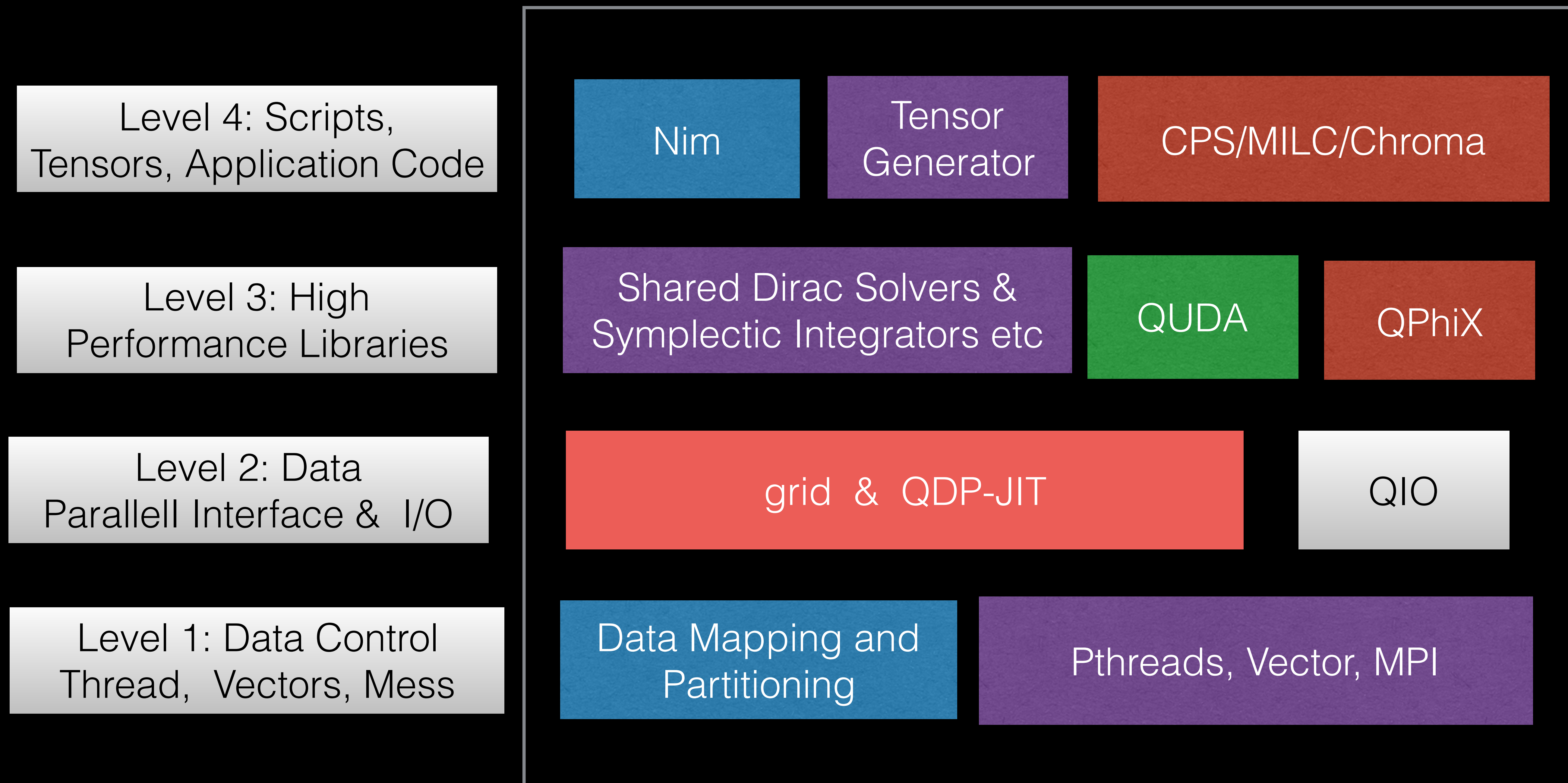
Evolving Lattice Field Theory Software Landscape



