

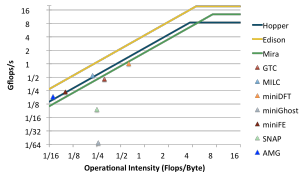
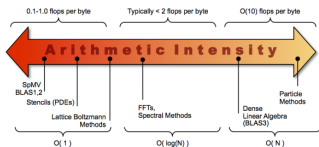
# Exascale HPC Technologies Performance, Portability, and Scalability

Peter Boyle

University of Edinburgh (Theoretical Particle Physics)  
Alan Turing Institute (Intel ATI codesign project for Machine Learning)

- Exascale processor landscape
- Performance portability
- Interconnects
- Some QCD examples
  - What can go right
  - What can go wrong
  - Importance of deterministic good performance

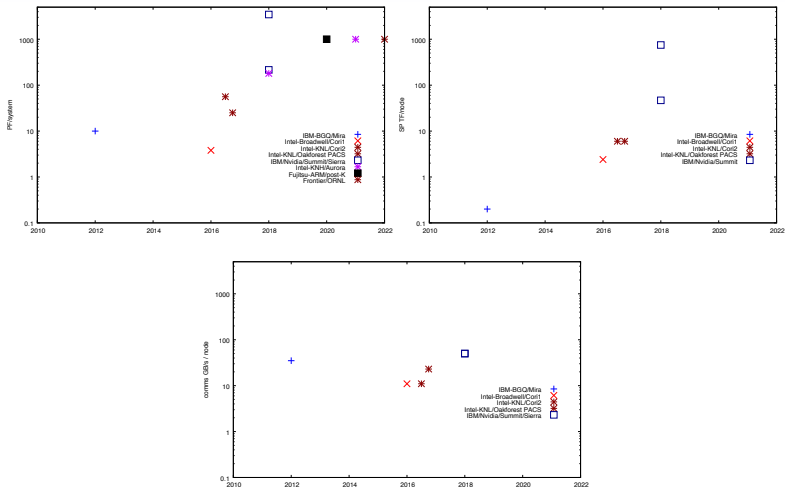
- Floating point is now free
- Data motion is now key
- Feeding the beast is the problem <sup>1</sup>



- Berkely roofline model:  $\text{Flops/Second} = (\text{Flops/Byte}) \times (\text{Bytes/Second})$ 
  - One dimensional - *only* memory bandwidth is considered
  - **Arithmetic intensity** = (Flops/ Byte)
- With more care can categorise data references by origin
  - Cache, Memory, Network

<sup>1</sup>possibly except for bitcoin mining

# Immediate roadmap



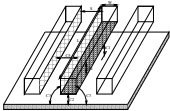
- 400x+ increase in SP node performance accompanied by NO increase in interconnect
  - FP16 gain is 6x more again!
- US exascale systems planned in 2021, 2022 (Aurora/Argonne, Frontier/ORNL)
- Not a case of business as usual for algorithms!

## Growing on chip parallelism...

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 multi-core parallelism, growth in SIMD parallelism
- Growth in complexity of memory hierarchy
- Industry standard response is: “dump it on the programmer”

## Wireloads and geometry



Gate Length (nm)	Dielectric Constant $\kappa$	Metal $\rho$ ( $\mu\Omega\text{-cm}$ )	Width (nm)	Aspect Ratio	$R_{\text{wire}}$ ( $\text{m}\Omega/\mu\text{m}$ )	$C_{\text{wire}}$ (fF/ $\mu\text{m}$ )
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

Physics creates computer architecture: model wire as rod of metal  $L \times \pi r^2$

- **Charge:** Gauss's law

$$2\pi rLE = \frac{Q}{\epsilon} \Rightarrow E = \frac{Q}{\epsilon 2\pi rl}$$

- Capacitance

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

- Resistance

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

- 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  $\epsilon$

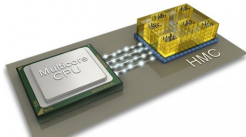
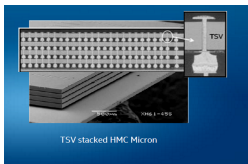
*Multi-core design with long-haul buses only possible strategy for 8 Billion transistors*

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

# Exciting technology directions

## 3D memory integration

- Apply to memory buses with through-silicon-via's (TSVs)!
- 10x increase in memory throughput at fixed power
  - **Exploited** in Nvidia GPU's, AMD GPU's, Intel Knight's Landing 😊
  - **Not yet exploited** in Intel or AMD server parts ☹
    - Direct mapped cache or distinct NUMA domain?



