1 field and 1 force term):

  fields:updateField(1, eps/2)

  fields:updateMomentum(1, 1, eps)

  fields:updateField(1, eps/2)

- Actual code completely general
  - arbitrary integration patterns (leapfrog; Omelyan, et al.; custom)
  - any number of fields and force terms per field
  - different number of steps for each force term

Current plans

- Add Domain Wall quark support
- Add QUDA (GPU) routines as alternatives
- Add SU(2) support

- Use as testbed for algorithmic research
  - improve HMC for large N_f (e.g. mass preconditioning)
  - improved integrators (e.g. force gradient)
  - improved actions
  - multigrid (solver, DD-HMC)

Summary

- Large effort has gone into lattice QCD software
- Not practical to repeat for every group/representation/action one might want to study
- Start with simple framework with minimal assumptions
- Provide easy way to plug in new routines as needed
- Most of the work is still in writing new routines, but can be done independently without constraints that a larger application suite might impose