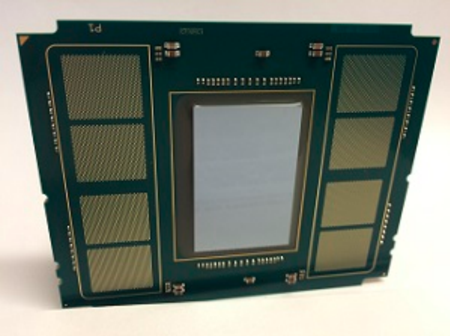
Software Development at JLab

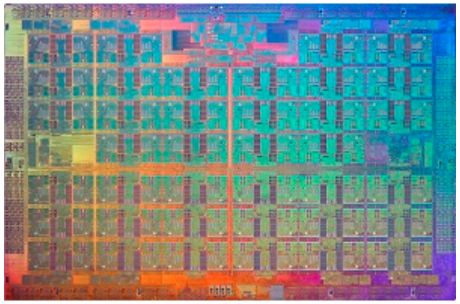
Balint Joo, Jefferson Lab
USQCD AHM, BNL, April 28/29/30

Developing For KNL

- QPhiX Optimization for AVX512
  - B. Joo (JLab), D. Kalamkar (Intel), T. Kurth (NERSC), A. Walden (ODU)
  - Bread & butter: Dslash, Clover, CG & BiCGStab
- Contraction code for Distillation (J. Chen, JLab)
- QDP-JIT/LLVM for x86 (F. Winter, JLab)
- Strong Collaboration with NERSC NESAP Program
  - Our NESAP contact Thorsten Kurth, already made many valuable contributions to QPhiX:
    - Added out of order receives to Dslash
    - Improved BLAS like kernels in the solvers
  - Next Target: Full x86 QDP-JIT/LLVM+Chroma+QPhiX stack on Cori Phase I.
- Collaboration with Intel
  - JLab had (has) access to a Beta KNL
  - JLab is working with Intel and other community developers to better understand and optimize Chroma for KNL
- Working on/towards Multigrid Implementation

Image Credits: intel.com
Images from Intel's Knight's Landing public disclosures page





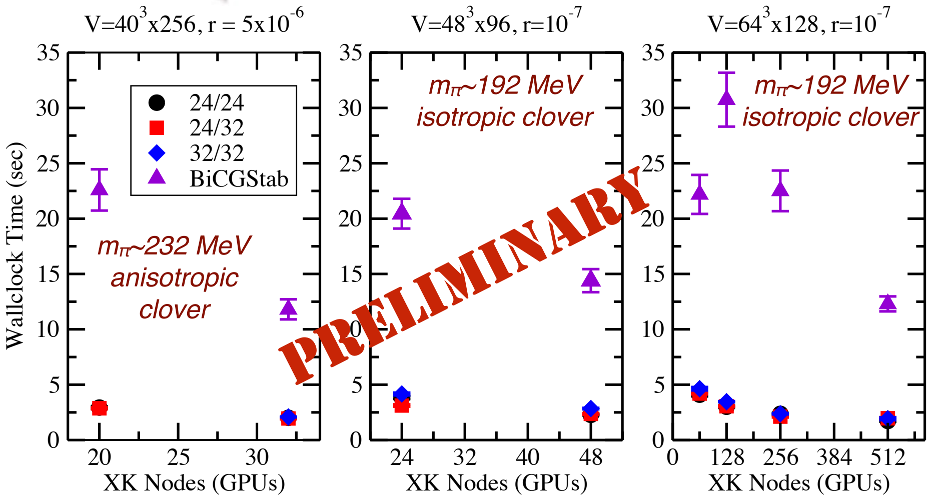
https://software.intel.com/en-us/articles/what-disclosures-has-intel-made-about-knights-landing

