Argonne NATIONAL LABORATORY

DOE Leadership Computing and Lattice QCD

James C. Osborn

Argonne Leadership Computing Facility

INFIERI workshop
October 21, 2016
FNAL

THE LEADERSHIP COMPUTING FACILITY

- Funded by DOE ***Advanced Scientific Computing Research*** *(ASCR)*
- Operates as two centers, at Argonne and at Oak Ridge National Laboratory, and are **fully dedicated** to open science.
- Operates two petascale architectures that are **many times more powerful** than systems typically available for open scientific research.
- Available on competitive basis to researchers around the world

OAK RIDGE National Laboratory

Argonne NATIONAL LABORATORY

Argonne Leadership Computing Facility

Argonne NATIONAL LABORATORY

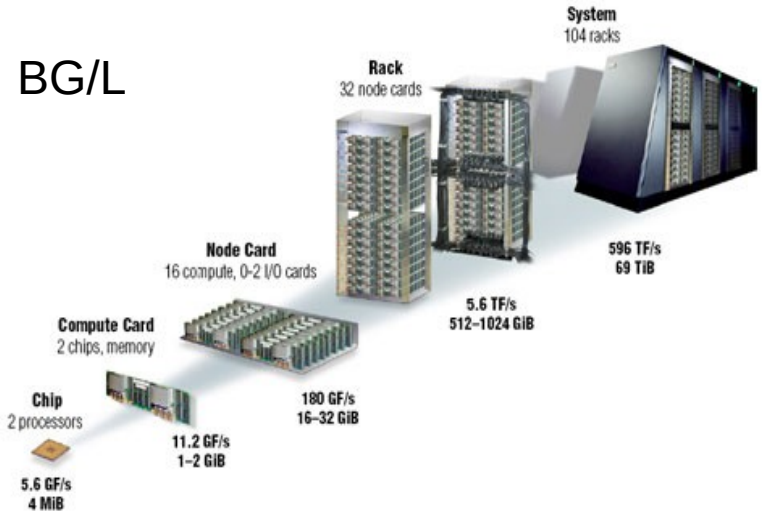
World's top supercomputers (www.top500.org)

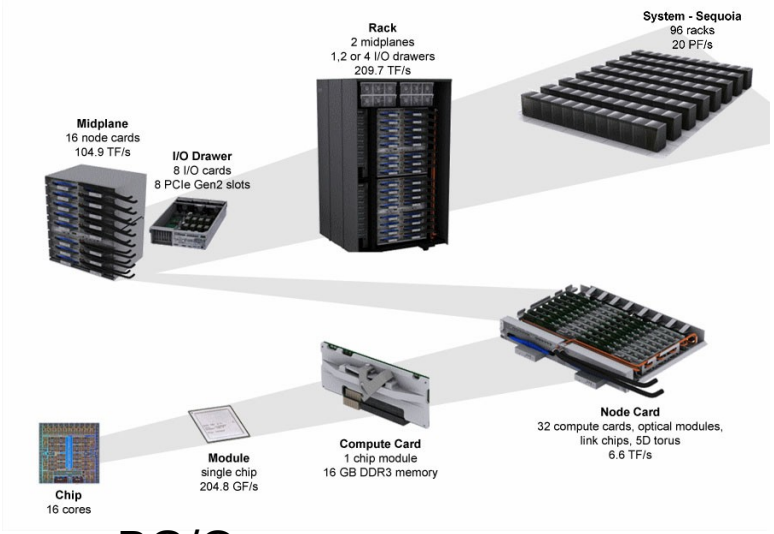
Rank	Site	System	Cores	Linpack (Tflop/s)	Peak (Tflop/s)
1	National Super-computing Center in Wuxi, China	TaihuLight Sunway MPP NRCPC	10,649,600	93,015	125,436
2	National Super Computer Center Guangzhou, China	Tianhe-2 - Intel Xeon & Xeon Phi NUDT	3,120,000	33,863	54,902
3	DOE/SC/Oak Ridge National Laboratory, USA	Titan - Opteron & NVIDIA K20x Cray Inc.	560,640	17,590	27,113
4	DOE/NNSA/LLNL United States	Sequoia Blue Gene/Q IBM	1,572,864	17,173	20,132
5	RIKEN AICS Japan	K computer SPARC64 Fujitsu	705,024	10,510	11,280
6	DOE/SC/Argonne National Laboratory, USA	Mira Blue Gene/Q IBM	786,432	8,587	10,066

