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

<table>
<thead>
<tr>
<th>System attributes</th>
<th>NERSC Now</th>
<th>OLCF Now</th>
<th>ALCF Now</th>
<th>NERSC Upgrade</th>
<th>OLCF Upgrade</th>
<th>ALCF Upgrades</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name</td>
<td>Edison</td>
<td>TITAN</td>
<td>MIRA</td>
<td>Cori 2016</td>
<td>Summit 2017-2018</td>
<td>Theta 2016</td>
</tr>
<tr>
<td>Planned Installation</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Aurora 2018-2019</td>
</tr>
<tr>
<td>System peak (PF)</td>
<td>2.6</td>
<td>27</td>
<td>10</td>
<td>&gt; 30</td>
<td>150</td>
<td>&gt;8.5</td>
</tr>
<tr>
<td>Peak Power (MW)</td>
<td>2</td>
<td>9</td>
<td>4.8</td>
<td>&lt; 3.7</td>
<td>10</td>
<td>1.7</td>
</tr>
<tr>
<td>Total system memory</td>
<td>357 TB</td>
<td>710TB</td>
<td>768TB</td>
<td>~1 PB DDR4 + High Bandwidth Memory (HBM) + 1.5PB persistent memory</td>
<td>&gt; 1.74 PB DDR4 + HBM + 2.8 PB persistent memory</td>
<td>&gt;480 TB DDR4 + High Bandwidth Memory (HBM)</td>
</tr>
<tr>
<td>Node performance (TF)</td>
<td>0.460</td>
<td>1.452</td>
<td>0.204</td>
<td>&gt; 3</td>
<td>&gt; 40</td>
<td>&gt; 3</td>
</tr>
<tr>
<td>Node processors</td>
<td>Intel Ivy Bridge</td>
<td>AMD Opteron</td>
<td>Nvidia Kepler</td>
<td>64-bit PowerPC A2</td>
<td>Intel Knights Landing many core CPUs &amp; Intel Haswell CPU in data partition</td>
<td>Multiple IBM Power9 CPUs &amp; multiple Nvidia Volta GPUs</td>
</tr>
<tr>
<td>System size (nodes)</td>
<td>5,600 nodes</td>
<td>18,688 nodes</td>
<td>49,152</td>
<td>9,300 nodes &amp; 1,900 nodes in data partition</td>
<td>~3,500 nodes</td>
<td>&gt;2,500 nodes</td>
</tr>
<tr>
<td>System Interconnect</td>
<td>Aries</td>
<td>Gemini</td>
<td>5D Torus</td>
<td>Aries</td>
<td>Dual Rail EDR-IB</td>
<td>Aries</td>
</tr>
<tr>
<td>File System</td>
<td>7.6 PB 168 GB/s, Lustre®</td>
<td>32 PB 1 TB/s, Lustre®</td>
<td>26 PB 300 GB/s GPFS™</td>
<td>28 PB 744 GB/s Lustre®</td>
<td>120 PB 1 TB/s GPFS™</td>
<td>10PB, 210 GB/s Lustre initial</td>
</tr>
</tbody>
</table>
Immediate roadmap

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

<table>
<thead>
<tr>
<th>Name</th>
<th>Planned Installation</th>
<th>NERSC Upgrade</th>
<th>OLCF Upgrade</th>
<th>ALCF Upgrade</th>
</tr>
</thead>
<tbody>
<tr>
<td>Edison</td>
<td>TITAN</td>
<td>NERSC Upgrade</td>
<td>OLCF Upgrade</td>
<td>ALCF Upgrade</td>
</tr>
<tr>
<td>TITAN</td>
<td>NERSC Upgrade</td>
<td>OLCF Upgrade</td>
<td>ALCF Upgrade</td>
<td></td>
</tr>
<tr>
<td>Summit 2017-2018</td>
<td>Theta 2016</td>
<td>ALCF Upgrade</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Theta 2016</td>
<td>ALCF Upgrade</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Aurora 2018-2019</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

- System peak (PF)
- Total system memory
- Node performance (TP)
- Node processors
- System size (nodes)
- System Interconnect
- File System
Growing on chip parallelism...

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

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

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

Standard industry solution is to dump it on the programmer!
Wireloads and geometry

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**

\[
C = \frac{Q}{V} = 2\pi L\varepsilon / \log(r_0 / r)
\]

- **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 \( \rho \) and \( \varepsilon \)

**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 year's 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/crystalline glass cell) should increase memory density. Micron/Intel 3D Xpoint branding: 4x higher density than DRAM; SSD’s → NVDIMM’s 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 transferred!