GPU Developments

- Code to help NPLQCD Contractions (F. Winter)
- Code for Distillation Contractions (J. Chen)
- MultiGrid for Wilson Clover props
  - CodeFest at JLab in January (K. Clark, J. Chen, R. Edwards, A. Gambhir, B. Joo, A. Strelchenko, W. Watson, F. Winter)
  - AMG Developed in QUDA (K. Clark)
  - Integrated into Chroma for props (B. Joo, A. Gambhir)
  - 6x-10x speedup over QUDA BiCGStab, ~10x over BlueWaters CPU (AMD Bulldozer) code using QOPMG
  - Paper Submitted to SC'16
  - AMG in HMC on drawing board
    - possibly at OLCF GPU Hackathon in October?
- Integrate other QUDA Improvements to Chroma
  - Multi-source (aka Multi-Right Hand Side) solvers

Tesla K80 GPU, Image: nvidia.com

NVIDIA Pascal GPU, Image: anandtech.com

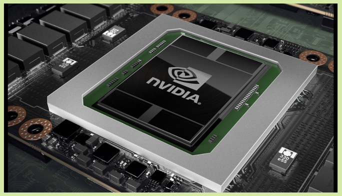


QUDA Clover Multigrid running from Chroma on Titan on configurations of Interest (K20x GPUs)

INTRODUCING TESLA P100

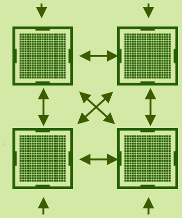
New GPU Architecture to Enable the World's Fastest Compute Node

Pascal Architecture



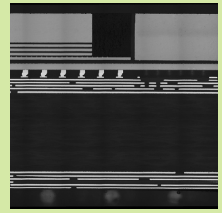
Highest Compute Performance

NVLink



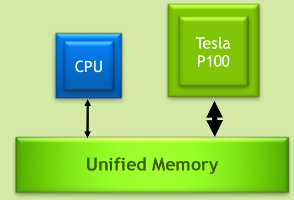
GPU Interconnect for Maximum Scalability

HBM2 Stacked Memory



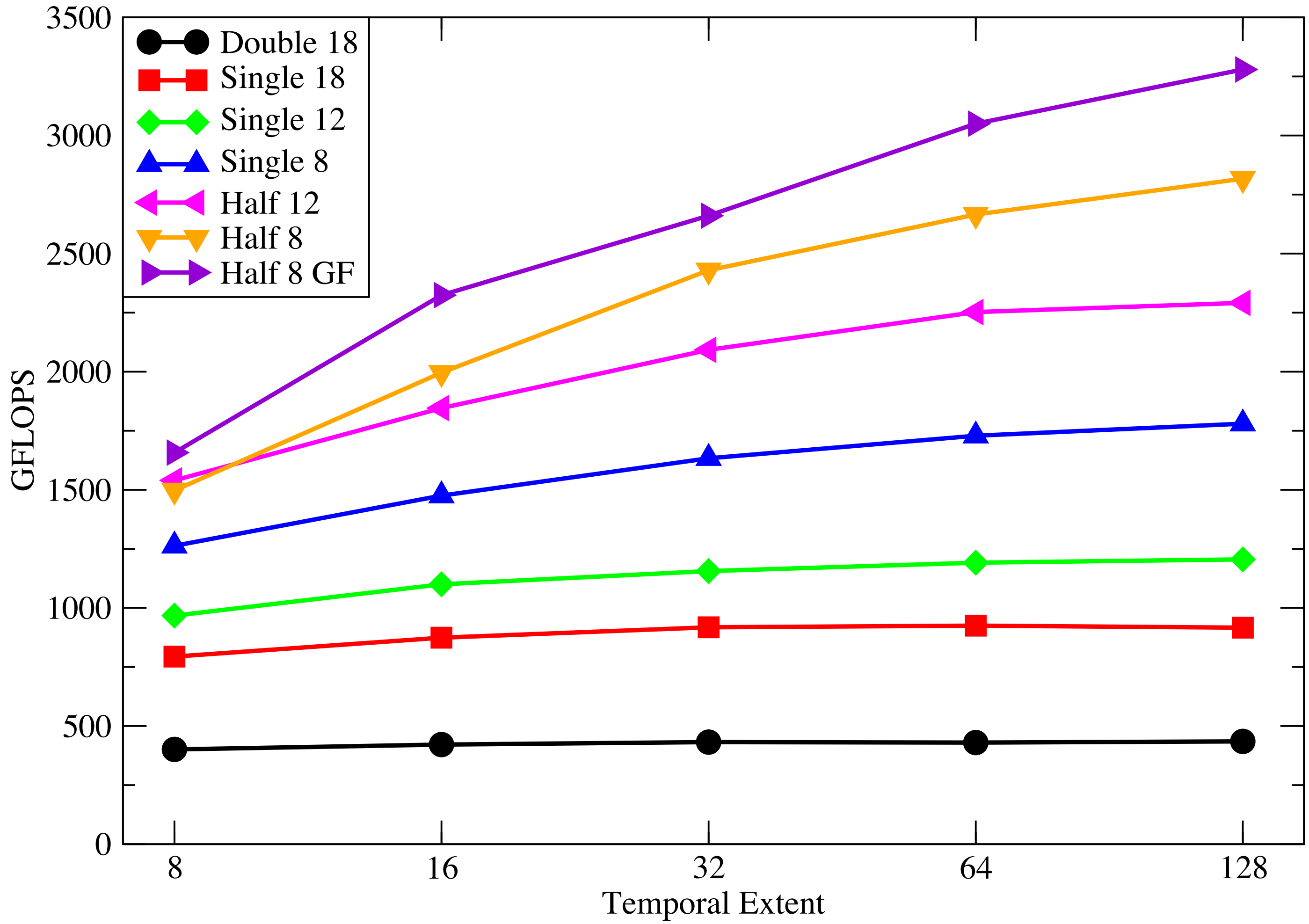
Unifying Compute & Memory in Single Package