IBM Blue Gene line

- Blue Gene/L
  - Grew out of QCDOC
  - Lots of low power CPUs (PowerPC 440)
  - Balanced memory and network bandwidth
- Blue Gene/P
- Blue Gene/Q
  - LQCD co-designed prefetcher
- End of Blue Gene line
  - Hard to compete with commodity processors and networks
  - Move toward high flop, low bandwidth architectures

BG/L





BG/Q

Upcoming DOE Machines

ASCR Computing Upgrades At a Glance

System attributes	NERSC Now	OLCF Now	ALCF Now	NERSC Upgrade	OLCF Upgrade	ALCF Upgrades	
Name Planned Installation	**Edison**	**TITAN**	**MIRA**	**Cori 2016**	**Summit 2017-2018**	**Theta 2016**	**Aurora 2018-2019**
System peak (PF)	2.6	27	10	> 30	150	>8.5	180
Peak Power (MW)	2	9	4.8	< 3.7	10	1.7	13
Total system memory	357 TB	710TB	768TB	~1 PB DDR4 + High Bandwidth Memory (HBM)+1.5PB persistent memory	> 1.74 PB DDR4 + HBM + 2.8 PB persistent memory	>480 TB DDR4 + High Bandwidth Memory (HBM)	> 7 PB High Bandwidth On-Package Memory Local Memory and Persistent Memory
Node performance (TF)	0.460	1.452	0.204	> 3	> 40	> 3	> 17 times Mira
Node processors	Intel Ivy Bridge	AMD Opteron Nvidia Kepler	64-bit PowerPC A2	Intel Knights Landing many core CPUs Intel Haswell CPU in data partition	Multiple IBM Power9 CPUs & multiple Nvidia Voltas GPUS	Intel Knights Landing Xeon Phi many core CPUs	Knights Hill Xeon Phi many core CPUs
System size (nodes)	5,600 nodes	18,688 nodes	49,152	9,300 nodes 1,900 nodes in data partition	~3,500 nodes	>2,500 nodes	>50,000 nodes
System Interconnect	Aries	Gemini	5D Torus	Aries	Dual Rail EDR-IB	Aries	2nd Generation Intel Omni-Path Architecture
File System	7.6 PB 168 GB/s, Lustre®	32 PB 1 TB/s, Lustre®	26 PB 300 GB/s GPFS™	28 PB 744 GB/s Lustre®	120 PB 1 TB/s GPFS™	10PB, 210 GB/s Lustre initial	150 PB 1 TB/s Lustre®

U.S. DEPARTMENT OF ENERGY | Office of Science

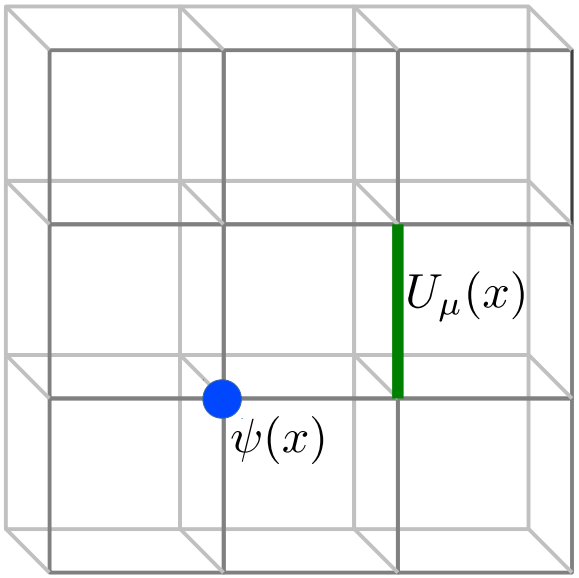
Emerging HPC hardware

- FPGAs
  - Growing interest in using as HPC accelerators
  - Work being done at ANL (LCF & MCS)
    - Prototyping OpenMP frontend
    - Investigating reducing compile times
    - Also exploring reduced precision
- Neuromorphic computing
  - Modeled on brain neurons
  - Mostly positioned for pattern matching, data processing could make way into HPC eventually
- Quantum Computing
  - Promising idea, but major technical challenges
- Above not likely major part of exascale computer, but could have some part (FPGA)
- Post-exascale era is very uncertain
  - Plan to need to rewrite some components of framework (but not all)