## Network integration

- KNL-F has on package 2 x 100 Gbit/s Omnipath interfaces (marginal cost \$300 per node)
- Skylake/Purley will also integrate Omnipath
- Nvidia can interconnect up to 8 GPU's with NVLINK

*How to balance the engineering effort between subsystems is key*

# Processor technologies

- **HBM / MCDRAM (500 - 1000 GB/s)**
  - Intel Knights Landing (KNL) 16 GB (AXPY 450 GB/s)
  - Nvidia Pascal P100 16-32 GB (AXPY 600 GB/s)
  - Nvidia Volta V100 16-32 GB (AXPY 840 GB/s)
- **GDDR (500 GB/s)**
  - Nvidia GTX1080ti
- **DDR (100-150 GB/s)**
  - Intel Xeon, IBM Power9, AMD Zen, Cavium/Fujitsu ARM
  - ... when will these adopt 3D memory & how to organise?
  - Cache vs. NUMA domain

Chip	Clock	SM/cores	SP madd	issue	SP madd	peak	TDP		Mem BW
<b>GPU</b>									
P100	1.48 GHz	56 SM's	32	112 × 2	3584	10.5 TF/s	300W	\$ 9000	700 GB/s
GTX1080ti	1.48 GHz	56 SM's	32	112 × 2	3584	10.5 TF/s	300W	\$ 800	500 GB/s
V100	1.53 GHz	80 SM's	32	160 × 2	5120	15.7 TF/s	300W	\$ 10000	840 GB/s
<b>Many-core</b>									
KNL	1.4 GHz	36 L2 × 2 cores	32	72 × 2	2304	6.4 TF/s	215W	\$ 2000	450 GB/s
<b>Multi-core</b>									
Broadwell	2.5	18 cores	16	18 × 2	576	1.4 TF/s	165W	\$4000	55 GB/s
Skylake	3.0	28 cores	32	28 × 2	1792	5.3 TF/s	205W	\$8000	95 GB/s
Skylake	3.0	12 cores	16	12 × 2	384	1.15 TF/s	85W	\$1000	80 GB/s

# IBM's Summit

- ORNL, 4608 nodes, will likely take top500 lead next week
  - 200 PF/s (double) at 10MW, 3 EF/s half precision for AI !
  - approx \$200M
- Each node:
  - 6 V100 GPU's, 90TF/s single precision, 750TF/s half precision for AI
  - 5000+ GB/s memory bandwidth
    - Strong interior, NVlink 75+75 GB/s per link interconnect (450 GB/s per GPU) 😊
- Dual rail 50GB/s EDR exterior interconnect
  - 100:1 memory to network ratio 😞
- Explicitly programme GPU-GPU and MPI (EDR) transfers for performance

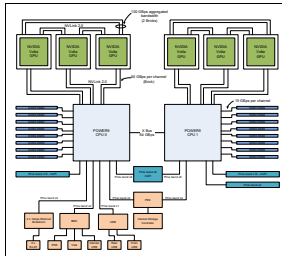
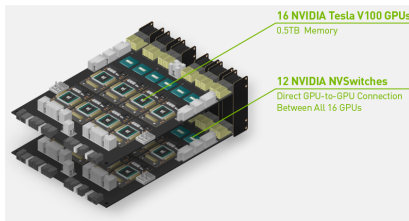


Figure 2-6 The Power AC922 server model 8335-GTW logical system diagram



## HGX-2

- NVlink has been given an accompanying switch NVswitch enabling 16 GPU HGX-2 systems
- Very strong, and very expensive base node; sticking a network card on-it is not a solution for large systems
- Would desperately like to see this in an extensible mesh



