Machines and algorithms

Peter Boyle
University of Edinburgh
Alan Turing Institute
Brookhaven National Laboratory

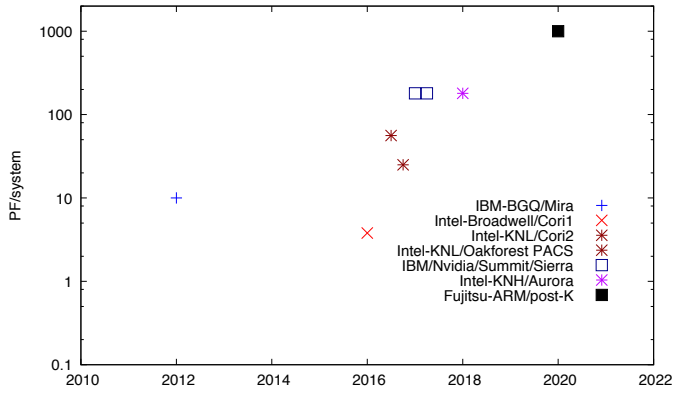
- New processors
- Single node QCD performance results
- Interconnects
- Algorithms

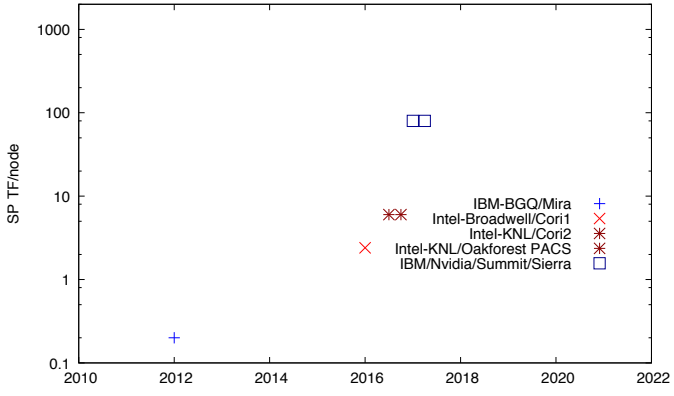
Immediate roadmap

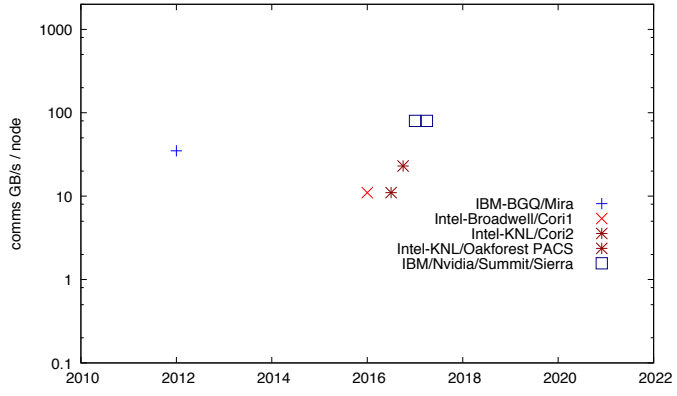
System attributes	NERSC Now	OLCF Now	ALCF Now	NERSC Upgrade	OLCF Upgrade	ALCF Upgrades	
Name Planned Installation	**Edison**	**TITAN**	**MIRA**	**Cori 2016**	**Summit 2017-2018**	**Theta 2016**	**Aurora 2018-2019**
System peak (PF)	2.6	27	10	> 30	150	>8.5	180
Peak Power (MW)	2	9	4.8	< 3.7	10	1.7	13
Total 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®

Immediate roadmap

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®







- 400x increase in SP node performance accompanied by 2x increase in interconnect
- Business as usual is not an option for algorithms

Growing on chip parallelism...

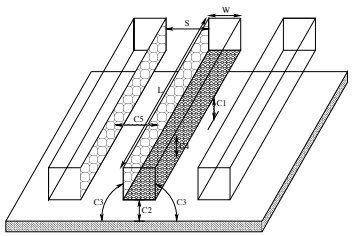
`http://www.agner.org/optimize/`

Core	simd	Year	Vector bits	SP flops/clock/core	cores	flops/clock
Pentium III	SSE	1999	128	3	1	3
Pentium IV	SSE2	2001	128	4	1	4
Core2	SSE2/3/4	2006	128	8	2	16
Nehalem	SSE2/3/4	2008	128	8	10	80
Sandybridge	AVX	2011	256	16	12	192
Haswell	AVX2	2013	256	32	18	576
KNC	IMCI	2012	512	32	64	2048
KNL	AVX512	2016	512	64	72	4608
Skylake	AVX512	2017(?)	512	64	28	1792

- Growth in core counts
- Growth in SIMD parallelism
- Growth in complexity of memory heirarchy
- Interconnect performance failing to grow as fast as processor and memory performance

Standard industry solution is to dump it on the progammer!

Wireloads and geometry



Gate Length (nm)	Dielectric Constant κ	Metal ρ ($\mu\Omega$-cm)	Mid-Level Metal			
			Width (nm)	Aspect Ratio	R_{wire} ($m\Omega/\mu m$)	C_{wire} ($fF/\mu m$)
250	3.9	3.3	500	1.4	107	0.202
180	2.7	2.2	320	2.0	107	0.333
130	2.7	2.2	230	2.2	188	0.336
100	1.6	2.2	170	2.4	316	0.332
70	1.5	1.8	120	2.5	500	0.331
50	1.5	1.8	80	2.7	1020	0.341
35	1.5	1.8	60	2.9	1760	0.348

Simple physics explains computer architecture: model wire as rod of metal $L \times \pi r^2$

- Charge: Gauss's law

$$2\pi rLE = \frac{Q}{\varepsilon}$$

- Resistance

$$R = \rho \frac{L}{\pi r^2}$$

- Capacitance

$$C = Q/V = 2\pi L\varepsilon / log(r_0/r)$$

- Time constant

$$RC = 2\rho\varepsilon \frac{L^2}{r^2} / \log(r_0/r) \sim \frac{L^2}{r^2}$$

RC wire delay depends *only* on geometry: Shrinking does not speed up wire delay!

- "copper interconnect" (180nm) and "low-k" dielectric (100nm) improved ρ and ε

Multi-core design with long-haul buses only possible strategy for 8 Billion transistors

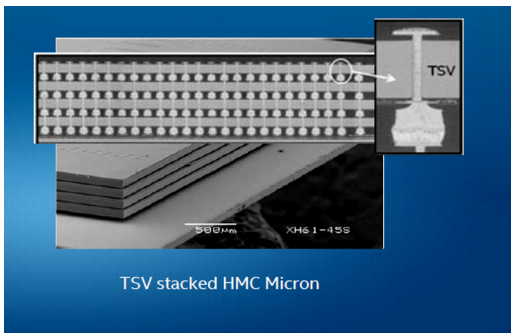
- Low number of long range "broad" wires (bus/interconnect)
- High number of short range "thin" wires

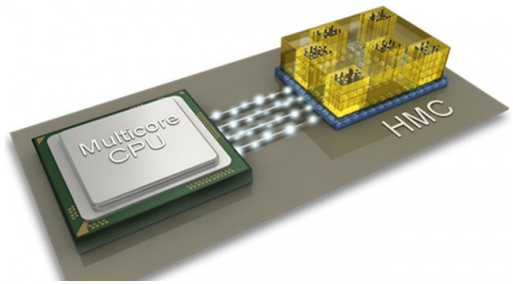
3D integration

- Apply to memory buses with through-silicon-via's (TSVs)!
- 2.5D : Integrate memory stacks on an *interposer* (Intel, Nvidia, AMD)
  In package memory: long thin wires $\rightarrow$ short broad fast wires
- 3D : Direct bond memory stacks to compute (PEZY, mobile, Broadcom)
  3D memory could grow the bus widths almost arbitrarily

Massive replica counts from silicon lithography compared to macroscopic assembly

There's plenty of room at the bottom (Feynman); Avagadro's number is big!

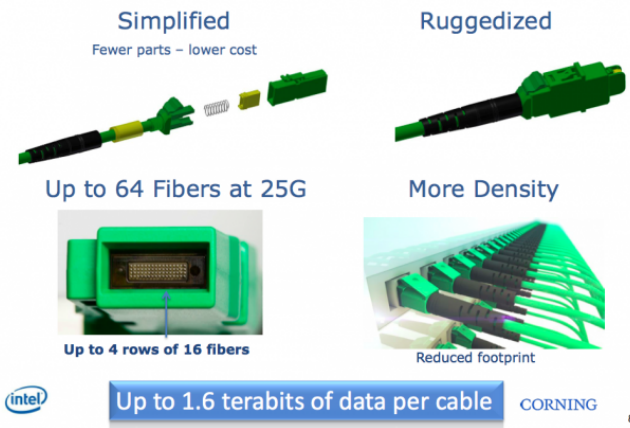




- This years tech:
  - 16 GB (AXPY 400 GB/s) Intel Knights Landing (KNL)
  - 16-32 GB (AXPY 600 GB/s) Nvidia Pascal P100
  - Regular Xeon ... when ?