Page Migration Engine



Simple Parallel Programming with 512 TB of Virtual Memory

Wilson-clover Dslash performance on Pascal P100



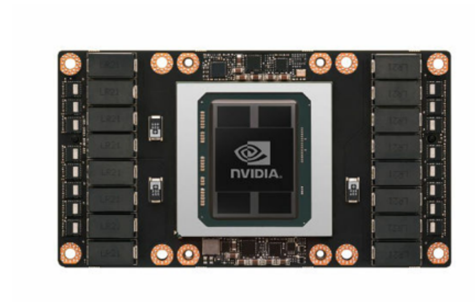
Volume = 24^3 x Lt

Figure form Kate Clark/NVIDIA

MG GPU NOW & COMING SOON

- >10x speedup from MG
  - Lower bound since much more optimization to do
- 6x node-to-node speedup
  - DGX-1 (8x GP100) vs Pi0g (4x K40)
- 3x speedup from multi-src solvers
  - Increased temporal locality from links and increased parallelism for MG
- Expect >100x speedup for analysis workloads versus current GPU workflow

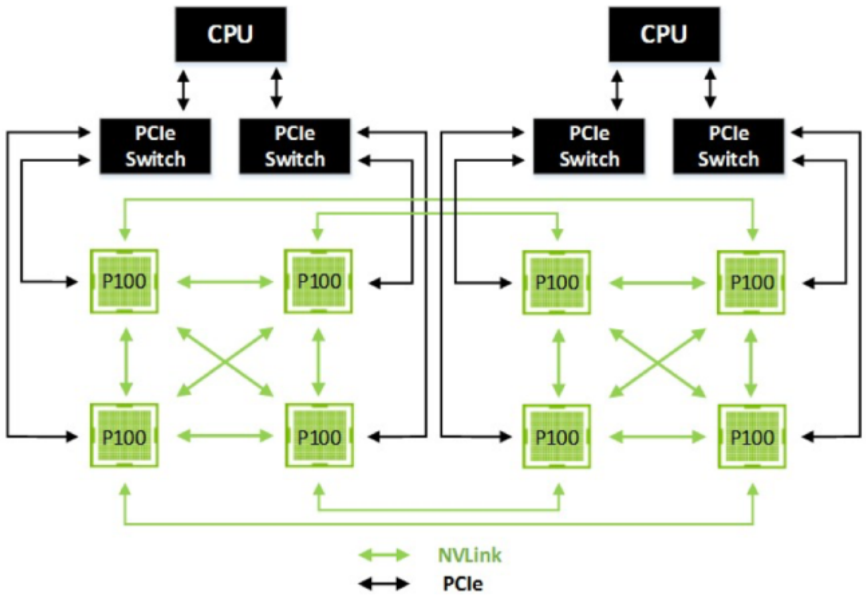
Pascal P100

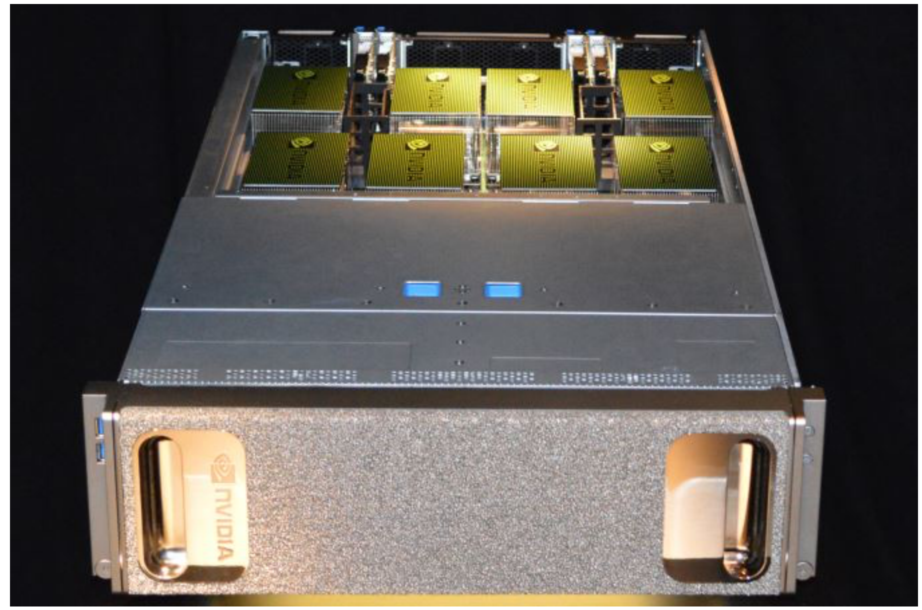


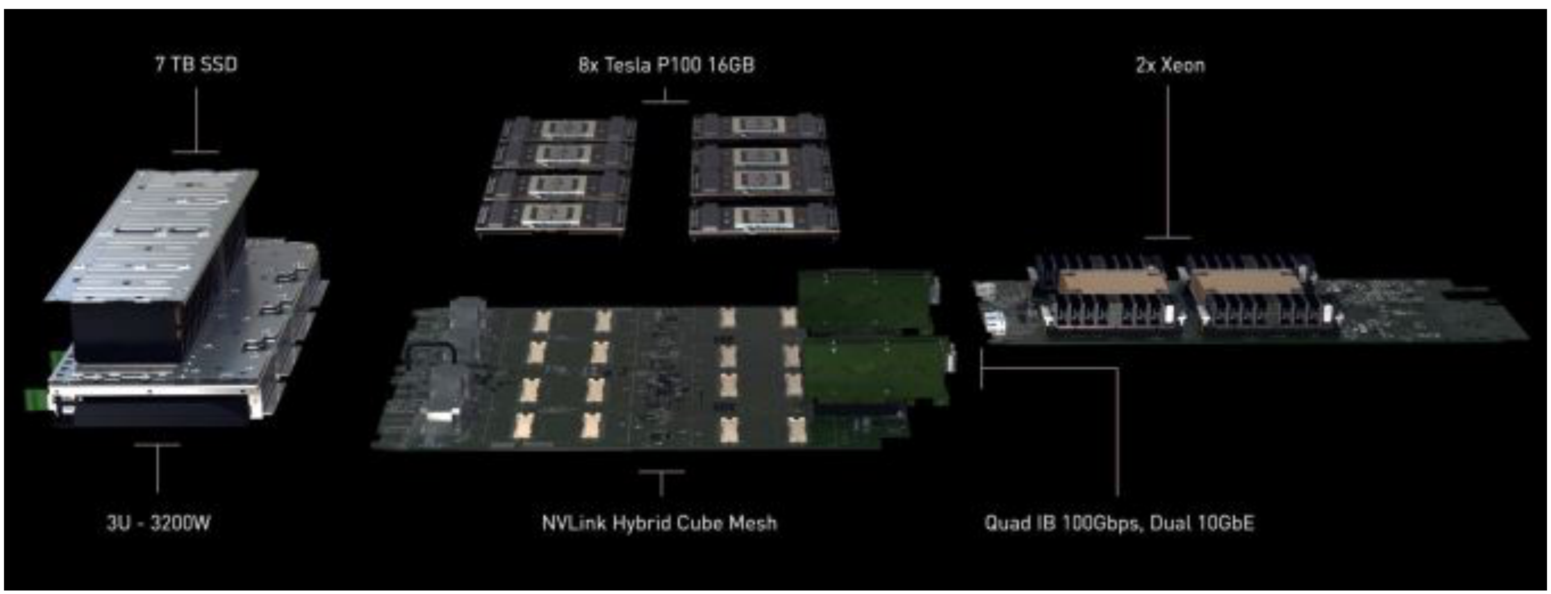
* More optimization is clearly possible: Need similar project on KNL