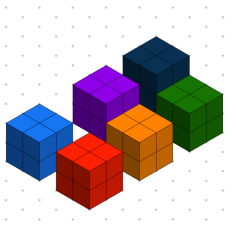
## Multiple chips/GPUs per node

### Multi-socket servers: NUMA aware code

- hybrid OpenMP + MPI
- Use 1:1 mapping between MPI ranks and sockets
- Unix shared memory between sockets (over UPI)
- Reserve MPI transfers for inter-node

### Multi-GPU servers: NVlink aware code

- Use 1:1 mapping between MPI ranks and GPU's
- Use Cuda to open up direct GPU-GPU device memory access over NVlink
- Reserve MPI transfers for inter-node, direct to GPU if possible



## Performance portable programming

# Caches and locality

Memory systems are *granular*

- Big gain from *spatial locality* of reference: use everything that gets transferred!



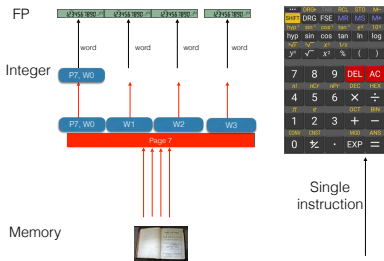
You don't buy a multipack if you only want one beer!

# 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

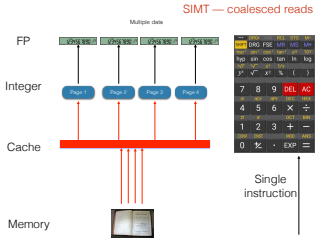
SIMD CPU



contiguous block memory accesses with same operations performed on adjacent words

- Both granular memory transfers *and* SIMD execution need to 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 since these dictate memory layout!

## 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

- **For performance** must arrange to have same operation applied to consecutive words
  - *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

contiguous block memory accesses with same operations performed on adjacent words

## Consistent emerging solution : advanced C++

Granularity exposed through ISA/Performance

⇒ data structures must change with each architecture

OpenMP, OpenAcc do not address data layout

Several packages arriving at similar conclusions:

- Kokkos (Sandia)
- RAJA (Livermore)
- Grid (Edinburgh)

Use fairly advanced C++11 features, inline header template library

- Traditional OOP performance negative (virtual functions, passing objects, STL)
- Discipline and coding standards are required.

## Consistent emerging solution : advanced C++

1. **Abstract arrays** & accessors through **C++ container** templates
2. Data **layout changes with architecture**
  - Trigger a partial vector transformation to suit architecture via template parameters
3. **Capture** potentially **offloaded code** in *device lambda function*

```
parallel_for(i,0,N, [=] accelerator (int i) mutable { {  
    // Lambda captures a,b,c,d, and constructs device function object  
    a[i] = b[i] + c[i]*sin(d[i]);  
}});
```

- Raja: exposes *policy* controls for where loop executes  
<https://github.com/LLNL/RAJA>
- Kokkos: exposes *policy* controls for where loop executes, and views of layout  
<https://github.com/kokkos>
- Grid: hide in higher level data parallel interface for structured grids  
inspired by QDP-JIT (JLAB), but without the JIT

# GRID data parallel template library

- [www.github.com/paboyle/Grid](https://www.github.com/paboyle/Grid), arXiv:1512.03487<sup>2</sup>

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

- 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.*

<sup>2</sup>Also: good, flexible C++ object serialisation using variadic macros. IDL's not required.

## Grid single node 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 Skylakex2	48	1200	9200
Intel Broadwellx2	36	800	2700
Intel Haswellx2	32	640	2400
Intel Ivybridgex2	24	270	920
AMD EPYCx2	64	590	3276
AMD Interlagosx4	32 (16)	80	628

- Dropped to inline assembly for key kernel in KNL and BlueGene/Q
- EPYC is MCM; ran 4 MPI ranks per socket, one per die

Common source *GPU port* is functional but under tuning.