Other trends in microelectronics

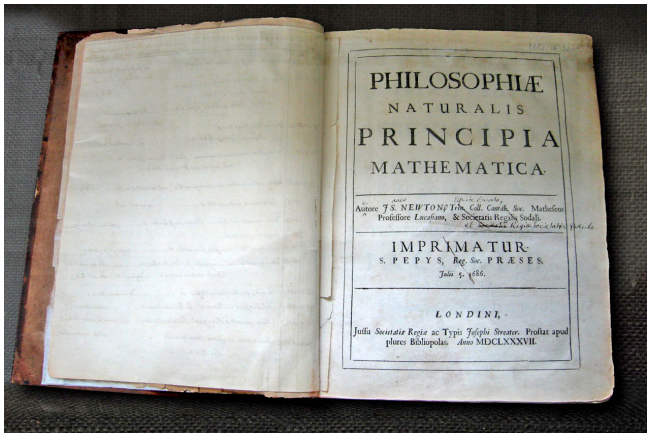
- Novel non-volatile memory, NVDIMM's
  - Phase change memory (amorphous/crystaline glass cell) should increase memory density. Micron/Intel 3D Xpoint branding: 4x higher density than DRAM; SSD's → NVDIMMs eetimes.com says it is PCM
  - Multiple JEDEC NVDIMM approaches
  - Disruptive for large memory applications (e.g. eigenvectors, multi-hadron)
- Integration of network
  - KNL-F will integrate 2 x 100Gbit/s Omnipath 1 network on package (50 GB/s bidi)
  - KNH will integrate Omnipath 2 network on die (Aurora)
  - Skylake will have integration with Omnipath (Intel SSF)
  - NVLink scales to 8 GPU's 160GB/s bidi; *not* a cluster interconnect
- Silicon photonics
  - 100Gbit/s copper cables cost under 100 USD
  - 100Gbit/s active optical cables cost around 1000 USD
  - reduce the cost and power of driving fibre cable to be closer to cost of copper use a normal silicon process for laser components
    - Hope for active optical ⟶ passive optical in future
    - Room sized networks will necessarily remain macroscopic and a problem



Computing basics

Computers retain data in *registers* and *memory*

- Registers are like the store/recall buttons in a calculator
- Memory is an indexable paper-pad for retaining values

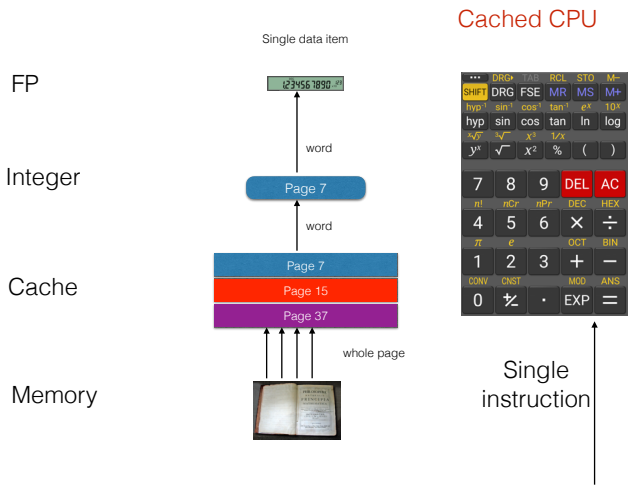


- Processors are merely state machines, containing internal variables (registers) and a current instruction address
- Von Neumann machines
  1. Fetch instruction from memory address pointed to by instruction pointer
  2. Interpret and carry out instruction
     Modifies registers or memory as appropriate
  3. Update instruction pointer (increment or branch)
  4. Goto 1
- What if memory access takes 500 cycles ?

Caches and locality

Text book computer engineering: (e.g. Hennessy & Patterson)

- Code optimisations should expose spatial data reference locality – Large cacheline, wide buses
- Code optimisations should expose temporal data reference locality – Large cache



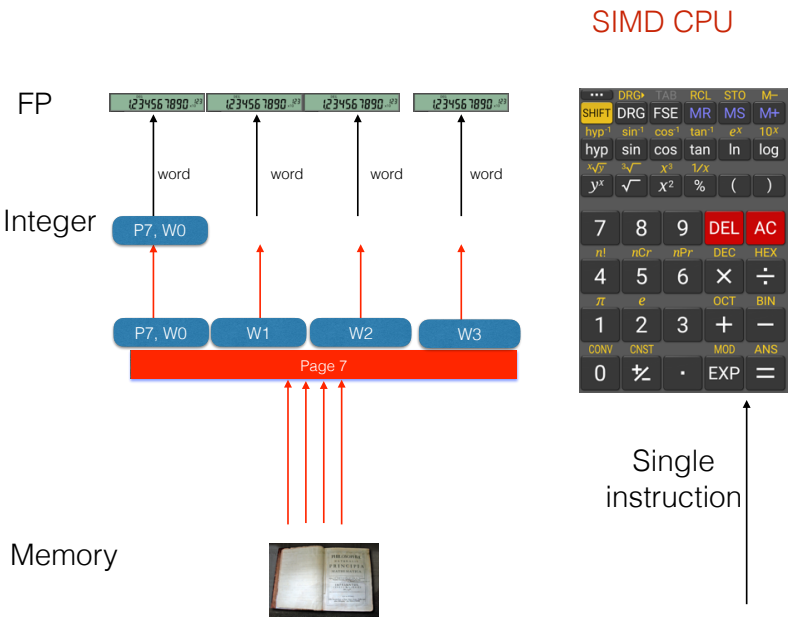
- Memory systems are *granular*
  - If you only access 1 byte of contiguous data, you still pay to transfer 128Bytes
  - Big gain from *spatial locality* of reference: use everything that gets transfered!

You don't buy a multipack if you only want one item!

CPU SIMD model

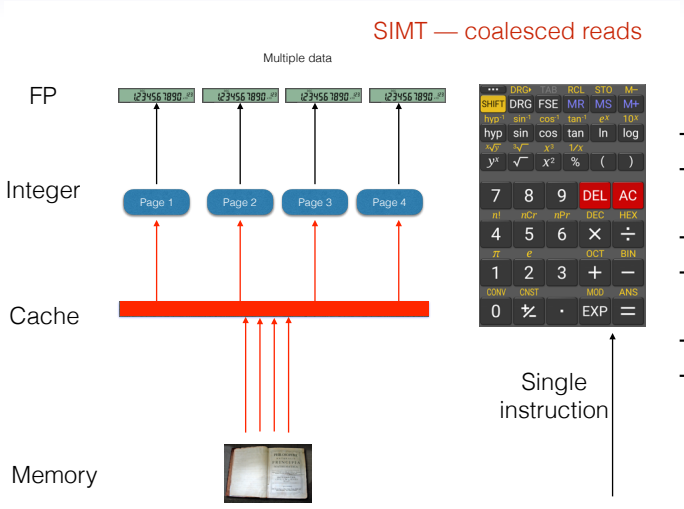
SIMD brings a new level of restrictivness that is much harder to hit

- Code optimisations should expose *spatial operation locality*
- Obvious applications in array and matrix processing but hard in general



- Must arrange to have same operation applied to consecutive elements of data
  - Only then can granular memory transfers *and* SIMD execution be exploited
  - typically drop to "intrinsic functions" or assembly for CPU's
  - change data layout from the standard language defined array ordering
  - we are fighting against the languages!

GPU SIMT model



	CPU	GPU
Integer	scalar	vector
Floating	vector	vector
Caching	yes	yes
Contiguous vec loads	default	dynamically coalesced
Random vec loads	gather/scatter	default
Thread divergence	write masks	hardware managed

- Any grouping of data references works
- **For performance** must arrange to have same operation applied to consecutive elements of data
  - *Coalesced accesses* detected at runtime by GPU's
  - granular memory transfers *and* SIMD execution can then be exploited
  - Performance loss if threads diverge in address or control flow
- Vectorisable loop ordering and data layout *identical* between GPU and CPU

contiguous block memory accesses with same operations performed on adjacent words.

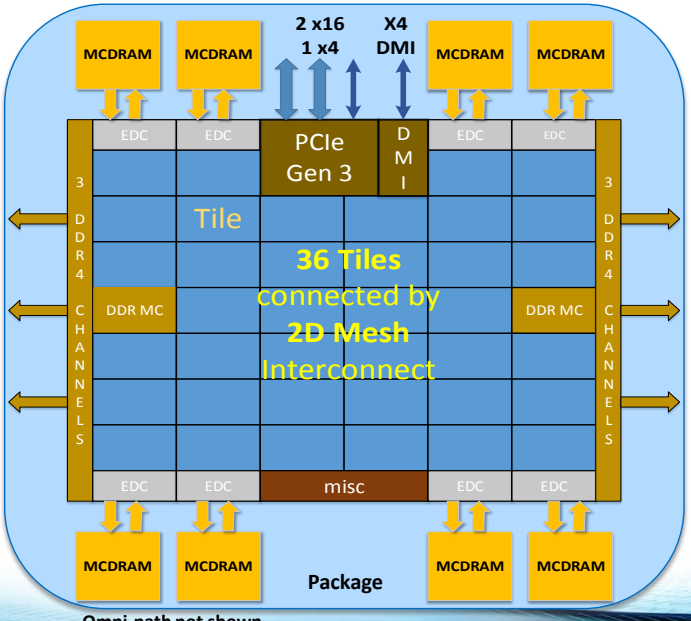
Chip	Clock	blocks	per bock	SP madd	issue	SP madd	peak
GP100	1.4 GHz	56 SM's	2 IB/RF	32	112×2	3584	10.5 TF/s
KNL	1.4 GHz	36 L2 tiles	2 cores	32	72×2	2304	6.4 TF/s
Broadwell	2.5		18 cores	16	18×2	576	1.4 TF/s
Skylake	?		28 cores	32	28×2	1792	4.4 TF/s (EST)

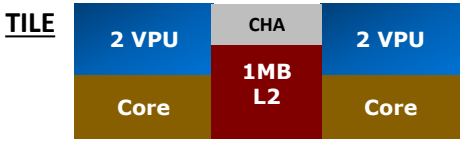
Intel Knight's Landing Deep Dive

Intel HotChips Talk Hyperlink

- 2016 NERSC (Cori-II), Argonne (Theta) with Cray Aries
- 2016 Cineca (Marconi), Tsukuba/Tokyo (Oakforest-PACS) with Omnipath
- 2018/19 Aurora (Knights Hill, Omnipath 2.0)

Knights Landing Overview





Chip: 36 Tiles interconnected by **2D Mesh**
Tile: 2 Cores + 2 VPU/core + 1 MB L2

Memory: MCDRAM: 16 GB on-package; High BW
DDR4: 6 channels @ 2400 up to 384GB
IO: 36 lanes PCIe Gen3. 4 lanes of DMI for chipset
Node: 1-Socket only
Fabric: Omni-Path on-package (not shown)

Vector Peak Perf: 3+TF DP and 6+TF SP Flops
Scalar Perf: ~3x over Knights Corner
Streams Triad (GB/s): MCDRAM : 400+; DDR: 90+

Source Intel: All products, computer systems, dates and figures specified are preliminary based on current expectations, and are subject to change without notice. KNL data are preliminary based on current expectations and are subject to change without notice. 1Binary Compatible with Intel Xeon processors using Haswell Instruction Set (except TSX). 2Bandwidth numbers are based on STREAM-like memory access pattern when MCDRAM used as flat memory. Results have been estimated based on internal Intel analysis and are provided for informational purposes only. Any difference in system hardware or software design or configuration may affect actual performance.

