Multi-threaded ATLAS Simulation on Intel Knights Landing Processors

Steve Farrell, Paolo Calafiura, Charles Leggett, Vakho Tsulaia, Andrea Dotti,
on behalf of the ATLAS collaboration

CHEP 2016
San Francisco

Ernest Orlando Lawrence
Berkeley National Laboratory

Oct 14, 2016

Overview

- Many-integrated-core (MIC) architectures
  - Intel Xeon Phi Knights Landing processors
  - MIC-equipped supercomputers
- Atlas multi-threaded simulation
  - Design and parallelism
- Performance measurements
  - Throughput and memory scaling
  - CPU profiling studies

Thanks to NERSC and Intel for providing
KNL machines to test our code on!

Setting the stage

- We’re now deep into the multi-core era
- Computer processor devices of today continue to push the limits of parallelization

Setting the stage

- We’re now deep into the multi-core era
- Computer processor devices of today continue to push the limits of parallelization
- Consider the Nvidia general-purpose GPUs

Powerful, but hard to adopt widely in ATLAS

Setting the stage

- We’re now deep into the multi-core era
- Computer processor devices of today continue to push the limits of parallelization
- Consider the Nvidia general-purpose GPUs

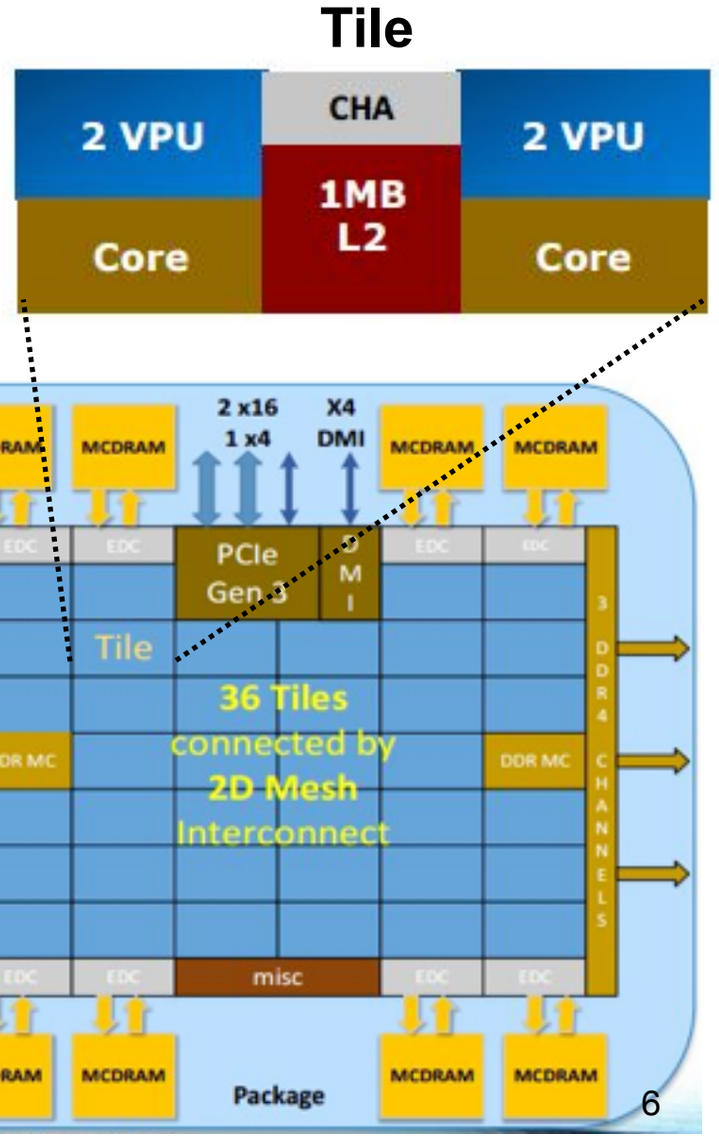
Powerful, but hard to adopt widely in ATLAS

- Intel’s answer: a highly parallel device called Xeon Phi

Can ATLAS utilize such a device effectively?