Lattice QCD Software Design

Lattice QCD (Quantum Chromodynamics)

- Simulations of strong nuclear force (quarks and gluons)
- Fields on 4d (space-time) lattice
- Gluons live on links
  3x3 complex matrix (unitary) $U_\mu(x)$
- Quarks on sites
  complex 3-vector (staggered) $\psi(x)$
- Logically everything represented as fields with objects on sites of lattice
- Major kernel is large sparse solver (Dirac equation)



Lattice field operations (i.e. data parallel interface)

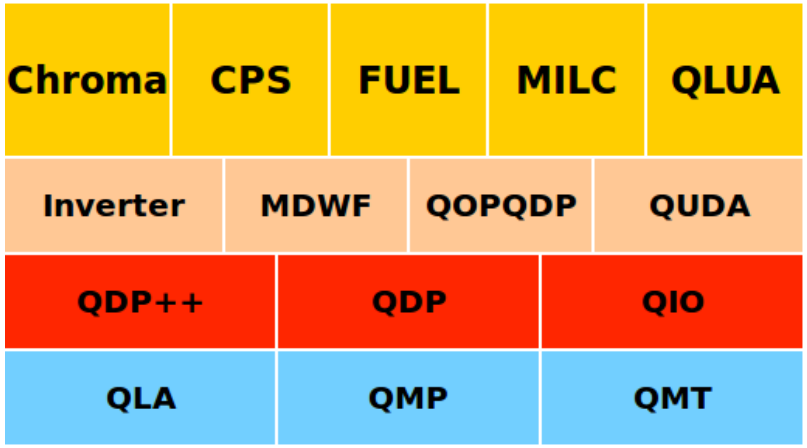
- Logically think of fields as array (lattice sites) of structures (vector, matrix, …)
- Data parallel abstraction takes care of distribution of sites to nodes
- Want to write high level expressions that generate efficient code (avoid temporary fields as much as possible)

```
v2 += m1 * v1
(v1,v2 vector field, m1 matrix field)
```

- Can be done with C++ expression templates
  - QDP++ (C++98 + PETE)
  - Grid (C++11)

USQCD software framework developed under DOE SciDAC program

- Development of software “levels” started in 2001
- Level 1 (lowest):
  - QMP – message passing
  - QLA – linear algebra
    (perl code generator for C code)
- Level 2 (middle):
  - QDP/QDP++ – data parallel
  - QIO – I/O
- Level 3 (upper):
  - Algorithms (sparse linear solvers, forces, …)
- Top is application software



- Designed with 2001+ architectures and algorithms in mind
  - Limited support for
    SIMD vectors and threading
  - Single lattice size (no multigrid)

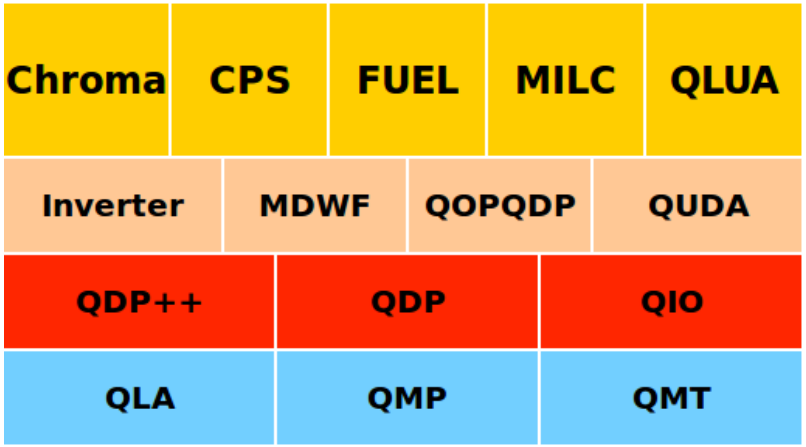
Adapting to new architectures

- GPU support
  - Originally required special-purpose languages (shaders, CUDA, …)
  - QUDA “level 3” library in CUDA: highly optimized solvers, etc. using Python code generator
  - LLVM gets JIT support for PTX
    QDP++ expression template framework ported to use it
  - Exploring GPU support from OpenMP 4.0+, OpenACC
- Xeon Phi (Knights Corner)
  - QPhiX library designed for KNC, extended to KNL and other CPUs
  - Using C++ code generator to generate C++
- Ultimate goal of portable performance still a ways off