Intel Knight's Landing Deep Dive

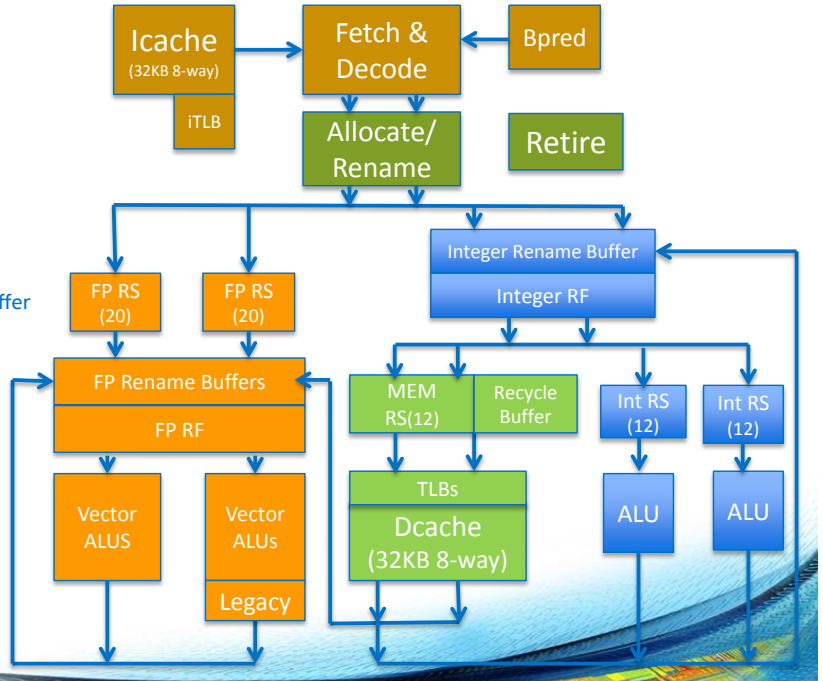
Core & VPU

- Out-of-order core w/ 4 SMT threads
- VPU tightly integrated with core pipeline

- 2-wide Decode/Rename/Retire
- ROB-based renaming. 72-entry ROB & Rename Buffers
- Up to 6-wide at execution
- Int and FP RS OoO.
- MEM RS inorder with OoO completion. Recycle Buffer holds memory ops waiting for completion.
- Int and Mem RS hold source data. FP RS does not.

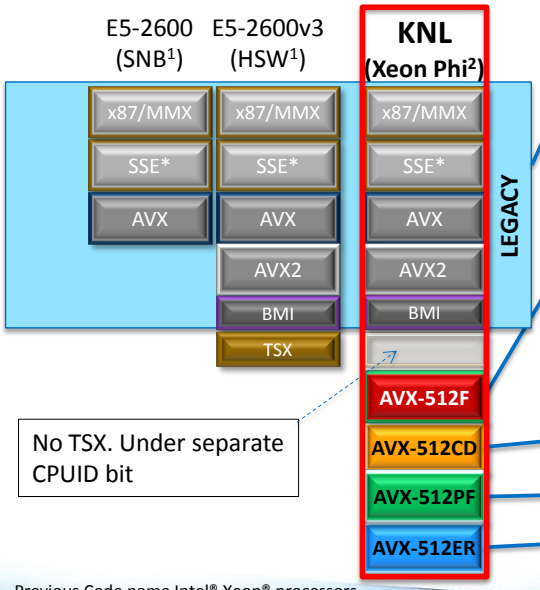
- 2x 64B Load & 1 64B Store ports in Dcache.
- 1st level uTLB: 64 entries
- 2nd level dTLB: 256 4K, 128 2M, 16 1G pages

- L1 Prefetcher (IPP) and L2 Prefetcher.
- 46/48 PA/VA bits
- Fast unaligned and cache-line split support.
- Fast Gather/Scatter support



Intel Knight's Landing Deep Dive

KNL ISA



KNL implements all legacy instructions

- Legacy binary runs w/o recompilation
- KNC binary requires recompilation

KNL introduces AVX-512 Extensions

- 512-bit FP/Integer Vectors
- 32 registers, & 8 mask registers
- Gather/Scatter

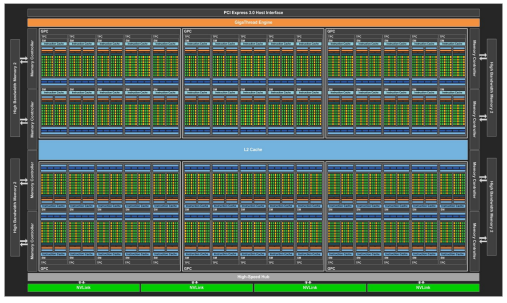
Conflict Detection: Improves Vectorization

Prefetch: Gather and Scatter Prefetch

Exponential and Reciprocal Instructions

1. Previous Code name Intel® Xeon® processors
2. Xeon Phi = Intel® Xeon Phi™ processor

Nvidia Pascal Deep Dive



Tesla Products	Tesla K40	Tesla M40	Tesla P100
GPU	GK110 (Kepler)	GM200 (Maxwell)	GP100 (Pascal)
SMs	15	24	56
TPCs	15	24	28
FP32 CUDA Cores / SM	192	128	64
FP32 CUDA Cores / GPU	2880	3072	3584
FP64 CUDA Cores / SM	64	4	32
FP64 CUDA Cores / GPU	960	96	1792
Base Clock	745 MHz	948 MHz	1328 MHz
GPU Boost Clock	810/875 MHz	1114 MHz	1480 MHz
Peak FP32 GFLOPs[1]	5040	6840	10600
Peak FP64 GFLOPs[1]	1680	210	5300
Texture Units	240	192	224
Memory Interface	384-bit GDDR5	384-bit GDDR5	4096-bit HBM2
Memory Size	Up to 12 GB	Up to 24 GB	16 GB
L2 Cache Size	1536 KB	3072 KB	4096 KB
Register File Size / SM	256 KB	256 KB	256 KB
Register File Size / GPU	3840 KB	6144 KB	14336 KB
TDP	235 Watts	250 Watts	300 Watts
Transistors	7.1 billion	8 billion	15.3 billion
GPU Die Size	551 mm^2	601 mm^2	610 mm^2
Manufacturing Process	28-nm	28-nm	16-nm FinFET

[1] The GFLOPS in this chart are based on GPU Boost Clocks.

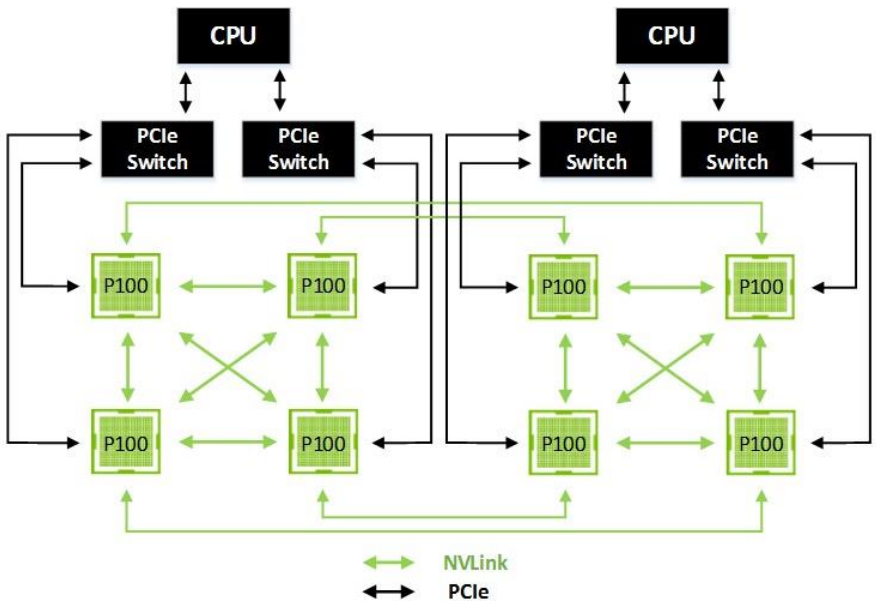
Nvidia Pascal Deep Dive

- Pascal uses virtual memory pages "Page Migration Engine"
- Programming model simplification, less reliant on exposed offload
  - Pulls pages from CPU vm system on demand, locks out CPU
  - With O/S support distinction between host and device memory eroded
  - Call special allocator to access from *both* host and device
- NVLink provides 160 GB/s bidi interconnect for up to 8 GPU's (DGX-1)
- Some tech press sites (trustable?) say next generation Volta will become cache coherent

Pascal Unified Memory*

```
void sortfile(FILE *fp, int N) {
  char *data;
  data = (char *)malloc(N);

  fread(data, 1, N, fp);

  qsort<<<...>>>(data,N,1,compare);
  cudaDeviceSynchronize();

  use_data(data);

  free(data);
}
```

*with operating system support



Fine grid Dirac matrix bandwidth analysis

- L^4 local volume; 8/16 point stencil
  - Multi-RHS and DWF take $L_s = N_{\rm rhs}$.
    Suppresses gauge field overhead;
  - Cache reuse $\times N_{\rm stencil}$ on Fermion possible

- Accesses per 4d site of result
  - Fermion: $N_{\rm stencil} \times (N_s \in \{1,4\}) \times (N_c = 3) \times (N_{\rm rhs} \in \{1,16\})$ complex
  - Gauge: $2N_d \times N_c^2$ complex
- Flops
  - $N_{\rm stencil} \times N_{hs}$ SU(3) MatVec: $66 \times N_{\rm hs} \times N_{\rm stencil}$ (+ spin projection)