You don’t buy a multipack if you only want one item!
CPU SIMD model

SIMD brings a new level of restrictiveness 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!
• 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

<table>
<thead>
<tr>
<th>Chip</th>
<th>Clock</th>
<th>blocks</th>
<th>per block</th>
<th>SP madd</th>
<th>issue</th>
<th>SP madd</th>
<th>peak</th>
</tr>
</thead>
<tbody>
<tr>
<td>GP100</td>
<td>1.4 GHz</td>
<td>56 SM’s</td>
<td>2 IB/RF</td>
<td>32</td>
<td>112 × 2</td>
<td>3584</td>
<td>10.5 TF/s</td>
</tr>
<tr>
<td>KNL</td>
<td>1.4 GHz</td>
<td>36 L2 tiles</td>
<td>2 cores</td>
<td>32</td>
<td>72 × 2</td>
<td>2304</td>
<td>6.4 TF/s</td>
</tr>
<tr>
<td>Broadwell</td>
<td>2.5</td>
<td></td>
<td>18 cores</td>
<td>16</td>
<td>18 × 2</td>
<td>576</td>
<td>1.4 TF/s</td>
</tr>
<tr>
<td>Skylake</td>
<td>?</td>
<td></td>
<td>28 cores</td>
<td>32</td>
<td>28 × 2</td>
<td>1792</td>
<td>4.4 TF/s (EST)</td>
</tr>
</tbody>
</table>
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. Binary compatible with Intel Xeon processors using Haswell Instruction Set (except TSX).

Bandwidth 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.
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
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

No TSX. Under separate CPUID bit

KNL (Xeon Phi²)

LEGACY

KNC (Xeon Phi²)

• SSE*
• AVX
• AVX2
• x87/MMX
• BMI
• TSX

• AVX-512F
• AVX-512CD
• AVX-512PF
• AVX-512ER

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

<table>
<thead>
<tr>
<th>Tesla Products</th>
<th>Tesla K40</th>
<th>Tesla M40</th>
<th>Tesla P100</th>
</tr>
</thead>
<tbody>
<tr>
<td>GPU</td>
<td>GK110 (Kepler)</td>
<td>GM200 (Maxwell)</td>
<td>GP100 (Pascal)</td>
</tr>
<tr>
<td>SMs</td>
<td>15</td>
<td>24</td>
<td>56</td>
</tr>
<tr>
<td>TPCs</td>
<td>15</td>
<td>24</td>
<td>28</td>
</tr>
<tr>
<td>FP32 CUDA Cores / SM</td>
<td>192</td>
<td>128</td>
<td>64</td>
</tr>
<tr>
<td>FP32 CUDA Cores / GPU</td>
<td>2880</td>
<td>3072</td>
<td>3584</td>
</tr>
<tr>
<td>FP64 CUDA Cores / SM</td>
<td>64</td>
<td>4</td>
<td>32</td>
</tr>
<tr>
<td>FP64 CUDA Cores / GPU</td>
<td>960</td>
<td>96</td>
<td>1792</td>
</tr>
<tr>
<td>Base Clock</td>
<td>745 MHz</td>
<td>948 MHz</td>
<td>1328 MHz</td>
</tr>
<tr>
<td>GPU Boost Clock</td>
<td>810/875 MHz</td>
<td>1114 MHz</td>
<td>1480 MHz</td>
</tr>
<tr>
<td>Peak FP32 GFLOPs¹</td>
<td>5040</td>
<td>6840</td>
<td>10600</td>
</tr>
<tr>
<td>Peak FP64 GFLOPs¹</td>
<td>1680</td>
<td>210</td>
<td>5300</td>
</tr>
<tr>
<td>Texture Units</td>
<td>240</td>
<td>192</td>
<td>224</td>
</tr>
<tr>
<td>Memory Interface</td>
<td>384-bit GDDR5</td>
<td>384-bit GDDR5</td>
<td>4096-bit HBM2</td>
</tr>
<tr>
<td>Memory Size</td>
<td>Up to 12 GB</td>
<td>Up to 24 GB</td>
<td>16 GB</td>
</tr>
<tr>
<td>L2 Cache Size</td>
<td>1536 KB</td>
<td>3072 KB</td>
<td>4096 KB</td>
</tr>
<tr>
<td>Register File Size / SM</td>
<td>256 KB</td>
<td>256 KB</td>
<td>256 KB</td>
</tr>
<tr>
<td>Register File Size / GPU</td>
<td>3840 KB</td>
<td>6144 KB</td>
<td>14336 KB</td>
</tr>
<tr>
<td>TDP</td>
<td>235 Watts</td>
<td>250 Watts</td>
<td>300 Watts</td>
</tr>
<tr>
<td>Transistors</td>
<td>7.1 billion</td>
<td>8 billion</td>
<td>15.3 billion</td>
</tr>
<tr>
<td>GPU Die Size</td>
<td>551 mm²</td>
<td>601 mm²</td>
<td>610 mm²</td>
</tr>
<tr>
<td>Manufacturing Process</td>
<td>28-nm</td>
<td>28-nm</td>
<td>16-nm FinFET</td>
</tr>
</tbody>
</table>