- Simple data parallel code saturates memory bandwidth
- Use Unified Virtual Memory (i.e. automatic host-device transfers)

## Interconnects<sup>3</sup>

---

<sup>3</sup>NB: programmed with MPI (message passing interface)

# Interconnect technologies

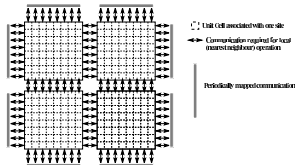
- Integration of network on package useful: SKL-F, KNL-F
  - Recall IBM BlueGene integrated torus router on compute chip
- Silicon photonics
  - 100Gbit/s copper cables cost under 100 USD
  - 100Gbit/s active optical cables (4 bits) cost around 1000 USD
    - Hopefully silicon photonics can lower cost of optics



Never before in the field of computing, has so much been paid, by so many, for so few bits!

# QCD sparse matrix PDE solver communications

- $L^4$  local volume (space + time)
- finite difference operator 8 point **stencil**



Action	Fermion Vol	Surface	$L_5$	Flops	Bytes	Bytes/Flops
DWF	$L^4 \times N$	$8 \times L^3$	16	$L_5 \times 1320$	$L_5 \times 864$	0.65

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

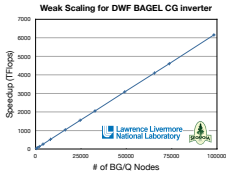
Scaling QCD sparse matrix requires interconnect bandwidth for halo exchange

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

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

- Aim: Distribute  $100^4$  datapoints over  $10^4$  nodes

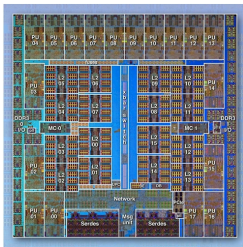
## QCD on DiRAC BlueGene/Q (2012-2018)



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

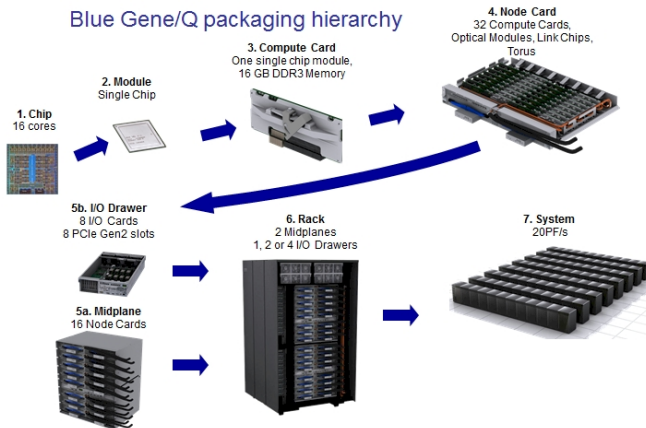


**Sustained 7.2 Pflop/s on 1.6 Million cores (Gordon Bell finalist SC 2013)**  
**Edinburgh system 98,304 cores (installed 2012)**



# QCD on DiRAC BlueGene/Q (2012-2018)

## Blue Gene/Q packaging hierarchy



- Integrated router and large, midplane racks suppress optics cost by surface-to-volume
- No “extra” HFI components or switches: marginal cost of pennies per node.
- Arguably the whole point of VLSI...

# Interconnect Requirements

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

- Project network requirement for balanced communication and computation

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	
V100	840	325	203	100	
Summit					
6xV100	5040	-	1950	1200	600

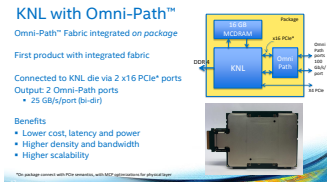
Node	Network	Delivered GB/s	Require
KNL	Cray Aries	11	64
KNL	Single EDR	23	64
KNL	Dual EDR	45	64
KNL	Dual Omnipath	45	64
Summit	Dual EDR	45	1200