Action	Fermion Vol	Surface	N_s	N_{hs}	$N_{\rm rhs}$	Flops	Bytes	Bytes/Flops
HISQ	L^4	$3\times8\times L^3$	1	1	1	1146	1560	1.36
Wilson	L^4	$8\times L^3$	4	2	1	1320	1440	1.09
DWF	$L^4\times N$	$8\times L^3$	4	2	16	$N_{\rm rhs}\times1320$	$N_{\rm rhs}\times864$	0.65
Wilson-RHS	L^4	$8\times L^3$	4	2	16	$N_{\rm rhs}\times1320$	$N_{\rm rhs}\times864$	0.65
HISQ-RHS	L^4	$3\times8\times L^3$	1	1	16	$N_{\rm rhs}\times1146$	$N_{\rm rhs}\times408$	0.36

- $\sim \frac{1}{L}$ of data references come from off node

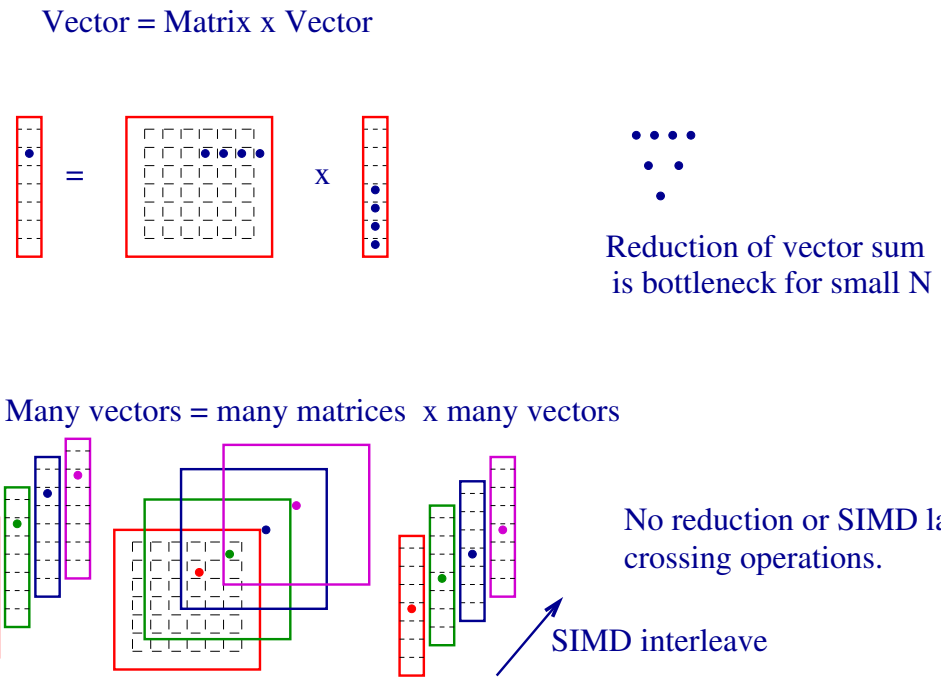
Scaling fine operator requires interconnect bandwidth

$$B_{network} \sim \frac{B_{memory}}{L} \times R$$

where R is the *reuse* factor obtained for the stencil in caches

Intel Knight's Landing Performance Results

Vectorisation strategy



- SIMD most efficient for *independent but identical* work
- Apply N small dense matrix-vector multiplies in parallel

Back to the Future

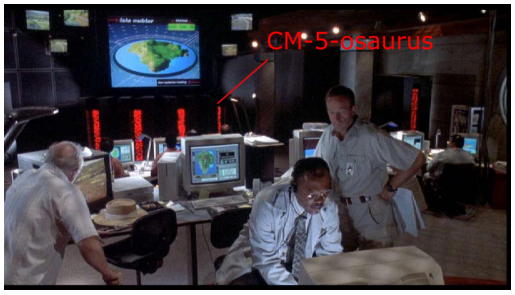
Q) How do we find copious *independent but identical* work?

A) Remember that SIMD was NOT hard in the 1980's (CM, APE...)

Connection Machine Model CM-2 and DataVault System

The Connection Machine Model CM-2 uses thousands of processors operating in parallel to achieve peak processing speeds of above 10 gigaflops. The DataVault mass storage system stores up to 60 gigabytes of data.

- Resurrect Jurassic data parallel programming techniques: **cmfortran, HPF**
- Address SIMD, OpenMP, MPI with *single* data parallel interface
  - Map arrays to virtual nodes with user controlled layout primitives
  - Conformable array operations proceed data parallel with 100% SIMD efficiency
  - CSHIFT primitives handle communications



GRID data parallel template library

Ordering	Layout	Vectorisation	Data Reuse
Microprocessor	Array-of-Structs (AoS)	Hard	Maximised
Vector	Struct-of-Array (SoA)	Easy	Minimised
Bagel	Array-of-structs-of-short-vectors (AoSoSV)	Easy	Maximised

- www.github.com/paboyle/Grid
- PAB, Cossu, Portelli, Yamaguchi: arXiv:1512.03487; Poster 184
- Automatically transform layout of arrays of mathematical objects using vSIMD template parameter
- Conformable array operations are data parallel on the *same* Grid layout

```
vRealF, vRealD, vComplexF, vComplexD

template<class vtype> class iScalar
{
    vtype _internal;
};
template<class vtype,int N> class iVector
{
    vtype _internal[N];
};
template<class vtype,int N> class iMatrix
{
    vtype _internal[N][N];
};

typedef Lattice<iMatrix<vComplexD> > LatticeColourMatrix;
typedef iMatrix<ComplexD>  ColourMatrix;
```

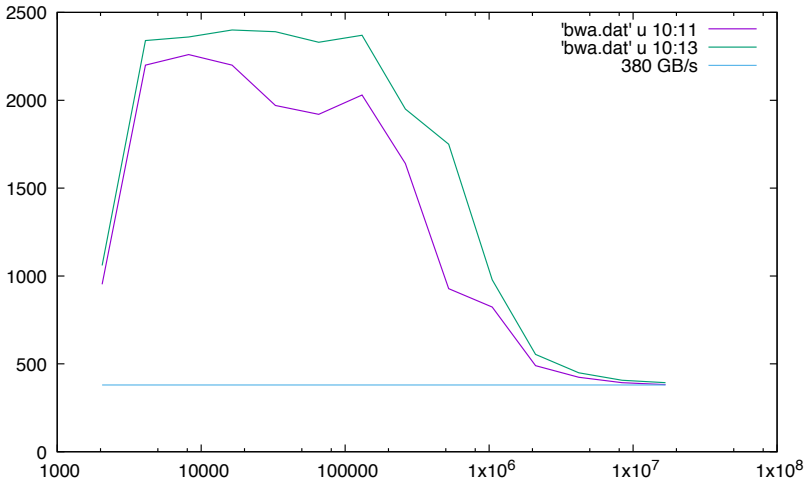
- Internal type can be SIMD vectors *or* scalars

```
LatticeColourMatrix A(Grid);
LatticeColourMatrix B(Grid);
LatticeColourMatrix C(Grid);
LatticeColourMatrix dC_dy(Grid);

C = A*B;

const int Ydim = 1;

dC_dy = 0.5*Cshift(C,Ydim, 1 )
      - 0.5*Cshift(C,Ydim,-1 );
```

- *High-level* data parallel code gets 65% of peak on AVX2
- Single data parallelism model targets BOTH SIMD and threads efficiently.

Grid performance

Architecture	Cores	GF/s (Ls x Dw)	peak
Intel Knight's Landing 7250	68	770	6100
Intel Knight's Corner	60	270	2400
Intel Broadwellx2	36	800	2700
Intel Haswellx2	32	640	2400
Intel Ivybridgex2	24	270	920
AMD Interlagosx4	32 (16)	80	628



Knight's Landing memory system profile (GB/s vs footprint bytes/core)

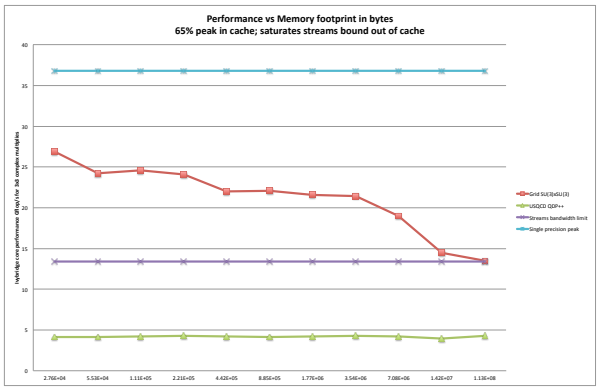
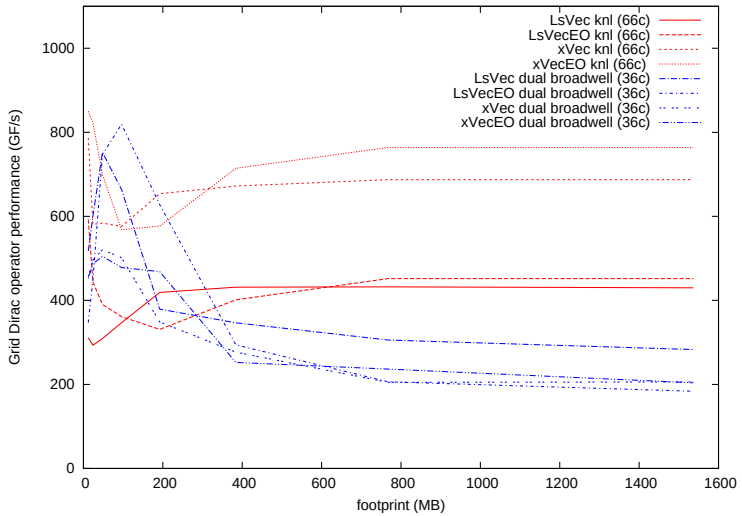


Figure 4: We compare the performance of Grid (red) on SU(3)×SU(3) matrix multiplication to peak (blue), the limit imposed by memory bandwidth (purple), and to that of the QDP++ code system (green).