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*

```c
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 $L_s = N_{\text{rhs}}$. Suppresses gauge field overhead;
  - Cache reuse $\times N_{\text{stencil}}$ on Fermion possible

- Accesses per 4d site of result
  - Fermion: $N_{\text{stencil}} \times (N_s \in \{1, 4\}) \times (N_c = 3) \times (N_{\text{rhs}} \in \{1, 16\})$ complex
  - Gauge: $2N_d \times N_c^2$ complex

- Flops
  - $N_{\text{stencil}} \times N_{\text{hs}}$ $SU(3)$ MatVec: $66 \times N_{\text{hs}} \times N_{\text{stencil}}$ (+ spin projection)

<table>
<thead>
<tr>
<th>Action</th>
<th>Fermion Vol</th>
<th>Surface</th>
<th>$N_s$</th>
<th>$N_{\text{hs}}$</th>
<th>$N_{\text{rhs}}$</th>
<th>Flops</th>
<th>Bytes</th>
<th>Bytes/Flops</th>
</tr>
</thead>
<tbody>
<tr>
<td>HISQ</td>
<td>$L^4$</td>
<td>$3 \times 8 \times L^3$</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1146</td>
<td>1560</td>
<td>1.36</td>
</tr>
<tr>
<td>Wilson</td>
<td>$L^4$</td>
<td>$8 \times L^3$</td>
<td>4</td>
<td>2</td>
<td>1</td>
<td>1320</td>
<td>1440</td>
<td>1.09</td>
</tr>
<tr>
<td>DWF</td>
<td>$L^4 \times N$</td>
<td>$8 \times L^3$</td>
<td>4</td>
<td>2</td>
<td>16</td>
<td>$N_{\text{rhs}} \times 1320$</td>
<td>$N_{\text{rhs}} \times 864$</td>
<td>0.65</td>
</tr>
<tr>
<td>Wilson-RHS</td>
<td>$L^4$</td>
<td>$8 \times L^3$</td>
<td>4</td>
<td>2</td>
<td>16</td>
<td>$N_{\text{rhs}} \times 1320$</td>
<td>$N_{\text{rhs}} \times 864$</td>
<td>0.65</td>
</tr>
<tr>
<td>HISQ-RHS</td>
<td>$L^4$</td>
<td>$3 \times 8 \times L^3$</td>
<td>1</td>
<td>1</td>
<td>16</td>
<td>$N_{\text{rhs}} \times 1146$</td>
<td>$N_{\text{rhs}} \times 408$</td>
<td>0.36</td>
</tr>
</tbody>
</table>

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

Scaling fine operator requires interconnect bandwidth

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

where $R$ is the reuse factor obtained for the stencil in caches
Intel Knight’s Landing Performance Results
Vectorisation strategy

Vector = Matrix x Vector

Many vectors = many matrices x many vectors

Reduction of vector sum is bottleneck for small N

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...)

- 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

---

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.
GRID data parallel template library

<table>
<thead>
<tr>
<th>Ordering</th>
<th>Layout</th>
<th>Vectorisation</th>
<th>Data Reuse</th>
</tr>
</thead>
<tbody>
<tr>
<td>Microprocessor</td>
<td>Array-of-Structs (AoS)</td>
<td>Hard</td>
<td>Maximised</td>
</tr>
<tr>
<td>Vector</td>
<td>Struct-of-Array (SoA)</td>
<td>Easy</td>
<td>Minimised</td>
</tr>
<tr>
<td>Bagel</td>
<td>Array-of-structs-of-short-vectors (AoSoSV)</td>
<td>Easy</td>
<td>Maximised</td>
</tr>
</tbody>
</table>

- [www.github.com/paboyle/Grid](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