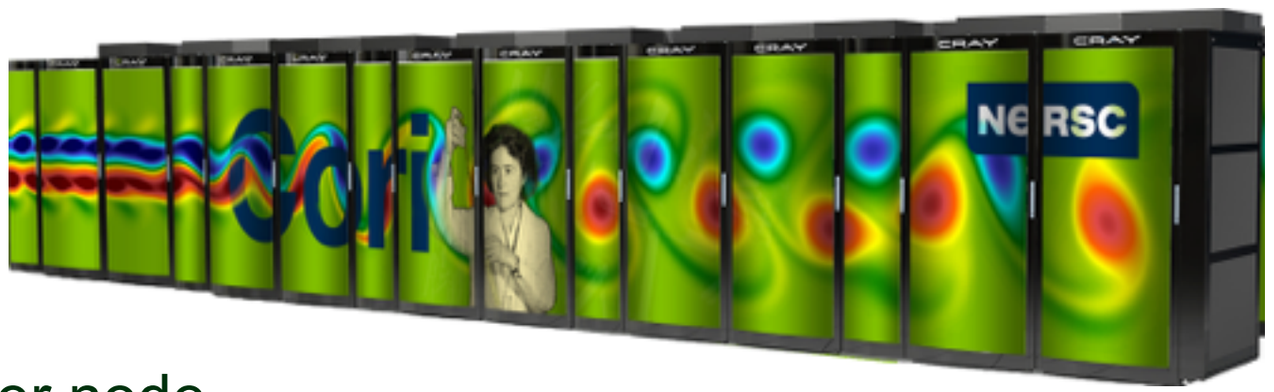
Intel Xeon Phi processors

- The Intel Many-Integrated-Core (MIC) architecture
  - Lots of power-efficient simple cores with 4-way hardware threads and deep vector registers
  - Linux OS with familiar programming model
- The current generation: ***Knights Landing (KNL)***
  - self-hosted or co-processor
  - fully x86 compatible
  - up to 72 Airmont cores (14nm process)
  - AVX-512 SIMD instruction support
  - up to 384 GB of DDR4
  - 8-16 GB of high-bandwidth MCDRAM
    - Flat, cache, and hybrid access modes



Xeon Phi supercomputers

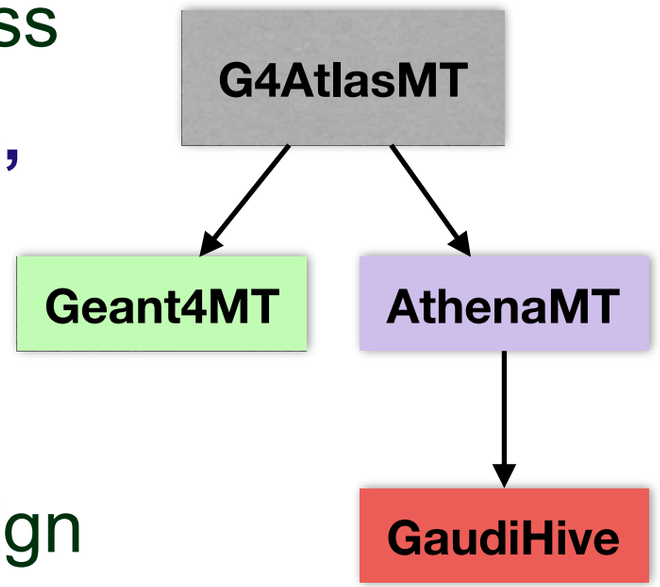
- **Cori (phase 2) at NERSC**
  - Coming online this month!
  - 9,304 KNL nodes with 68 cores each
    - 2.5 million threads
  - 96 GB DDR4 and 16 GB MCDRAM per node
  - “Burst-buffer” file system
  - 1,630 Haswell nodes with 32 cores each (phase 1)



- **Theta at Argonne**
  - Stepping-stone to Aurora
  - >2,500 KNL nodes
  - 192 GB DDR4 and 16 GB MCDRAM per node
- **Aurora at Argonne**
  - >50,000 nodes with 3rd-gen Knights Hill processors
  - To be delivered in 2019

Multi-threaded ATLAS simulation

- **ATLAS simulation could be a good application for KNL**
  - Known to be CPU intensive
  - Relatively little I/O, meta-data, and database access
- **To run effectively on modern/future architectures, we're migrating to a multi-threaded design**
  - Leveraging concurrency support in Geant4MT and AthenaMT/GaudiHive
  - Refactoring the infrastructure to a thread-safe design
- **However, multi-threading may not be enough for KNL**
  - There may be scalability limitations
  - Bad code layout and memory access patterns may hurt performance
  - Simulation code known not to vectorize well



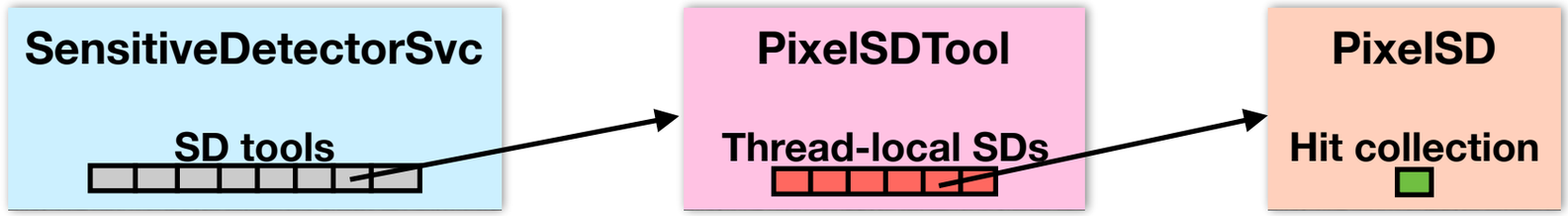
See C. Leggett's slides from Monday for AthenaMT details and motivations
See A. Dotti's slides from Tuesday for performance of Geant4 standalone on KNL