SU3 x SU3 example

Grid multi-RHS Wilson Dslash and DWF

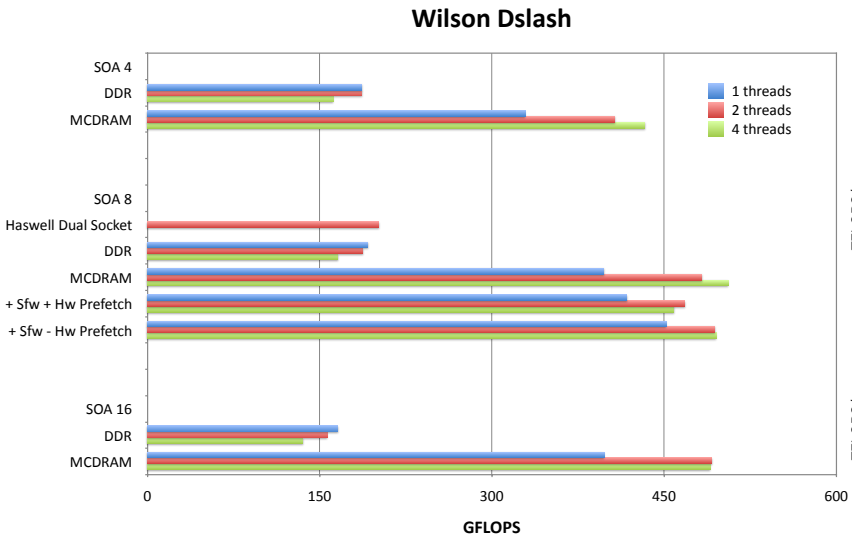


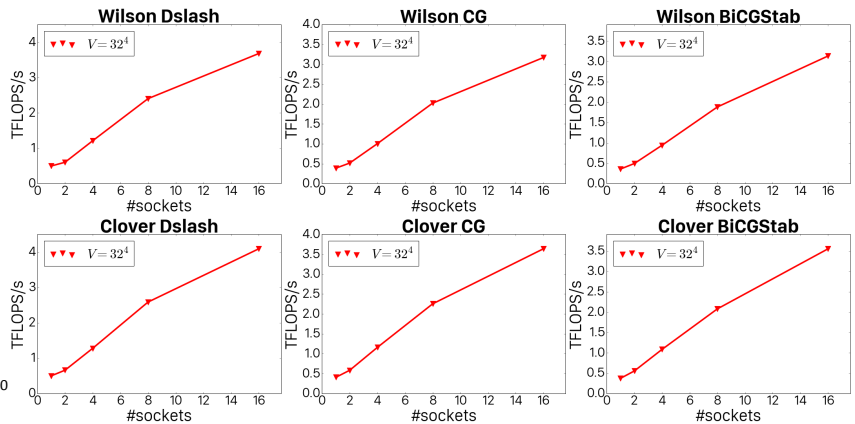
- Grid single node, single precision performance for multiRHS Wilson term
- Knight's Landing 7250, 68 core
  - Used 66 cores - a few empty cores usually faster
- One KNL substantially faster than two Broadwell's (18+18) out of cache

- 1 thread per core fastest after writing in assembler (*not* intrinsics)
  - Macro system and mixed C++/asm minimises pain
  - Hand allocation of registers evades stack eviction, cache more deterministic
  - Hand prefetch to L2 and to L1
  - 8.2.2.2 cache blocking
  - Less reuse than I hoped for
- Single core instructions-per-cycle is 1.7 (85% of theoretical)
- Multi-core L1 hit rate is 99% (perfect SFW prefetching)
- Multi-core MCDRAM bandwidth 97% (370GB/s)
- Provably unimprovable ?

QPhiX performance on KNL and broadwell

- "Optimizing Dirac Wilson Operator and linear solvers for Intel(R) KNL"
  B. Joo - Jefferson Lab, Newport News, VA, USA
  D. D. Kalamkar - Intel Parallel Computing Labs, India T. Kurth - NERSC
  K. Vaidyanathan - Intel Parallel Computing Labs, India
  A. Walden - Old Dominion University, Norfolk, VA, USA

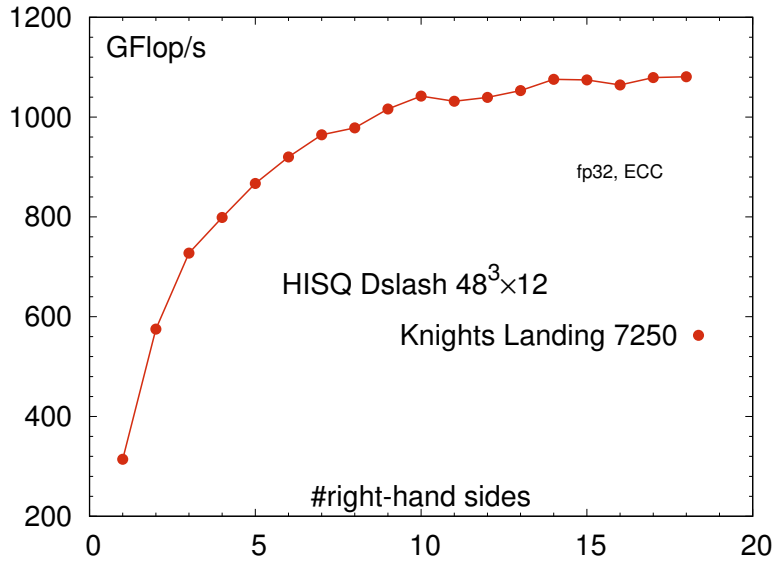




- Single precision
- first results I have seen on multi-node performance with Omnipath

Multi RHS single node HISQ

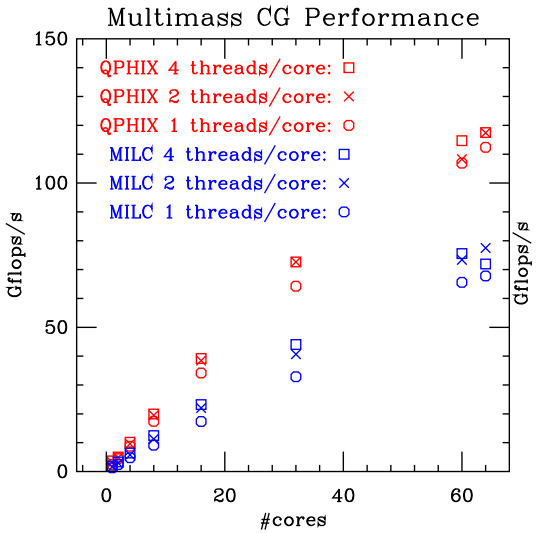
Patrick Steinbrecher, PhD student, Bielefeld & BNL

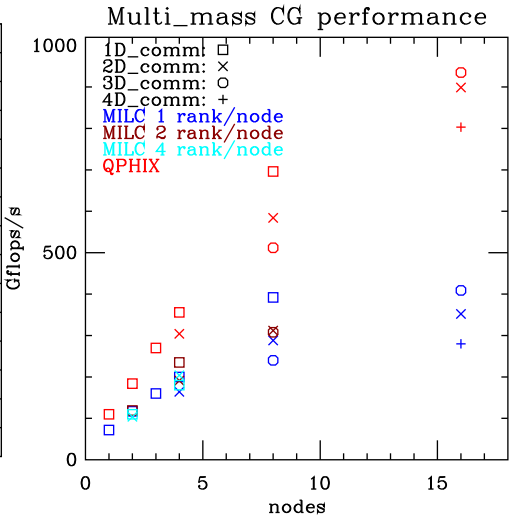


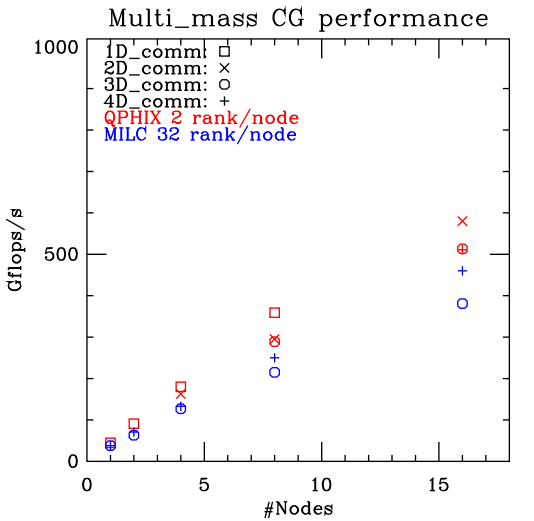
- Single precision, single node performance library for valence measurement
- Particular emphasis on use in disconnected diagrams at finite temperature
- Adopted similar wrapping of vector classes to Grid approach
- Patrick has done a really good job

MILC on KNL

- MILC multi-mass CG (Ruizi Li, Thursday@15:40)
- Double precision







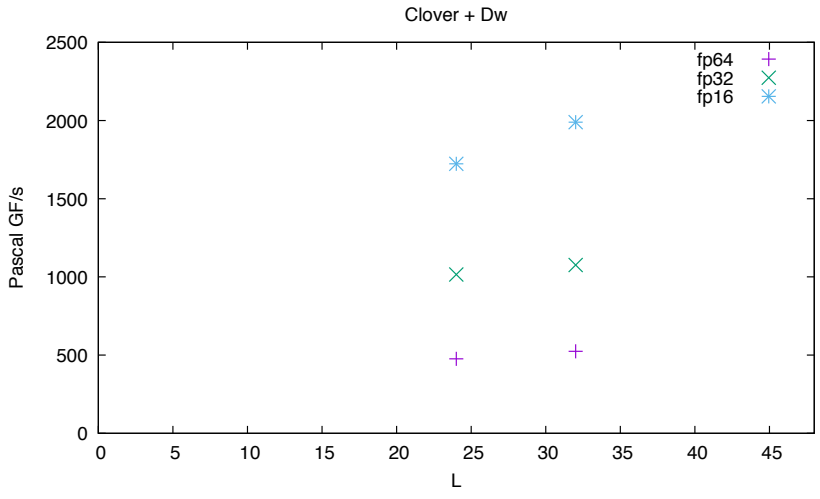
- Also: Ishikawa-san KNC results Thursday@15:20