- Cori and Theta (Cray Aries) could really have done with dual rail EDR or Omnipath
- Dual 100Gbit/s KNL has proved scalable on fine operator (e.g. Brookhaven cluster)

# Networks and locality

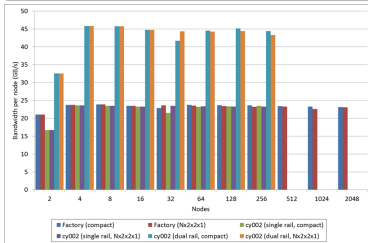
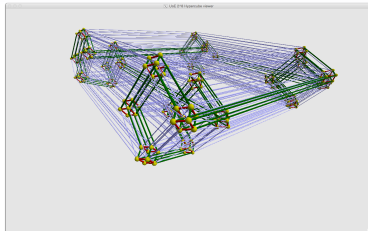
## Can we reduce the cost of networks?

- We accept locality optimisation in almost every level of a computing system
- Providing full bisection bandwidth through a fat-tree switch is expensive
- Torus networks have done well in the past; need smart application mapping in large systems
  - e.g. cartesian communicators
- KNL-F and SKL-F integrate two HFI's on package; very few dual rail systems due to switch & cable costs



- BlueGene/Q : integrated routing network on compute chip ; the point of VLSI  
As near to glueless assembly as possible with marginal additional cost

# HPE ICE-XA hypercube network



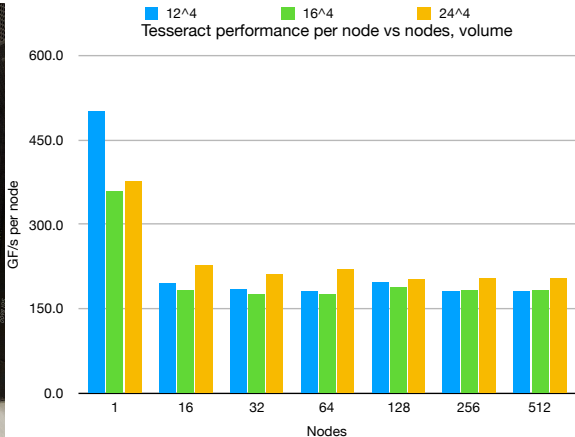
## Improvement over Default Process Placement

Nodes	Decomp	Bandwidth
2	2x1x1x1	0.98
4	2x2x1x1	1.00
8	2x2x2x1	1.00
16	4x2x2x1	1.27
32	4x4x2x1	1.70
64	4x4x4x1	2.00
128	8x4x4x1	2.09
256	8x8x4x1	2.60
512	8x8x8x1	3.30
1024	16x8x8x1	3.51
2048	16x16x8x1	3.84

- Small project with SGI/HPE on Mellanox EDR networks
- Embed  $2^n$  QCD torus inside hypercube so that nearest neighbour comms travels single hop  
4x speed up over default MPI Cartesian communicators on large systems  
⇒ Customise HPE 8600 (SGI ICE-XA) to use  $16 = 2^4$  nodes per leaf switch

# DiRAC HPE ICE-XA hypercube network

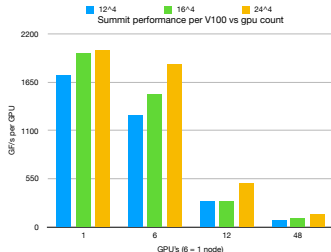
- Edinburgh HPE 8600 system (Installed April 2018)
  - Low end Skylake Silver4116, 12 core parts
  - Single rail Omnipath interconnect
  - Relatively cheap node: high node count and scalability



## Same tightly coupled problem on Summit

Use Nvidia QUDA code: **This is a bad (apples to oranges) comparison at present for three reasons**

1. Tesseract node is 1/2 price of a Volta GPU, currently 2x performance
  - But, Summit has 2x better price/performance for communication light code
2. Summit does not yet have Gpu Direct RDMA (GDR) enabling MPI from device memory
  - Anticipate 4x gain when GDR is enabled on Summit
  - If this bears up, break even on price/performance for this interconnect heavy code
3. Many problems, even in QCD, are not so communication heavy (e.g. multigrid Wilson)