Application details

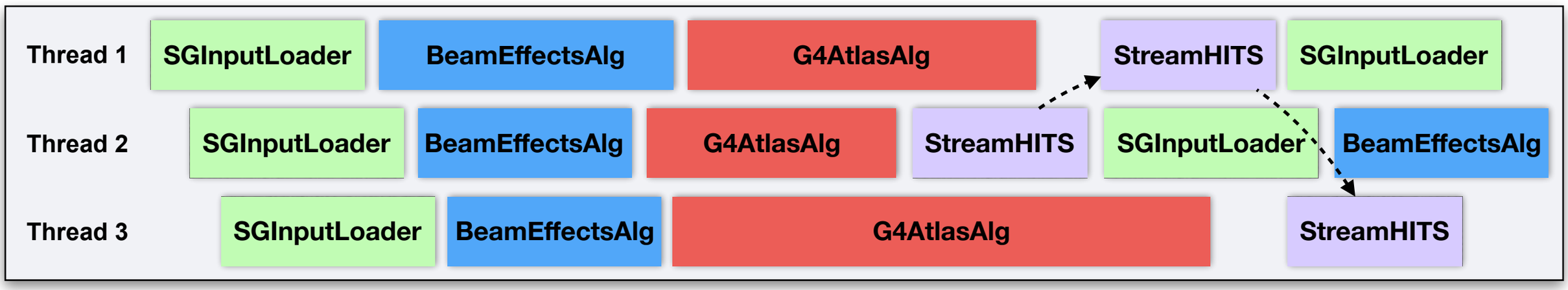
- **Thread-safe design**

More details in A. Di Simone's slides from Tue

  - Composition model of Gaudi/Athena components that create + manage thread-local G4 components



  - Thread-local workspaces setup via new thread initialization tool framework mechanism
  - Processing algorithms are cloned to execute on different events concurrently
  - One instance of the output stream algorithm (StreamHITS) services all worker threads