Nvidia Pascal Performance Results

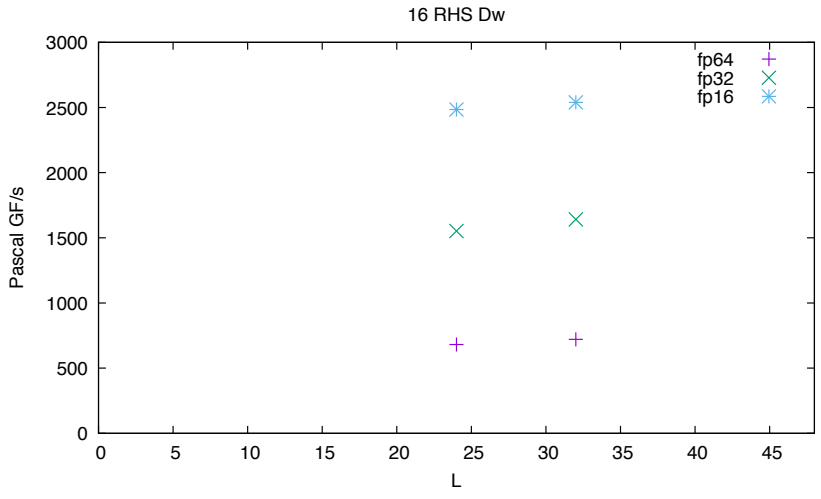
QUDA performance

- Algorithms and machines, Thu @ 14:20, Wagner, Thu @ 14:40, Clark

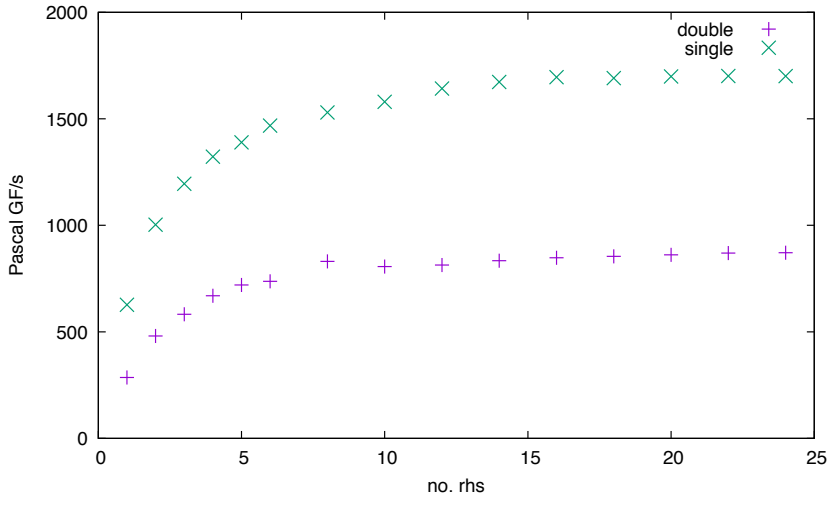
Pascal results and code by Kate Clark, Nvidia



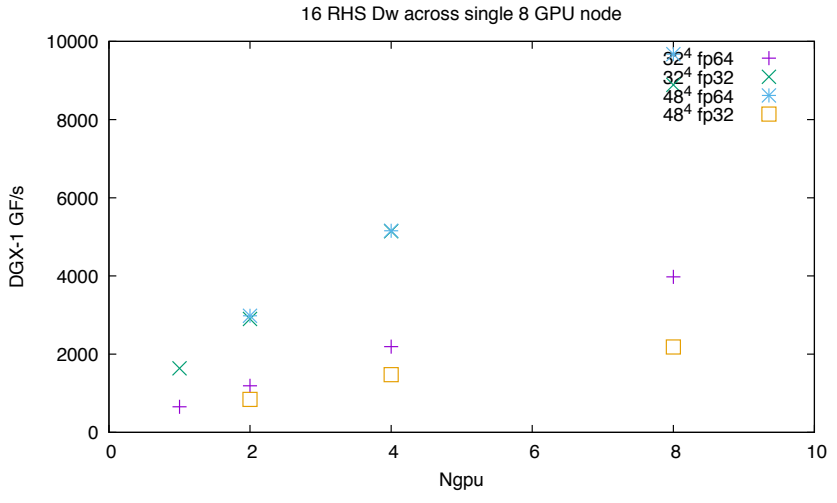
Clover term + D_W



16 RHS D_W



Multi RHS Staggered



Scaling across DGX-1 8 gpu system

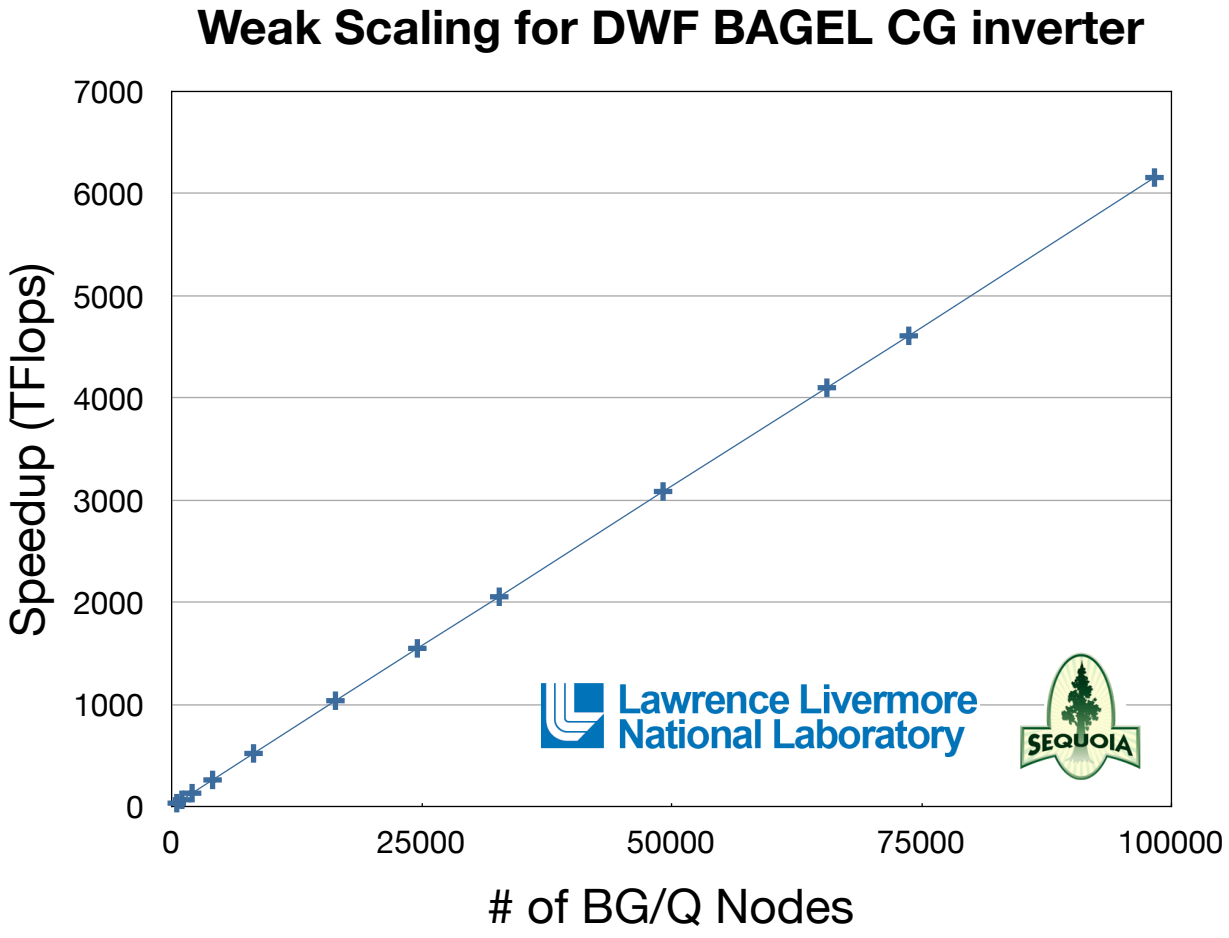
Programming more generally for GPU

- **Offload to GPU poses difficulty to code maintainance and performance portability**
  - Big investment in QDP-JIT for example
  - PoS LATTICE2011 50 (Winter)
    PoS LATTICE2012 (2012) 185 (Winter)
  - "*operator* =" prints simple GPU code, compiles, dynamic links, caches
- **PB, Meifeng Lin reduced Grid ET engine to 200 line example**
  - Remove use of C++ libraries in assembling "Expression objects"
  - Remove host references in expression objects
- Can offload with CUDA kernel call to evaluate expression using "compile time compilation"
- Will become even easier with unified memory model

```
initialised arrays
v1={1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1}
v2={2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2}
v3 = v1+v2 = {3,3,3,3,3,3,3,3,3,3,3,3,3,3,3,3}
v3 = v1+v2+v1*v2 = {5,5,5,5,5,5,5,5,5,5,5,5,5,5,5,5}
```

- **James Osborn** QEX based on "NIM" language; controllable "nim to C" mapping in principle enables GPU translation
  Algorithms and Machines, Thursday@14:00

Reminder of BG/Q scalability



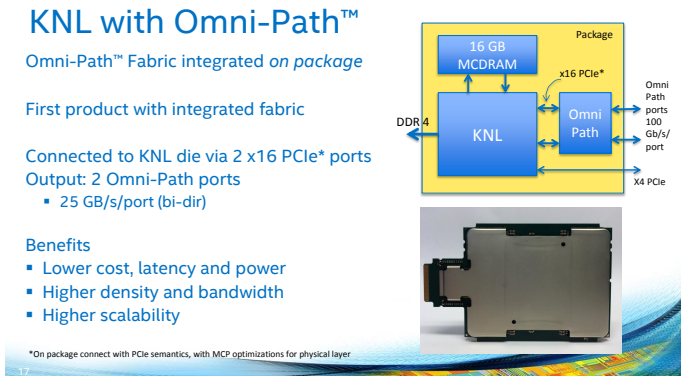
Code developed by Peter Boyle at the STFC funded DiRAC facility at Edinburgh