# All this worked out the box right?

- Collaboration with Intel: concurrency updates to Intel MPI and Omnipath software stack
- Reentrancy to MPI needed with hybrid threads + MPI when many HFI's
- Avoid 4KB pages due to per page software overhead

<https://arxiv.org/pdf/1711.04883.pdf>

## Accelerating HPC codes on Intel® Omni-Path Architecture networks: From particle physics to Machine Learning

Peter Boyle,<sup>1</sup> Michael Charlevet,<sup>2</sup> Guido Cossu,<sup>3</sup> Christopher Kelly,<sup>4</sup> Christoph Lehner,<sup>2</sup> and Lawrence Meadows<sup>2</sup>

<sup>1</sup>The University of Edinburgh and Alan Turing Institute

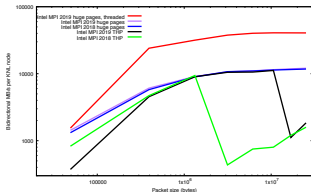
<sup>2</sup>Intel

<sup>3</sup>The University of Edinburgh

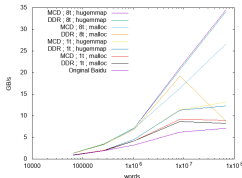
<sup>4</sup>Columbia University

<sup>5</sup>Brookhaven National Laboratory

- Edinburgh-Brookhaven-Columbia-Intel paper
- Brookhaven dual rail KNL/OPA system



- Baidu "optimised reduction" code available open source online
- <http://research.baidu.com/bringing-hpc-techniques-deep-learning/>
- <https://github.com/baidu-research/baidu-allreduce>



- DiRAC procurement benchmarks also required software updates for Mellanox HPCX 2.0

<https://www.nextplatform.com/2017/11/29/the-battle-of-the-infinibands/>

## Importance of deterministic performance

- Linux VM will fragment over time under use; probability of transparent huge pages decreases
- In a 1000 node system, random slow down translates to “convoy” mode for the whole system
- Similar effects from dynamic frequency scaling events
- Can manifest itself unexplained poor scalability

Year	Arch/ CPU	Reg file	Memory	Page Page/RF	Time to read page / Page count
1985	intel32 80386	32B	640KB	4KB 127	64 us 160
1993	+x87 80486DX	32B + 80B	4MB	4KB 36	16 us 1024
2003	intel64 Pentium4	128B + 256B	512MB	4KB 10	128ns 128,000
2011	AVX Sandybridge	128B + 512B	4GB	4KB 6	32ns 1,000,000
2017	AVX512 Skylake	128B + 2048B	64GB	4KB 1.9	16ns 16,000,000

## Summary...

- 3D and 2.5D in package memory alleviates bandwidth constraints
- Massive floating point throughput for weakly coupled problems, or problems of limited size
  - Intel Knights Landing: 0.5-1TF/s single node SP for QCD
  - Nvidia Pascal: 1-2 TF/s single node SP for QCD
  - Nvidia Volta: 2-3 TF/s single node SP for QCD
  - Nvidia Volta: 100+ TF/s half precision AI
- Interconnects are not keeping pace
- HPC codes and algorithms must adapt or die

## Some controversial commentary

- The promise of GP-GPU has to some degree fallen victim to a market segmentation strategy
  - first person shooters pay \$ 800 USD,
  - cancer researchers pay \$ 10000 USD,
  - is this the ideal scenario?
- Desperately need glueless interconnect from compute chips for strongly coupled problems
  - NVlink provides this but does not scale beyond 8-16 GPU's
  - On-package Omnipath Intel parts interesting; dual rail system uptake limited due to switch and cable costs
  - Distributed machine learning may end up driving this
- Linux VM has significant shortcomings for cluster nodes. 2MB base page Linux-for-HPC, or lightweight kernel?
- I have not addressed FPGA's: because they do not have the memory, I/O systems, flop/s to compete with Nvidia