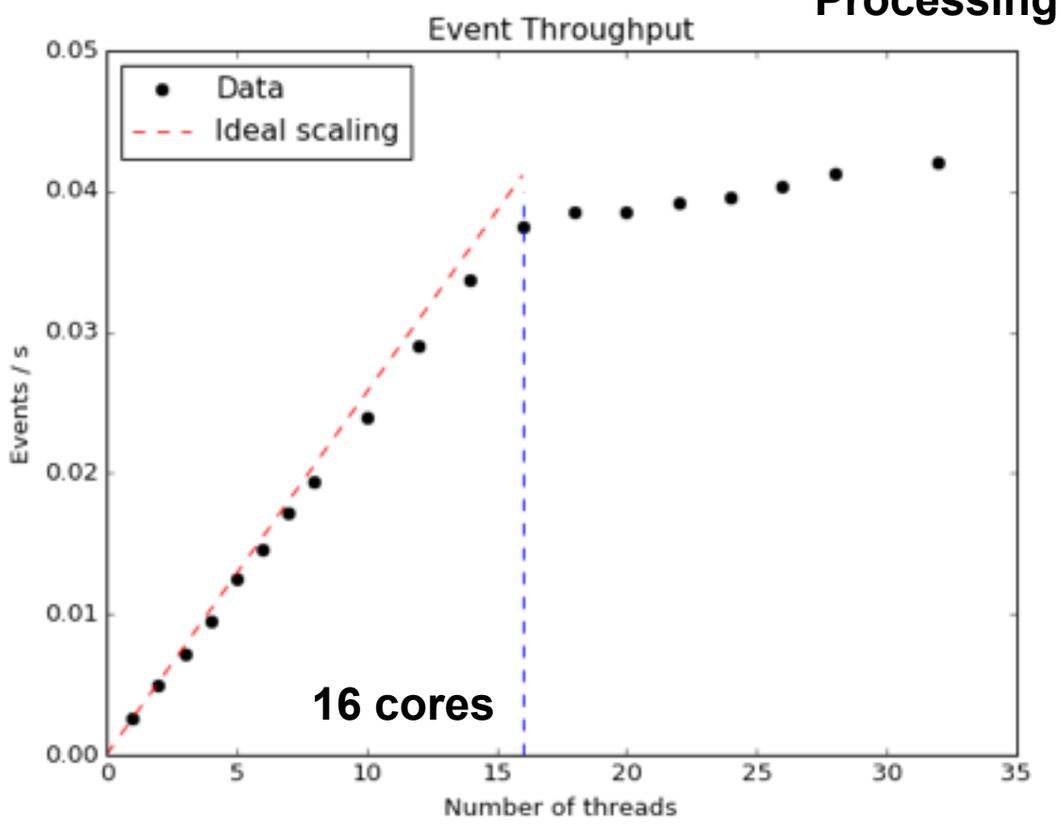


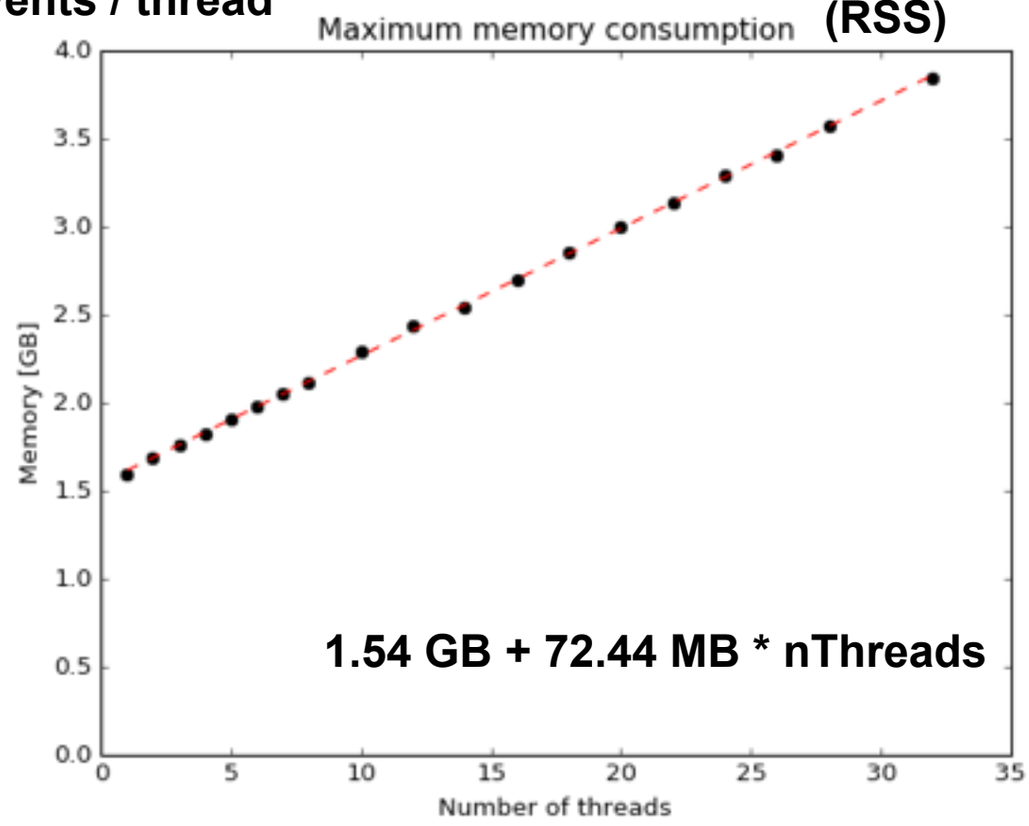
- **Migration status**

  - Complete: geometry, physics, user actions, most sensitive detectors, truth writing, magnetic field, frozen shower library
  - Ongoing development: LAr sensitive detectors, fast simulations