- RBC-UKQCD simulation programme has regularly sustained over 1 PF/s on MIRA

Interconnect

$$B_{network} \sim \frac{B_{memory}}{L} \times R$$

- Determine reuse factor via $B_{network} = P_{dwf} * 0.65/L$
- This is the *reception bandwidth*, and double this required bidirectional
- Integration of 2x100 Gbit/s network ports on KNL package significant
- Integration of Omnipath-2 on KNH is significant
- Results from Edison, Cori Phase-1, and by Silicon Graphics

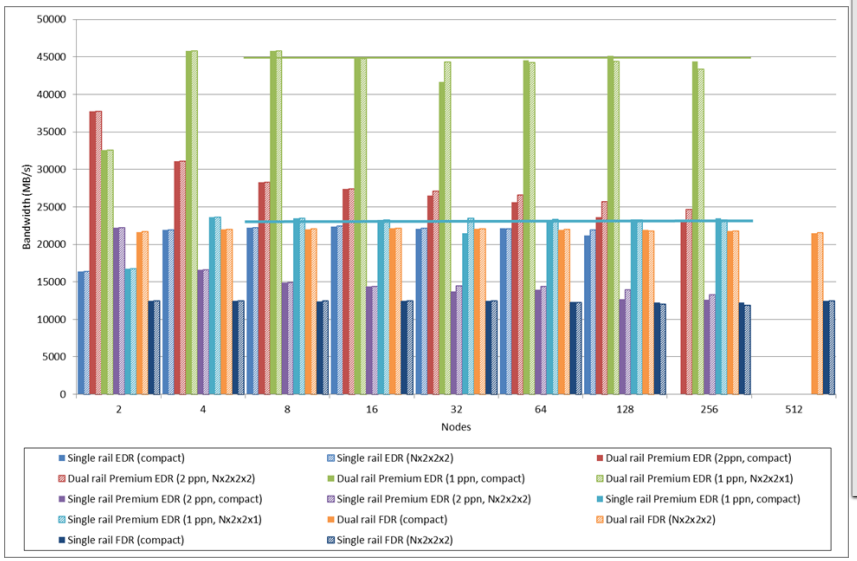


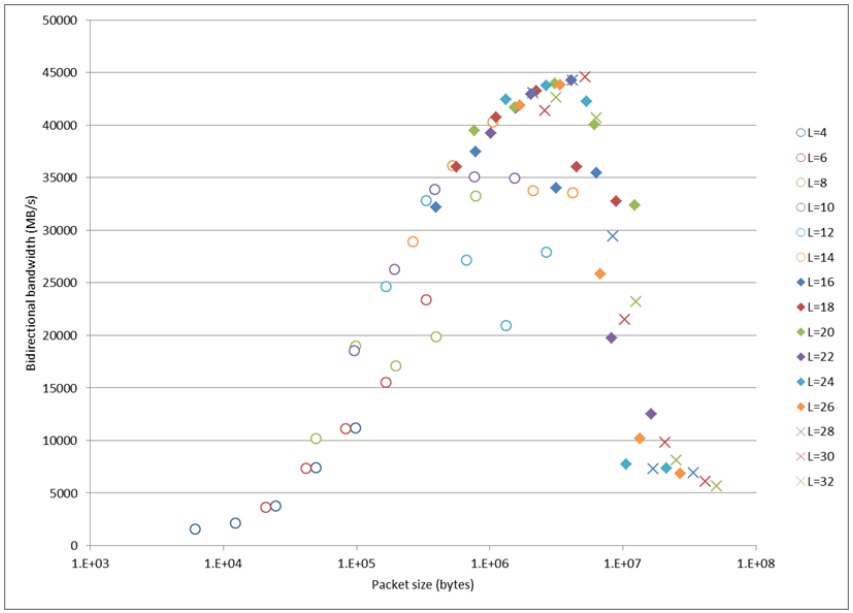
Nodes	Memory (GB/s)	Bidi network requirement (GB/s)			
		L=10	L=16	L=32	L=64
2xBroadwell	100	100	16	8	
KNL	400	100	64	32	
P100	700	200	128	64	
DGX-1	5600	-	975	487	243

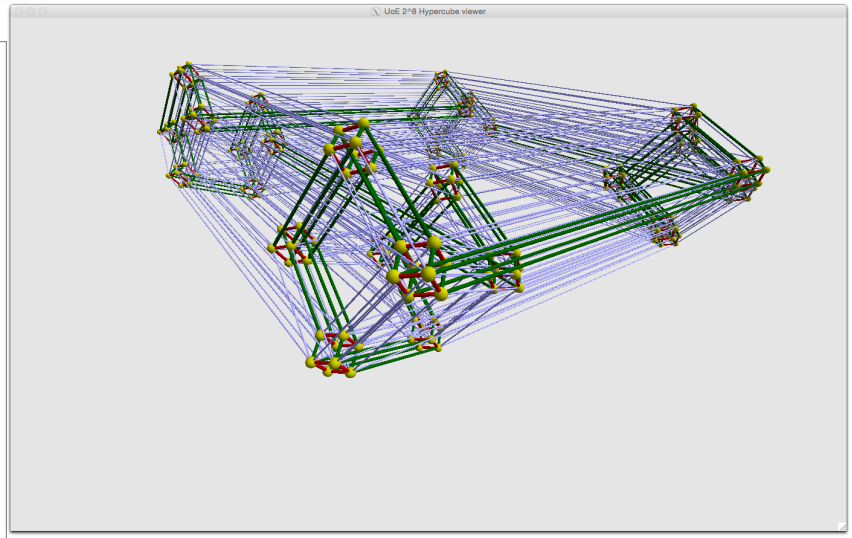
Node	Network	Delivered	Require
KNL	Cray Aries	11	64
KNL	Single EDR	23	64
KNL	Dual EDR	45	64
KNL	Dual Omnipath	50 peak	64

- Summit and Sierra are unlikely to scale beyond one node
- Cori and Theta could really have done with dual rail EDR or Omnipath
- Aurora likely scalable
  - Systems useful for ensemble valence analysis, DD preconditioner in multigrid
- Dual 100GBit/s KNL likely scalable on fine operator

Silicon Graphics ICE-X network







- Can embed 2^n QCD torus inside hypercube so that nearest neigbour comms travels single hop (PAB, SGI)
- Gray counter encode node coordinates; alternative to large torus machines
- Dual rail fat tree would also work, greater switch/cable cost, limit to system size
- Perfect weak scaling obtained; results on 256 nodes
- Results from dual Broadwell cluster, Mellanox EDR (single/dual)
- Drop in performance from out of cache is *expected* to be better on KNL

Algorithms

I have chosen to focus on two aspects that I feel are most fundamental to continued progress

- Multi-scale fermion solvers
- Multi-scale integration

Not covered in detail (but also fundamental):

- Topological sampling
  - Covered by Michael Endres Tuesday @ 9:45
  - Metadynamics, Sanfilippo Tuesday @ 18:10
    - Caution on both: Symptomatic relief is not necessarily a cure
    - Want solutions that address all forms of critical slowing down in an exact MCMC run far enough to converge on the fixed point of the process
- Approaches to free energy, density of states and derived observables, reviewed by Langfeld
  - Applications of Jarzynski's relation in lattice gauge theories; Nada Tuesday@17:10
  - Computing the density of states with the global HMC;Pellegrini Tuesday@17:30
  - Overcoming strong metastabilities with the LLR method;Lucini Tuesday@17:50

Multi-scale fermion solvers

- Index theorem: expect a set of topological modes protected only by quark mass
  - Deflate these modes: big reduction in condition number of Dirac operator
- Cost reduced to $O(V)$: concurrent works
  - arXiv:0706.2298 Luscher
  - arXiv:0707.4018 Brower/Clark/Brannick/Osborn/Rebbi
- Solved problem in valence sector for
  - Wilson
    arXiv:0706.2298 (Luscher)
    arXiv:0707.4018, arXiv:0710.3612, arXiv:0811.4331 (BCBOR)
  - Clover fermions
    arXiv:1011.2775 Osborn, Babich, Brannick, Brower, Clark, Cohen, Rebbi,
    arXiv:1202.2462, arXiv:1303.1377, arXiv:1307.6101 Frommer, Kahl, Kreig, Leder, Rothman
- Gauge evolution: coarsening basis must recomputed after each timestep, reversibility requires higher accuracy
  - arXiv:0710.5417 Luscher
  - arXiv:1307.6101 Frommer, Kahl, Kreig, Leder, Rothman
- Nested solver approaches for overlap
  - arXiv:1410.7170 (Brannick, Frommer, Kahl, Rottman, Strebel
- 5d domain wall approaches using the normal equations
  - arXiv:1205.2933 Cohen
  - arXiv:1402.2585 PAB

Multi-scale fermion solvers

- Capture IR dynamics in a subspace $M\phi_i \approx 0$
- Local coherence $\Rightarrow$ chop into blocks ϕ_i^b
- Schur decompose the matrix into a subspace and the orthogonal complement

$$M = UDL = \begin{bmatrix} M_{\bar{s}\bar{s}} & M_{\bar{s}s} \\ M_{s\bar{s}} & M_{ss} \end{bmatrix} = \begin{bmatrix} 1 & M_{\bar{s}s}M_{ss}^{-1} \\ 0 & 1 \end{bmatrix} \begin{bmatrix} S & 0 \\ 0 & M_{ss} \end{bmatrix} \begin{bmatrix} 1 & 0 \\ M_{ss}^{-1}M_{s\bar{s}} & 1 \end{bmatrix}$$