```cpp
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**

<table>
<thead>
<tr>
<th>Architecture</th>
<th>Cores</th>
<th>GF/s (Ls x Dw)</th>
<th>peak</th>
</tr>
</thead>
<tbody>
<tr>
<td>Intel Knight’s Landing 7250</td>
<td>68</td>
<td>770</td>
<td>6100</td>
</tr>
<tr>
<td>Intel Knight’s Corner</td>
<td>60</td>
<td>270</td>
<td>2400</td>
</tr>
<tr>
<td>Intel Broadwellx2</td>
<td>36</td>
<td>800</td>
<td>2700</td>
</tr>
<tr>
<td>Intel Haswellx2</td>
<td>32</td>
<td>640</td>
<td>2400</td>
</tr>
<tr>
<td>Intel Ivybridgex2</td>
<td>24</td>
<td>270</td>
<td>920</td>
</tr>
<tr>
<td>AMD Interlagosx4</td>
<td>32 (16)</td>
<td>80</td>
<td>628</td>
</tr>
</tbody>
</table>

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

*Figure 5:* We compare the SU(3) x SU(3) performance (Gflop/s) versus footprint (bytes) under AVX-1 instructions of a slightly slower clocked quad-core Haswell (Crystalwell) to a quad-core Ivybridge. The effect of 128MB integrated on-package eDRAM cache is clearly visible.

**Knight’s Landing memory system profile**

*SU3 x SU3 example*  
(GB/s vs footprint bytes/core)
Grid multi-RHS Wilson Dslash and DWF

- 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?

- 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
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 maintenance 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

Weak Scaling for DWF BAGEL CG inverter

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

\[ B_{\text{network}} \sim \frac{B_{\text{memory}}}{L} \times R \]

- Determine reuse factor via \( B_{\text{network}} = P_{\text{dwf}} \times 0.65/L \)
- This is the \textit{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

<table>
<thead>
<tr>
<th>Nodes</th>
<th>Memory (GB/s)</th>
<th>Bidi network requirement (GB/s)</th>
<th>Delivered</th>
<th>Require</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>(L=10)</td>
<td>(L=16)</td>
<td>(L=32)</td>
</tr>
<tr>
<td>2xBroadwell</td>
<td>100</td>
<td>100</td>
<td>16</td>
<td>8</td>
</tr>
<tr>
<td>KNL</td>
<td>400</td>
<td>100</td>
<td>64</td>
<td>32</td>
</tr>
<tr>
<td>P100</td>
<td>700</td>
<td>200</td>
<td>128</td>
<td>64</td>
</tr>
<tr>
<td>DGX-1</td>
<td>5600</td>
<td>-</td>
<td>975</td>
<td>487</td>
</tr>
</tbody>
</table>

- 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 neighbour 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
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
- Solved problem in valence sector for
  - Wilson
    arXiv:0706.2298 (Luscher)
  - Clover fermions
    arXiv:1011.2775 Osborn, Babich, Brannick, Brower, Clark, Cohen, Rebbi,
- Gauge evolution: coarsening basis must recomputed after each timestep, reversibility requires higher accuracy
  - arXiv:0710.5417 Luscher
- 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 $\phi_i^b$
- Schur decompose the matrix into a subspace and the orthogonal complement

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

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

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

- 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 $\mathcal{D}$ directly
- Project low subspace into definite chirality basis prior to coarsening
5D chiral fermion multigrid

Poster 184: Yamaguchi, PAB

- Coarsen indefinite $\gamma_5 D$ 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

\[ H = \gamma_5 R_5 D_{dwf} \]

\(^1\)trick borrowed from Clark

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

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

- \textit{Phase response} is the problem: make the system real indefinite using $\gamma_5$
- These operators are nearest neighbour and preserve sparsity in a coarse space.
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 $R$

$\rightarrow \text{Cost } \propto N_0 \times N_1$

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

$D^{-1}(x,y) \approx (-1)^{m-l} \prod_{i=l}^{m+1} D^{-1}_{\Omega_i} D_{N_i,i-1}(x, \cdot) D^{-1}_{\Omega_{m+2}}(\cdot, y)$
Other algorithmic work

Incomplete list:

- **Twisted mass multigrid**
  Simone Bacchio, Algorithms and Machines, Wednesday@10:00.

- **DD-\(\alpha\)-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 ≫ 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

<table>
<thead>
<tr>
<th>Precision of inner communication</th>
<th>Exponent</th>
<th>Mantissa</th>
<th>Outer iteration count</th>
</tr>
</thead>
<tbody>
<tr>
<td>64 bit</td>
<td>11 bit</td>
<td>52 bit</td>
<td>168</td>
</tr>
<tr>
<td>32 bit</td>
<td>8 bit</td>
<td>23 bit</td>
<td>168</td>
</tr>
<tr>
<td>16 bit</td>
<td>8 bit</td>
<td>7 bit</td>
<td>168</td>
</tr>
</tbody>
</table>
• 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 $\rightarrow$ Metropolis