Scaling on a Xeon* - Zττ sample

Processing 50 events / thread



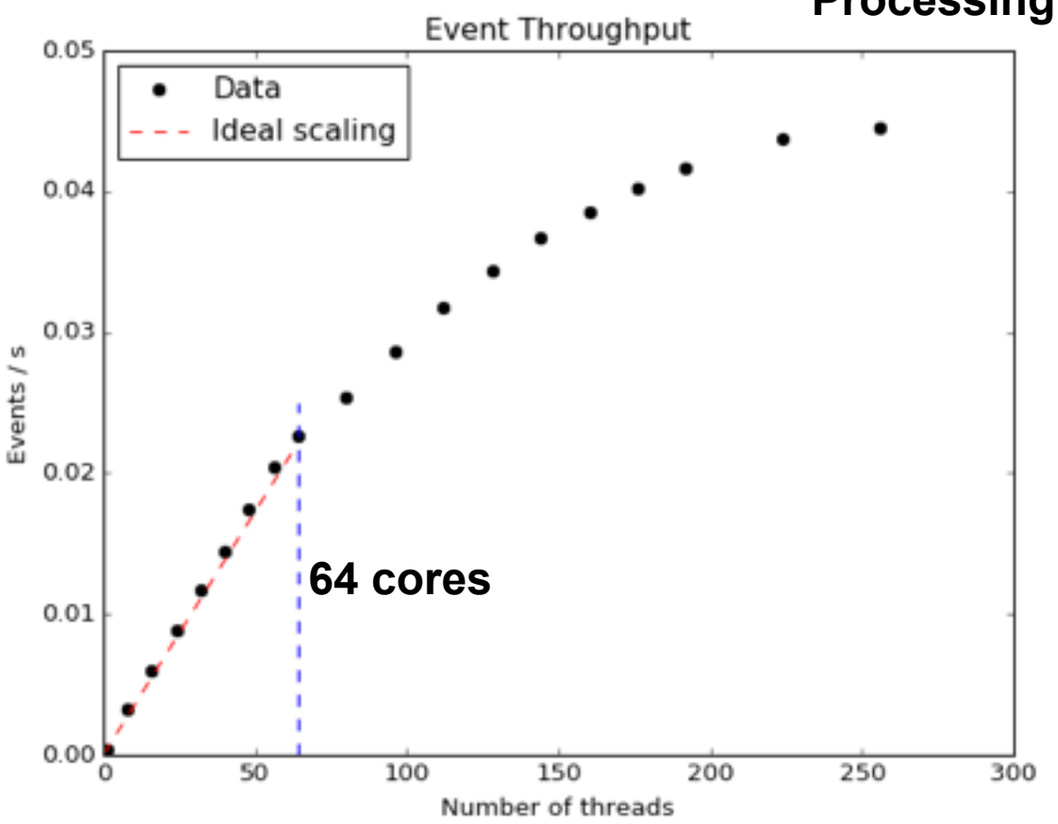


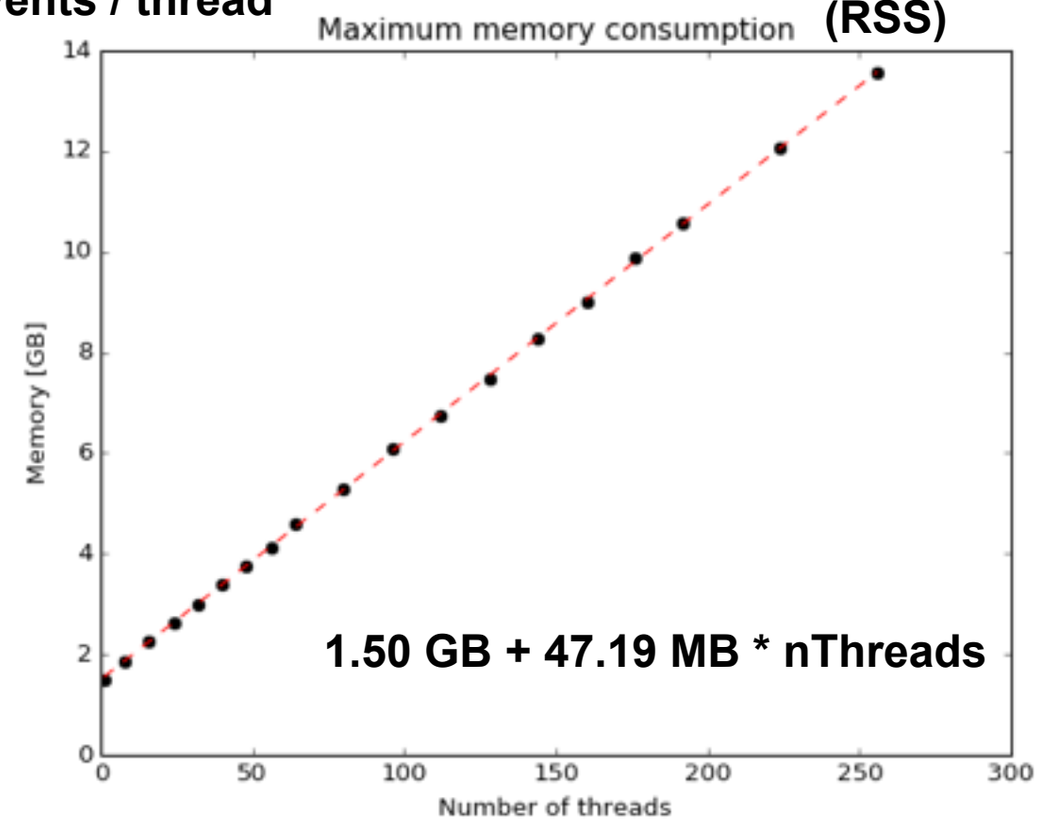
- Event throughput scales very well up to the number of physical cores and then plateaus with marginal gain from hyper-threading
- Memory scaling is linear and gradual