Adapting to new algorithms

- Multigrid solvers
  - Support added later for multiple lattices
  - Not as elegant as if in original design
- Mixed (reduced) precision
  - GPU fast support for 16 bit fixed and floating point - supported in QUDA
  - Developed new algorithms to improve stability
- Reduce communications
  - Block Jacobi, Schwartz preconditioners
  - Mostly done in new code, not existing framework

Other high-level approaches



- Scripting languages
  - Ease of use, rapid prototyping & development
  - Wrap optimized libraries
- Use Lua to wrap C libraries
  - QLUA uses expressions (with lattice temporaries)
  - FUEL (functions instead of expressions)

- Exploring alternative high-level languages that also allow low level optimization

Nim (nim-lang.org)

- Modern language started in 2008
- Borrows from: Modula 3, Delphi, Ada, C++, Python, Lisp, Oberon
  - “Efficient like C, expressive like Python and flexible like Lisp”
- High-level language – feels more like Python (but is statically typed)
- Generates C/C++/JS/PHP code from Nim
- Extensive meta-programming support
  (nearly full language available at compile time)
- Still young for language
  - Current version 0.15
  - Strong desire to work towards 1.0 (backward stability)
  - Small, but growing community (users and developers)
  - Some company support (mostly web, games), many waiting for 1.0

Tensor operations

- General tensor support in development:

```
tensorOps:
  v2 = 0
  v2 += v1 + 0.1
  v3 += m1 * v2
```

(above code block transforms to the pseudocode)

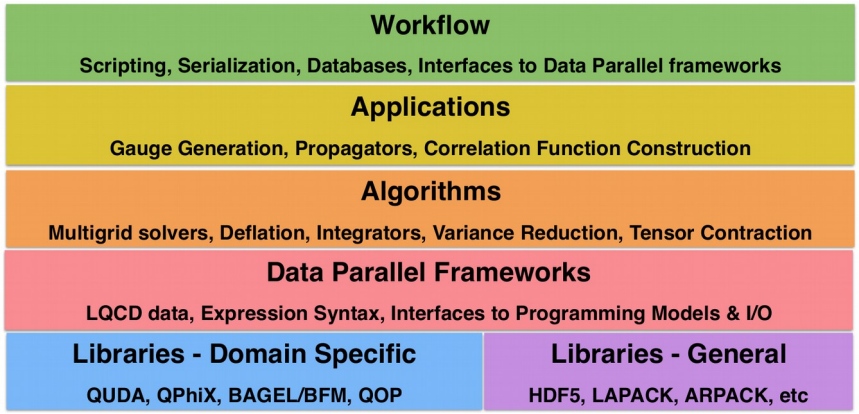
```
for j in 0..2:
  v2[j] = 0
  v2[j] += v1[j] + 0.1
  for k in 0..2:
    v3[k] += m1[k,j] * v2[j]
```

- Can also use Einstein notation (autosummation):

```
v1[a] = p[mu,mu,a,b] * v2[b]
```

USQCD Exascale Computing Project

- Goal to be ready for exascale systems in early 2020's
- Redesign software infrastructure for future machines and *algorithms*
  - Threading and vectorization key design points
- Keep flexibility in mind
  - Don't know all possible architectures and algorithms
  - Make abstractions as lightweight and flexible as possible
- Prepare for unknown future, and also changes in known plans



Updates in architecture roadmaps

- Intel Knights Mill
  - Announced in August, expected in 2017
  - Enhancements for high performance machine learning training
  - Mixed precision performance
  - High memory bandwidth
- Need to be ready to adapt to new architecture features
- Post-exascale will likely hold many new surprises

Summary

- HPC architectures had major divergence with introduction of GPUs
  - Few, if any, codes were prepared for it
  - Expression level abstraction (QDP++ & JIT) able to adapt
- Important to have flexible design, high-level abstractions
- C++ metaprogramming and code generators written in Perl/Python/C++ can provide flexibility and help with optimizations. Nim is a promising new language that offers best of both; will hopefully increase usage in HPC.
- Want ability to take advantage of new hardware features without changing algorithmic code
- New hardware may prefer (require) new algorithms
  - High-level abstraction for easy implementation of algorithms