Alexei, Kate, Evan and I are working on extending this to Multigrid Staggered — we'll see?

DGX-1 8 P100's


CPU
CPU
PCIe Switch
PCIe Switch
PCIe Switch
PCIe Switch
P100
P100
P100
P100
P100
P100
P100
P100
NVLink
PCIe


NVIDIA


7 TB SSD
8x Tesla P100 16GB
2x Xeon
3U - 3200W
NVLink Hybrid Cube Mesh
Quad IB 100Gbps, Dual 10GbE

DWF wth or without Gparity(CPS)

- Grid(https://github.com/paboyle/Grid)(P. Boyle, et. al.):
  Data parallel C++ object library. Divides local volumes to subvolumes. Gather corresponding numbers from each subvolume to fill vectorized types. Aggressive use of new C++11 features. All in C++ except machine specific intrinsics. AVX512 already present.

  Grid SP Mobius CG performance (2016/1)

	BlueWaters	Edison	CoriP1	Babbage
Cores/node	16	24	32	60
Peak(SP) GF/s	627	921	2335	1000
Bidi Network (GB/s)	9.5	11	11.5	
Single node (Gflops/s)	117	265	630	290
8^4 multinode	29	82	88	
16^4 multinode	43	130	190	

  A lot of updates/progress on KNL and beyond, under NDA....
  New vectorization scheme (along Ls) being added to 5d precon. DWF.
  Better network will be very worthwhile!
  Mobius & DSDR CG, Fermion force term and Lanczos with and without Gparity integrated and tested with CPS.
  OpenACC for GPU being explored (M. Lin, C. Kelly)
- Gparity contraction code written with Grid-defined data types + FFT & Lapack. (C. Kelly)

- Exact one-flavor(Chiu et. al.) : Non-Gparity implementation progressing(D. Murphy). Focusing on getting ready for Gparity production on BG/Q in the short term. Will allow more optimization with Hasenbusch, mixed CG..
- Exact deflation: Lanczos can generate $\geq$ 2000 5D eigenvectors with similar number of dslashes as $20 \sim 30$ undeflated CG on 48^3 or 64^3 ensemble. 2000 5d $16^4 \times 12 \sim$ 70GB. Careful profiling showed popular methods for eigenpair calculation (QR, etc) can take a significant portion of total time. Changed it to parallelizable methods (Bisection or DQDS(Lapack)).
- QUDA DWF/Mobius: Latest Double-half DWF CG performance (from K. Clark, Quad K80, $V = 24^4 \times 16$)

	1 GPU	2	4	8
GF/s	561	1096	2104	3932

  Recent optimization with peer-to-peer comms improves scaling significantly on systems with multiple GPUs per node. Further development for zMobius planned. So far without Gparity.
- Vectorized wilson dslash from R-stream(Reservoir, M. Lin, E. Papenhausen)
- Network BW is the limiting factor for evolution, deflation, etc. Finding better ways to distribute work (e.g. Lanczos on CPU + deflated solvers on GPU?) without disk I/O would be beneficial.
- Algorithmic development (Multigrid, delayed deflation, in flight data rearrangement...) are crucial for improving strong scaling further.