*Intel Xeon CPU E5-2650 v2 @ 2.60GHz

Scaling on a Xeon Phi* - Zττ sample

Processing 10 events / thread



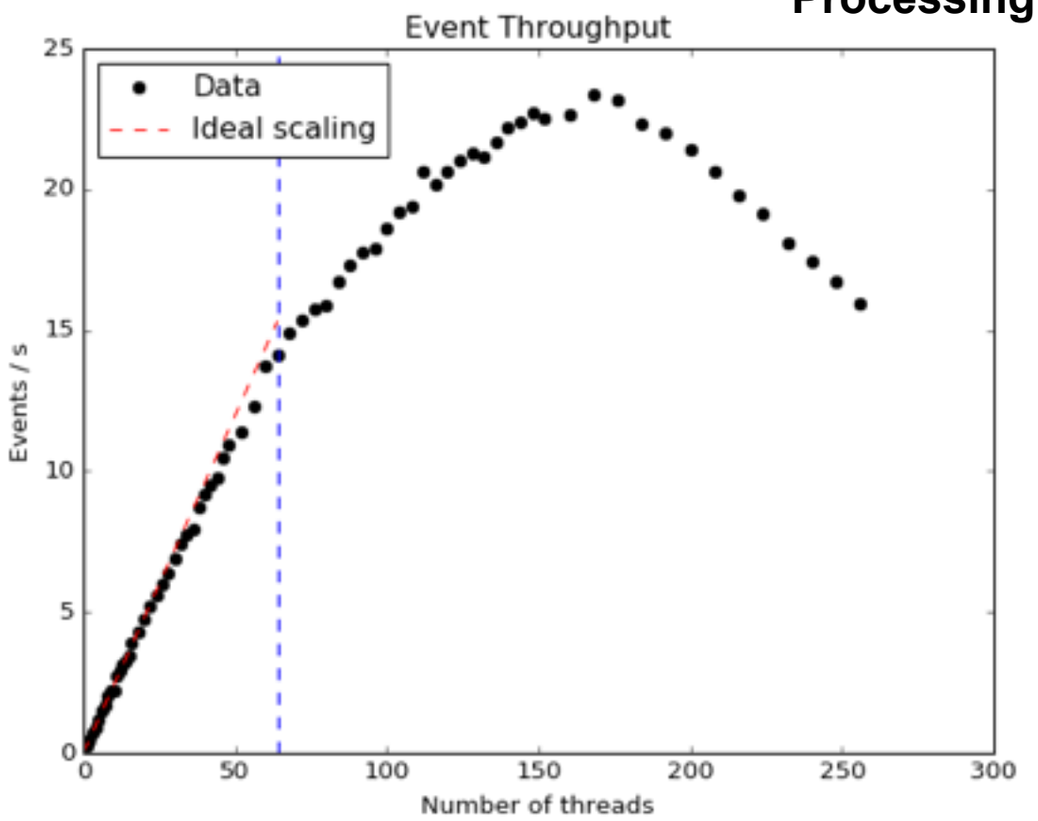


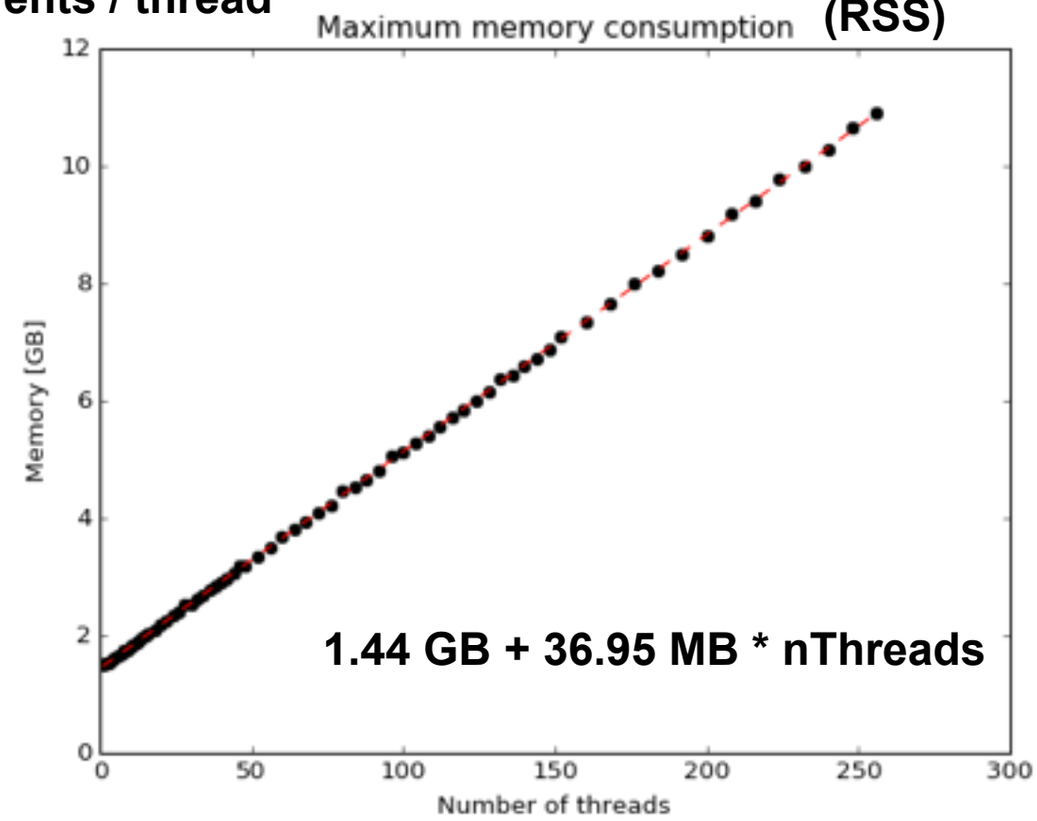
- As with the Xeon, we see great throughput scaling up to number of physical cores
- In this case we see substantial gains from hyper-threading
- Memory scaling is very nice, showing considerable reduction from using multi-threading

*Intel Xeon Phi CPU 7210 @ 1.30GHz

Scaling on a Xeon Phi - single-muon sample

Processing 1k events / thread





- Good scaling up to around 180 threads, followed by *negative* scaling above that!
- **In this case, the the output stream (StreamHITS) is a bottleneck!**
  - *Single algorithm instance cannot serve >180 threads at such high event rate*
- Luckily, this is an unrealistic configuration for full-simulation
  - but it might be relevant for fast-simulations in MT
- Some work is still needed to improve scaling in this type of scenario

Xeon vs. Xeon Phi summary

Xeon vs. Xeon Phi summary

Single thread

Throughputs [events/s]

Sample	Ivy-bridge Xeon	KNL Xeon Phi
Zττ	0.00257	0.000345
single-μ	1.38	0.239

=> Per-core performance is 6-7 times worse on KNL than on the Xeon!

Xeon vs. Xeon Phi summary

Throughputs [events/s]

Single thread

Sample	Ivy-bridge Xeon	KNL Xeon Phi
Zττ	0.00257	0.000345
single-μ	1.38	0.239

=> Per-core performance is 6-7 times worse on KNL than on the Xeon!

Full machine
(or best throughput)

Zττ	0.0421	0.0445
single-μ	24.6	23.2*

***176 threads**

=> Best throughput on Xeon Phi similar to 16-core Ivy bridge Xeon

Xeon vs. Xeon Phi summary

Throughputs [events/s]

Single thread

Sample	Ivy-bridge Xeon	KNL Xeon Phi
Zττ	0.00257	0.000345
single-μ	1.38	0.239

=> Per-core performance is 6-7 times worse on KNL than on the Xeon!

