Chroma I: A High Level View

Bálint Joó (bjoo@jlab.org)
Jefferson Lab, Newport News, VA

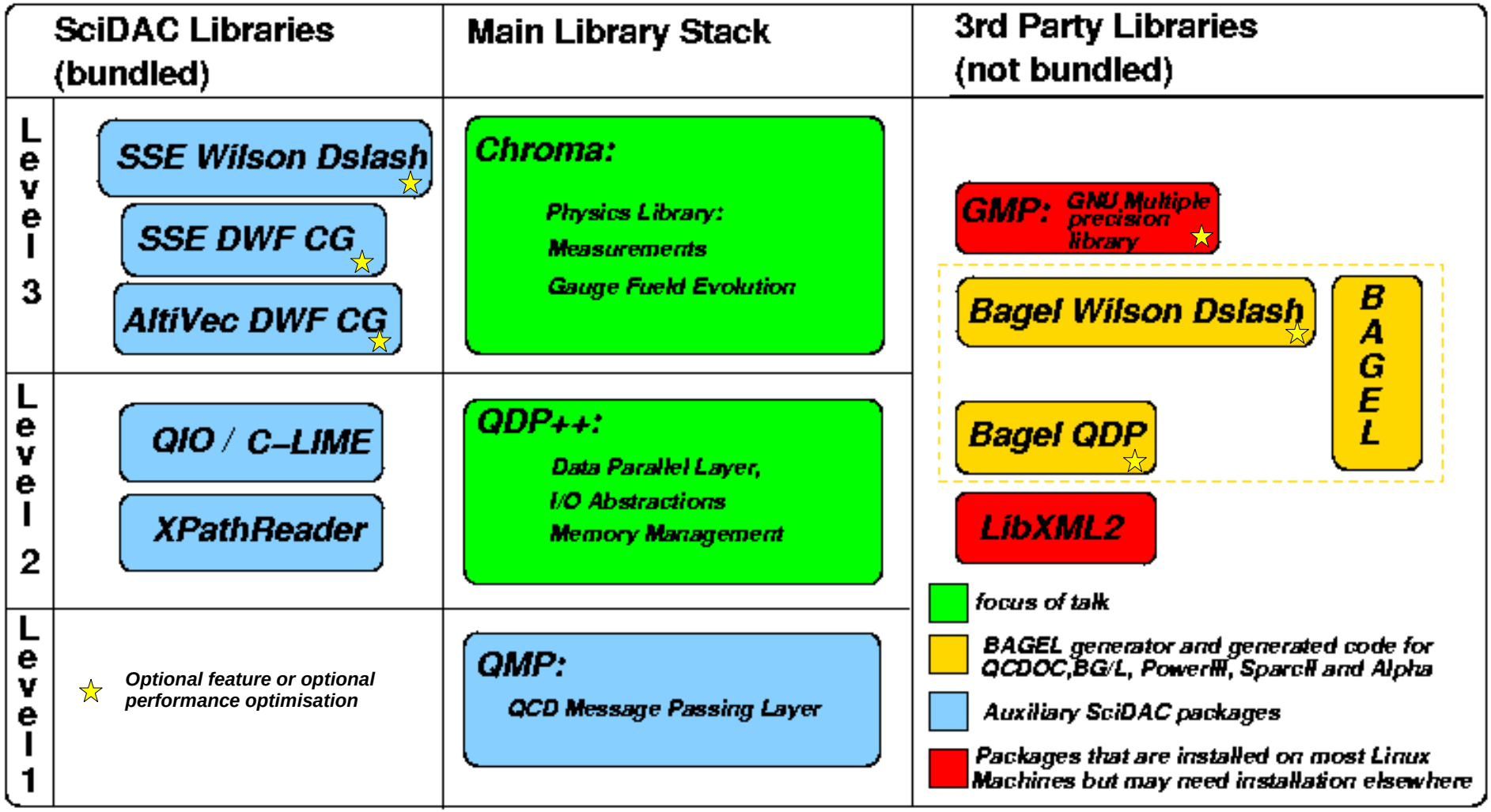
given at

HackLatt'06
NeSC, Edinburgh
March 29, 2006

Introduction

- Chroma and its dependent software components
- Getting it
- Building it
- Linking against it
- What's in it?
- Running it
- Believing it (?)