Nim & QEX

James C. Osborn & Xiao-Yong Jin

ALCF

USQCD All Hands Meeting
BNL
April 29-30, 2016

Exploring high-level languages for LFT: Nim (nim-lang.org)

- Recently started using Nim to develop new high-level LFT framework (QEX)
- Nim: modern language started in 2008, “efficient, expressive, and elegant”
- Feel of high level scripting language (Python): extensive type inference, but is statically typed systems language (full access to low-level objects & code)
- Generates C or C++ code, then compile with any compiler
  - Easy integration with C/C++: intrinsics (simd), pragmas (OpenMP)
  - GPU support (OpenCL) on roadmap, but probably long ways off
  - LLVM-IR backend recently contributed (still in development)
- Integrated build system (no Makefile necessary): copy main program, modify, compile
- Extensive meta-programming support (nearly full language available at compile time)
  - Transform any Nim code to new Nim code using Nim code
- Openly available on github (MIT license)
- Started by Andreas Rumpf (still main developer)
- 12 contributors with 50+ commits, 89 total in past 2 years
- Soon to be one more

jcosborn commented 3 days ago

This adds a variant of getType that also includes the generic arguments. It was inspired by getType2/getTypeImpl in #3709 but doesn't add any new magics. The name was chosen since the major difference from getType is in the handling of the tyGenericInst node. The intention is also to make the output resemble a syntactically correct declaration of the type, though it wouldn't typically work in a declaration since it doesn't use the correct symbols for the generic types (and I couldn't find a symbol that would work). The output doesn't handle all cases correctly yet, but I wanted to check that it is suitable for inclusion before finishing the rest.

```
import macros
type
  Foo[N:static[int],T] = object
    bar:T
var
  a:Foo[1,float]
  b:Foo[2,Foo[3,tuple[a:int,b:distinct range[-1..4]]]]
macro test(x:typed):auto =
  result = newEmptyNode()
  echo x.getTypeInst.repr
test(a)  # Foo[1, float]
test(b)  # Foo[2, Foo[3, tuple[int, distinct range[-1, 4]]]]
```

jcosborn added some commits 11 days ago

`added getTypeInst which includes generic parameters` 5086e67

`changed getTypeInst handling for distinct types` ✓ c752664

Araq commented 2 days ago — nim-lang member

It's perfectly suitable, great work so far! :-)

QEX (Quantum EXpressions) development plans (https://github.com/jcosborn/qex)

- 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]
```

- Status & plans:
  - framework has full threading and vectorization support
  - have staggered solver & simple analysis running
  - working on finishing basic high-level interface
  - start adding more physics code (HMC, other actions)
  - longer term plans: GPU, refinement/additions to high-level interface, more optimization

- Physics problems
  - actions beyond SU(3) gauge + fundamental fermions
  - new algorithms (with many parameters to tune)
  - Dirac inverters will not dominate computational budget
  - …
- Hardware (driven by heat dissipation)
  - more available flops, vector FPUs
  - less memory bandwidth, higher latency (per Flops)
  - slower network
  - complex memories

- Physics problems
  - actions beyond SU(3) gauge + fundamental fermions
  - new algorithms (with many parameters to tune)
  - Dirac inverters will not dominate computational budget
  - …
- Hardware

It is not known what the next machine will be.

We need to be ready to run on it.

- High level scripting
    - hardware independent
    - segregated from implementation details
    - long term stability

- High level scripting
  - hardware independent
  - segregated from implementation details
  - long term stability
- Target machines
  - networked fat nodes
  - large local memory
  - many (but not a power of 2) cores per node
  - vector FPUs (of unknown vector length)
  - tightly coupled accelerators with separate memory
  - deficient compilers (e.g., OpenMP is evil)
  - use 3rd party components (MPI, HDF5, …)
  - borrow ideas from other fields
    - image processing
    - event-driven computations
    - coherent cache protocols