Full machine
(or best throughput)

Zττ	0.0421	0.0445
single-μ	24.6	23.2*

***176 threads**

=> Best throughput on Xeon Phi similar to 16-core Ivy bridge Xeon

- **Performance falls short of KNL potential**
  - Usual comparison is to a 32-core Haswell!
- **Can we understand what's going on under the hood?**

Profiling the application

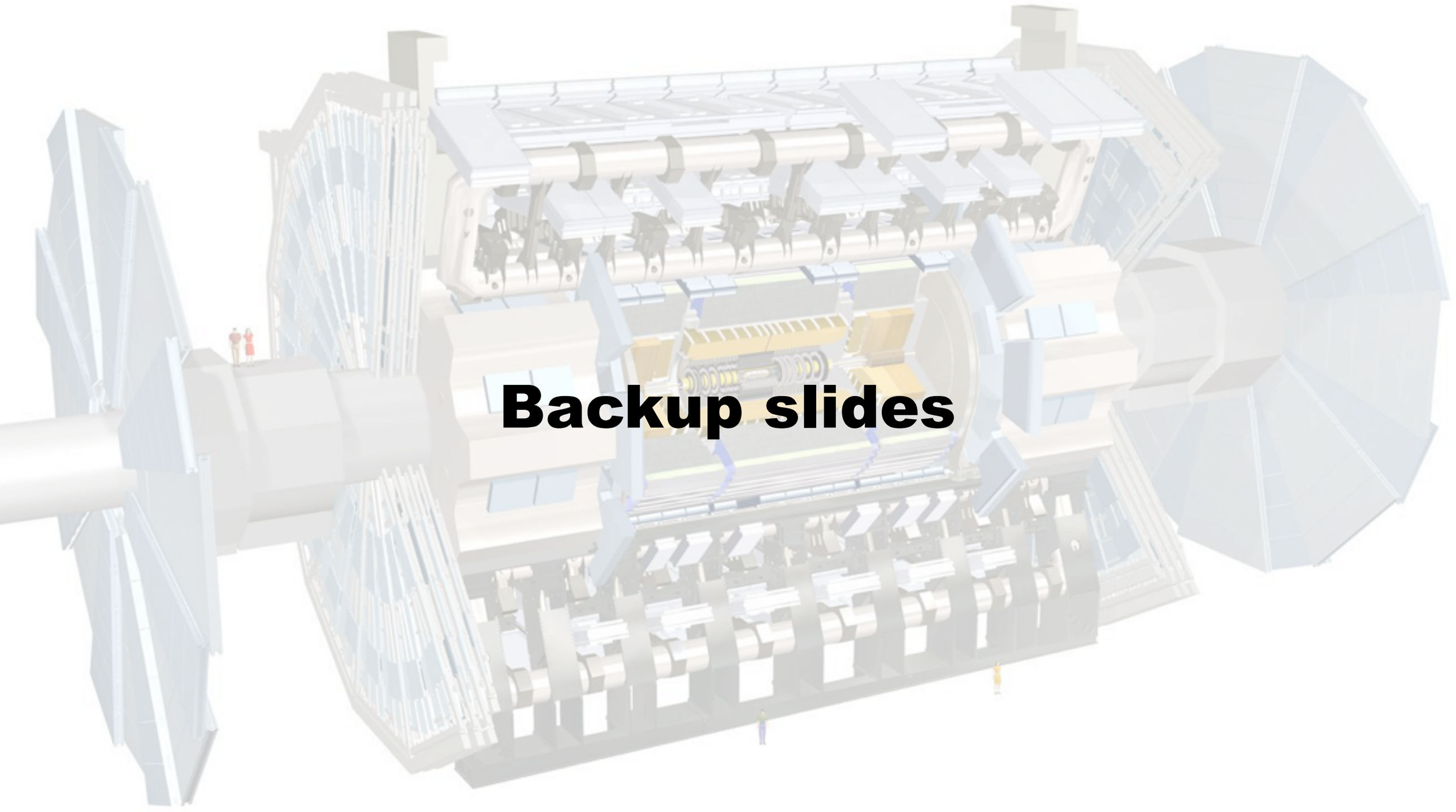
- Using VTune, we can start to understand the performance differences between the Xeon and Xeon Phi architectures
  - These results measured with a Zμμ sample and a single worker thread

Architecture	CPI rate	Front-end bound	ICache misses	Bad speculation	Back-end bound
KNL	3.0	60.2%	0.96	2.4%	18.6%
Haswell	0.9	31.5%	0.086	11.7%	27.6%

- On KNL, the application seems to be held up in the instruction front-end, with a high clocks-per-instruction rate of 3.0!
  - High rate of instruction cache misses
  - Seems to be due to relatively poor handling of large ATLAS+G4 code size

Conclusion

- ATLAS can now run a nearly complete multi-threaded simulation setup in AthenaMT
  - Performance looks good on a Xeon
- We've shown that we *can* utilize Xeon Phi Knights Landing machines
  - Scalability is generally good (except some "extreme" configurations)
  - Throughput similar to typical grid machine (16-core Ivy Bridge)
- However, we still have a lot of room for improvement to use them *effectively*
  - Front-end, CPI issues
  - Would like to do as well as a 32-core Haswell
- Plenty of work to be done
  - Try to improve on the results from the profiler
  - Improve our I/O layer for configurations with high event rate
  - Find opportunities for vectorization