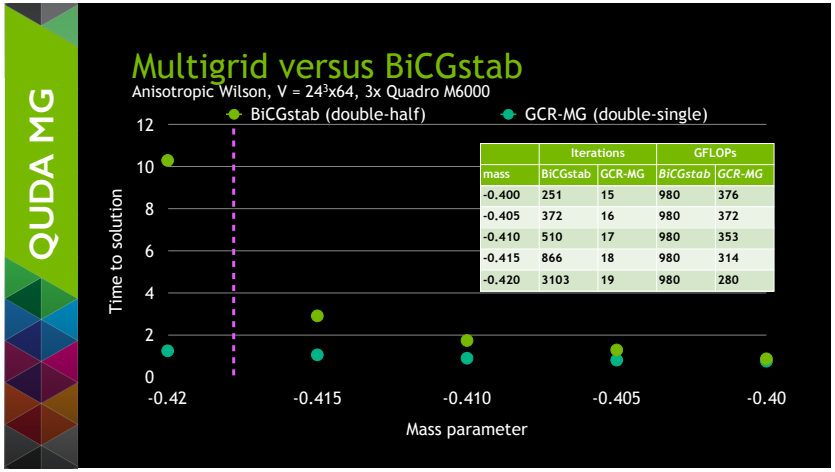
- Represent the matrix M exactly on this IR subspace by computing its matrix elements *little Dirac operator* or coarse grid matrix

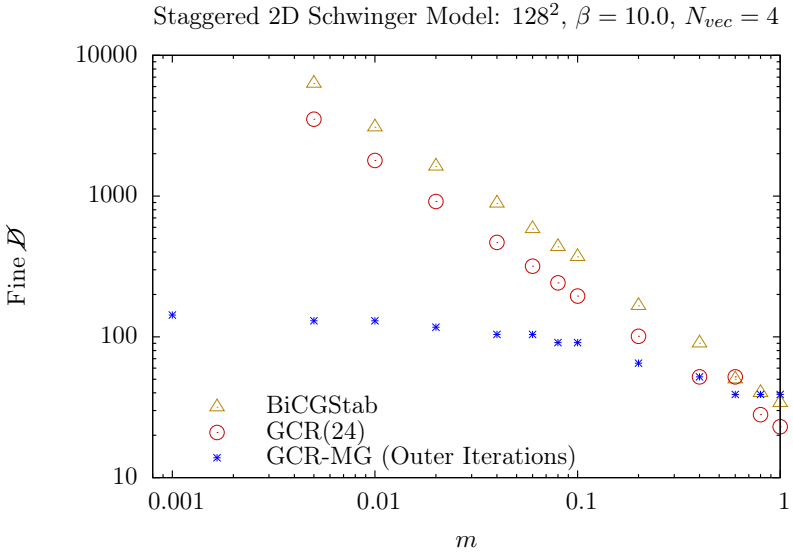
$$A_{jk}^{ab} = \langle \phi_j^a | M | \phi_k^b \rangle \quad ; \quad (M_{SS}) = A_{ij}^{ab} |\phi_i^a\rangle\langle\phi_j^b|. \tag{1}$$

- Inversion via Krylov methods; use in a preconditioner accelerating IR modes
  Smoother (e.g. M_{SAP}) used as preconditioner to address UV modes.
- Double precision outer Krylov solver mops up the rest in a few iterations

Staggered multigrid

Weinberg, Brower, Clark, Strelchenko, Algorithms and Machines, Thursday@15:00.



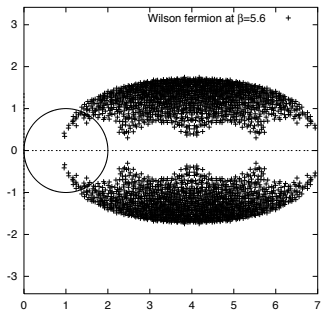


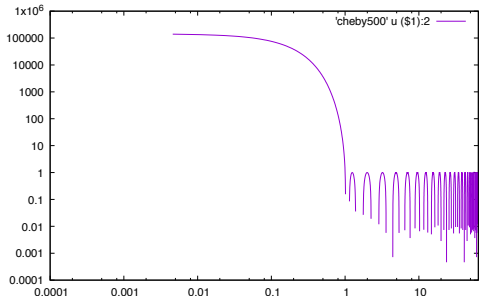
- Coarsen indefinite $\not{D}$ directly
- Project low subspace into definite chirality basis prior to coarsening

5D chiral fermion multigrid

Poster 184: Yamaguchi, PAB

- Coarsen indefinite $\gamma_5 \not{D}$ directly
- Project low subspace into definite chirality basis prior to coarsening [1]
- Use $H = \gamma_5 R_5 D_{dwf}$ Hermitian operator and conjugate residual as basis
- Also works for continued fraction overlap





- Krylov polynomial approximates $P(z) \to \frac{1}{z}$ over region in complex plane *encircling* the pole at zero
- *impossible* to reproduce phase winding over this region with any polynomial

$$\oint z^{-1} dz = 2\pi i \neq \oint P(z) dz = 0$$

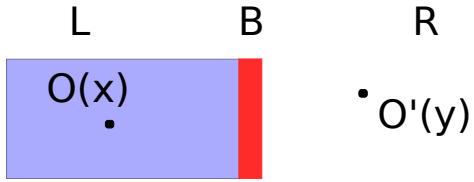
- *Phase response* is the problem: make the system real *indefinite* using γ_5
- These operators are *nearest neighbour* and preserve sparsity in a coarse space.
- Chebyshev filters for subspace generation

[1] trick borrowed from Clark

Multilevel integration for (quenched) fermionic observables

- Domain decomposition and multilevel integration
- Stefan Schaefer, Macro Ce, Algorithms and Machines, Wednesday@09:00,09:20.
- Presently quenched only

Two-Level algorithm



Level-0

N_0 realizations of boundary field B

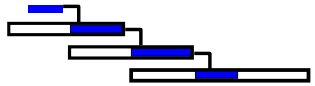
Level-1

For each of the N_0 B fields: N_1 gauge fields in L and R

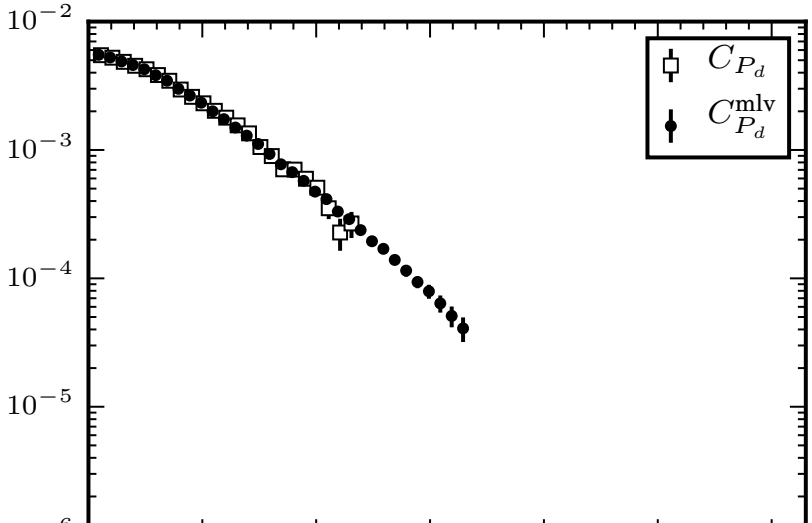
$\rightarrow$ Cost $\propto N_0 \times N_1$

Construction of $N_0 \times N_1^2$ configurations.

Iterate .....



$$D^{-1}(x,y) \approx (-1)^{m-l} \Big[\prod_{i=l}^{m+1} D_{\Omega_i^*}^{-1} D_{\Lambda_{i,i-1}} \Big](x,\cdot)\, D_{\Omega_{m+2}}^{-1}(\cdot,y)$$



Other algorithmic work

Incomplete list:

- Twisted mass multigrid
  Simone Bacchio, Algorithms and Machines, Wednesday@10:00.
- DD-α-AMG solver library:
  Matthais Rottmann Algorithms and Machines, Wednesday@09:40.

  www.github.com/DDalphaAMG
- Implementation of TWQCD's Exact one flavour algorithm for DWF (Murphy Wednesday@10:20)

Multigrid and machines

Machine problems with multigrid

- Amdahl:
  - Coarse space becomes difficult to fine grain parallelise
  - Sublattice site parallelism (Clark)
  - Inexact deflation (Luscher), HDCG: dense matrix deflation with many (eigen)vectors at coarsest levels
- Communications:
  - Is domain decomposition in multigrid smoothers the best option in future?
  - Smoothers should minimise use of network and maximise cache reuse
  - Preserve information selectively on domain boundaries when compute $\gg$ communication
  - HDCG:
    polynomial smoother & reduce precision to 7 mantissa bits in smoother
    same flop count in both cases
    preserves the most significant bits of information flow
    whereas replacing with DD solve (flush to zero) suffers reduced convergence rate

Precision of inner communication	Exponent	Mantissa	Outer iteration count
64 bit	11 bit	52 bit	168
32 bit	8 bit	23 bit	168
16 bit	8 bit	7 bit	168

Summary...

- Tremendous growth in computer power from many core CPU's and GPU's
  - Knights Landing: 0.5-1TF/s single node SP
  - Nvidia Pascal: 1-2 TF/s single node SP
- Interconnects are not keeping pace
  - Fine grid operator requires at least 2:1 ratio of EDR/OPA to compute chips
- Multigrid solver algorithms solved critical slowing down in valence sector for Wilson/Clover
  - Multigrid algorithms appearing for other actions (Staggered, DWF, Twisted Mass)
  - Successful application in HMC exists for Wilson/Clover, but not yet widespread
- Multilevel integration algorithms interesting
- Algorithms that maintain ergodicity are a big challenge to using this power usefully (Endres talk)
- Use of fp64/fp32/fp16 arithmetic in preconditioners or variance reduction is not yet fully explored

And finally...

- The end of Moore scaling has long been anticipated.
- But 3rd space dimension is unused: increase transistor density, reduce wire delays
- Unlikely to give more than several orders of magnitude but very important changes
- Engineering barriers exist but easier than many problems EE has already solved
- We are now seeing first steps in this direction

Suburban sprawl $\longrightarrow$ Metropolis