Chroma and Other Software Components



Getting the bits and pieces

- QMP, QDP++ and Chroma (and bundled packages) and BAGEL QDP from anonymous CVS:
  - Root :pserver:anonymous@cvs.jlab.org:/group/lattice/cvsroot
  - Modules: bagel_qdp, qmp, qdp++, chroma
  - USQCD Web page:
    - http://www.usqcd.org/usqcd/software
- BAGEL and Wilson Dslash from Peter's web page
  - http://www.ph.ed.ac.uk/~paboyle/bagel/Bagel.html
- GMP from: http://www.swox.com/gmp
- LibXML2 from: http://www.xmlsoft.org

Building

- Easiest Target: Anyone Can Build
  - Scalar Workstation Build
  - Assumption: libXML and GMP already installed with OS distribution
  - Modules to build: qdp++, chroma
- Most Difficult Targets:
  - QCDOC, BG/L and IBM Pseries or new Machines
  - Modules to build: gmp, bagel modules, libxml, qdp++, chroma. Possibly qmp too (except where it is native)
  - Best done by developing a script

General Build Information

- Uses GNU autoconf 2.59 and GNU automake 1.9.3
- Typical configure; make ; make install type build
- Atypical aspects:
  - Work in "cross compilation" mode most of the time
    - Autoconf cannot autodetect many features of target system (Custom OS, Queues etc etc)
    - pervasive use of --enable-feature and --with-package switches
  - Want most efficient compiler flags
    - CXXFLAGS and CFLAGS on configure command line get complicated

Some Typical Flags: QDP++

- QDP++
  - --prefix=<install location>
  - --enable-parallel-arch=(scalar|parscalar)
  - --enable-precision=(single|double)
  - --enable-sse, --enable-sse2 (SSE compatible only)
  - --with-libxml2=<location of libxml2 installation>
  - --with-qmp=<location of QMP installation>
  - --with-bagel-qdp=<location of Bagel QDP installation>
  - --enable-qcdoc (QCDOC Specific memory allocator)
  - CXXFLAGS="-O2 -finline-limit=50000" CFLAGS="-O2"

Some Typical Flags: Chroma

- Chroma
  - --prefix=<install location>
  - --with-qdp=<location of QDP++ installation>
  - --enable-sse-wilson-dslash (SSE Only)
  - --enable-sse-dwf-cg (SSE single prec only)
  - --enable-altivec-dwf-cg (AltiVec Single prec only)
  - --with-bagel-wilson-dslash=<location of BAGEL dslash>
  - --with-gmp=<location of GMP library>
  - CXXFLAGS="" CFLAGS="" (disables default -g flag)
    - Other flags picked up from lower layers eg QDP++

Hints to help you get building

- There is a work in progress installation guide
  - http://www.ph.ed.ac.uk/~bj/HackLatt05/Installation/html/index.html
- There are possibly out of date example configurations in
  - qdp++/install_scripts/
  - chroma/install_scripts/
- There is a standardized build setup for the JLAB based on the nightly builds (useful for clusters)
  - CVS Module: jlab-standard-chroma-build

Linking against already installed chroma

- Suppose chroma is installed in /foo/chroma
- Use script chroma-config in /foo/chroma/bin
  - CXX=`chroma-config --cxx`
  - CXXFLAGS=`chroma-config --cxxflags`
  - LDFLAGS=`chroma-config --ldflags`
  - LIBS=`chroma-config --libs`
- Compile your program (prog.cc) with:
  - $(CXX) $(CXXFLAGS) prog.cc $(LDFLAGS) $(LIBS)
  - NB: Ordering of flags may be important.

What is in Chroma?

Measurements:
(sequential) sources, smearings propagators spectroscopy, 3pt functions, hadron structure, wilson loops, eigenvalues

Fermion Actions:
wilson, tm, clover, 4D and 5D overlap, variety of coeffs, DWF, AsqTAD

Monomials:
two flavor 4D&5D, one flavor rational 4D&5D, Hasenbusch Term (4D), LogDetEvenEven

GaugeActions
plaquette, rectangle, tree level and 1 loop LW, RG impr. plaq+rect, DBW2

I/O Support:
NERSC, CPPACS, UKY, SciDAC and ILDG Configurations

Inverters:
CG, CGNE, BiCGStab, Multi Shift CG, SUMR, GMRESR, MINRES

Chronological Predictors:
Zero Guess, Last Solution, Linear Extrapolation, Minimum Residual

Eigensystems:
Kalkreuter-Simma Ritz

MD Integrators:
Leapfrog, Omelyan (SW?) and Multi Time Scale versions of same

Boundaries:
(anti)periodic,Dirichlet, twisted, Schroedinger Functional

Measurement (chroma) | HMC (hmc) | Pure Gauge Heatbath (purgaug)

Jefferson Lab
Exploring the Nature of Matter

Chroma Applications

- Measurement Application: chroma
- Gauge Generation Applications: hmc and purgaug
- Installed in same place as chroma-config
  - eg: /foo/chroma/bin
- Typical usage flags (-i, -o, -geom):
  - ./chroma -i in.xml -o out.xml -geom "Px Py Pz Pt"
  - in.xml – Input Parameter XML File
  - out.xml – Output XML Log File
  - "Px Py Pz Pt" the (possibly virtual) Processor Geometry (eg -geom "4 4 8 8" for QCDOC Rack)

XML Driven Programs – Chroma Input File

```
<?xml version="1.0" encoding="UTF-8"?>
<chroma>
<annotation>Your annotation here</annotation>
<Param>
  <InlineMeasurements>
    <elem>
      <Name>MAKE_SOURCE</Name>
      <Frequency>1</Frequency>
      <Param/>
      <NamedObject>
        <source_id>sh_source_0</source_id>
      </NamedObject>
    </elem>

    <elem>
      <Name>PROPAGATOR</Name>
      <Frequency>1</Frequency>
      <Param/>
      <NamedObject>
        <source_id>sh_source_0</source_id>
        <prop_id>sh_prop_0</prop_id>
      </NamedObject>
    </elem>
  </InlineMeasurements>
  <nrow>4 4 4 8</nrow>
</Param>
<RNG/>
<Cfg>
 <cfg_type>SCIDAC</cfg_type>
 <cfg_file>foo.lime</cfg_file>
</Cfg>
</chroma>
```

Array of Measurements (Tasks)

Task (array element)

Task Name

Task specific parameters

Named Objects (communicate between tasks -- like "in memory" files)

Global Lattice Size

Input Configuration Details

Jefferson Lab
Exploring the Nature of Matter

XML Input File Examples

- Numerous Measurement Task Examples in
  - chroma/tests/chroma/hadron/
- Measurement Tasks for:
  - sources, smearings, propagators, spectroscopy, 3pt functions, eigenvalues
  - Reading and Writing Named Objects
- Also MD and HMC input files in
  - chroma/tests/t_leapfrog
  - chroma/tests/hmc
- Input file names usually contain the string "ini"

Software Quality Assurance (Testing)

- Nightly builds at the JLab (and elsewhere)
  - framework developed by Zbyszek and Craig in Liverpool and Robert Edwards at the JLAB
  - Tests compatibility with compilers, configuration, building, linking and linking to installed libraries.
  - Runs Regression Tests on single node targets
  - Current nightly builds:
    - parscalar build with SSE, QMP-single comms, g++ v4
    - parscalar build with SSE, QMP-MPICH over Infiniband with g++ v3
    - parscalar build with QMP-single comms and BAGEL noarch targets
    - scalar build with SSE

Chroma Regression Tests

- Framework from Craig and Zbyszek and Robert.
- Verifies that new code does not break old behaviour (not that it is necessarily correct)
- Uses xmldiff utility from EPCC to compare XML files
- Test coverage constantly growing (never enough)
- Single node only for now: make xcheck

```
Source /home/bj/Devel/QCD/chroma/tests/chroma/hadron/propagator/regres.pl

Program         Candidate                                   Conclusion
chroma          unprec-zolo-ev-multi-v8.candidate.xml         PASS
chroma          prec_wilson-v9.candidate.xml                  PASS
chroma          prec_clover-v9.candidate.xml                  PASS
chroma          unprec_clover-v9.candidate.xml                PASS
chroma          prec_wilson-twisted-v9.candidate.xml          PASS
```

Future improvements (SciDAC2)

- Infrastructure improvements
  - Improve native speed of QDP++ (more PETE)
  - More Regression tests (more more more!!!)
  - Automated regression and unit tests for QMP, QDP++, QIO etc
  - Documentation (Yes! Really!)
  - Interface with SciDAC level I (QLA)
  - MultiCore/Threaded optimised code (QMC)
  - SSE3 Code
  - BlueGene Code?

Potential Chroma Improvements

- Better Eigensolvers
- Dynamical Fermion Algorithms
- Stout links in MD evolution
- Multiple timescales for RHMC pieces
  - already have them for 2 flavour code and Hasenbusch thanks to Carsten Urbach
- Currently input configuration is special – change this
  - will be able to smear a config inline without having to start a second chroma run
- Priority dictated by project need as always

Summary (Weighing it all up)

Good Side:

- Layered – Extensive use of SciDAC and 3rd party libraries
- Very portable
- Speed through cliche-d operations and assembly
- Quality assurance

Bad Side:

- High complexity. Can be difficult to build on some systems
- Compiler constraints
- Slow without assembly code
- Needs documentation

Capable of delivering a Wide Variety of Physics