Backup slides

Thermal design power

Ivy bridge Xeon E5-2650 v2 8-core	**100 W**
Haswell Xeon E7-8860v3 16-core	**165 W**
Knights Landing Xeon Phi 7210 64-core	**215 W**

- Typical grid machines have 2 Ivy-Bridge procs (16-core) **~200 W**
- Cori phase 1 nodes have 2 Haswell procs (32-core) **~330 W**

Intel Many-Integrated-Core architecture

- A “supercomputer on a chip”
  - Lots of threads, wide vector registers, with low power footprint
  - Particularly suited to highly-parallel, CPU-bound applications

- The Xeon Phi product line:

Knights Corner (KNC)	**Knights Landing (KNL)**	**Knights Hill (KNH)**
previous generation	current generation	maybe 2017
57-61 Pentium cores (~1GHz)	72 Airmont cores (3x faster)	60-72 Silvermont cores
6-16 GB on-chip RAM	8-16 GB MCDRAM	???
coprocessor only	up to 384 GB RAM	
	host or coprocessor	

- Supercomputers:

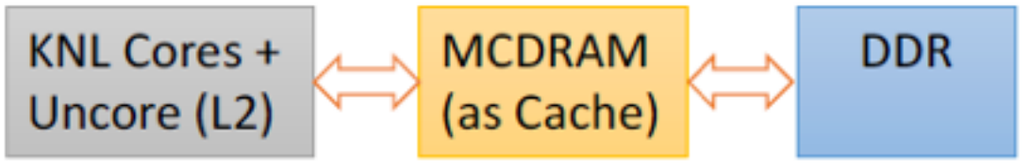
• Tianhe-2 @ NSCC-GZ	• Cori @ NERSC	• Aurora @ ANL
• Stampede @ TACC	• Theta @ ANL	

Haswell vs. KNL memory

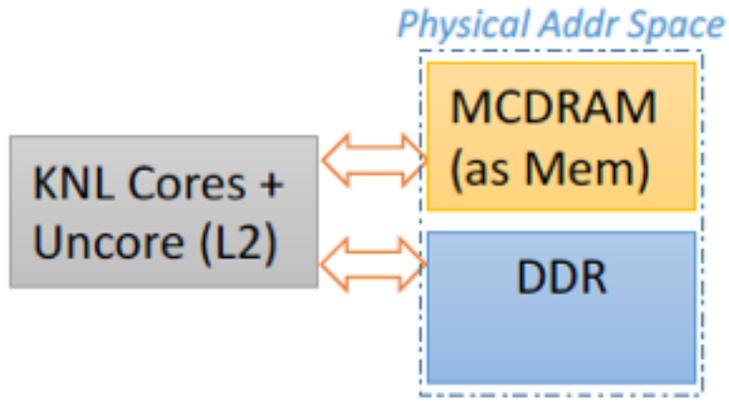
	Latency Haswell Nanosec (clocks)	Latency KNL Nanosec(clocks)	Bandwidth Haswell GB/sec Stream Triad	Bandwidth KNL GB/sec Stream Triad
L1 Cache	2.7 (6)	2.9 (4)		
L2 Cache	8.11 (19)	13.6 (19)		
MCDRAM (Cache)	24.35 (56) (Level 3)	173.5 (243)		329
MCDRAM (Flat)		174.2 (244)		486
DDR (Flat)		151.3 (212)	102	90
DDR (Cache)				59.3

MCDRAM Modes

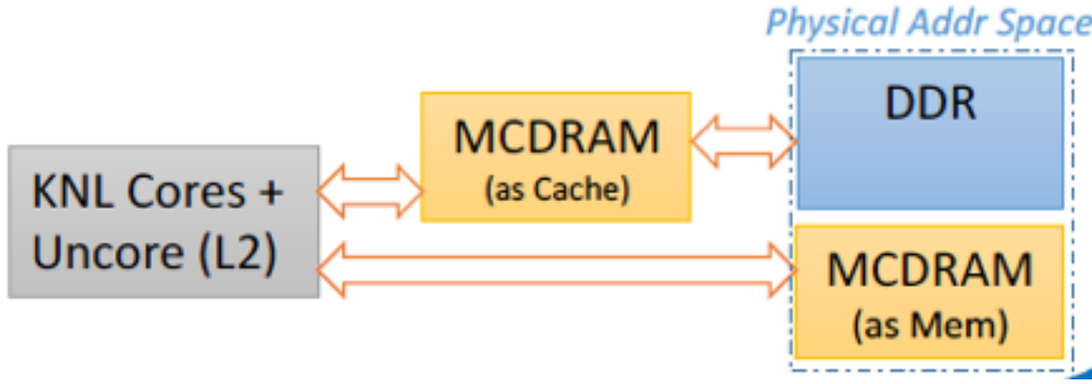
- Cache mode
  - No source changes needed to use
  - Misses are expensive (higher latency)
    - Needs MCDRAM access + DDR access



- Flat mode
  - MCDRAM mapped to physical address space
  - Exposed as a NUMA node
    - Use numactl --hardware, lscpu to display configuration
  - Accessed through memkind library or numactl

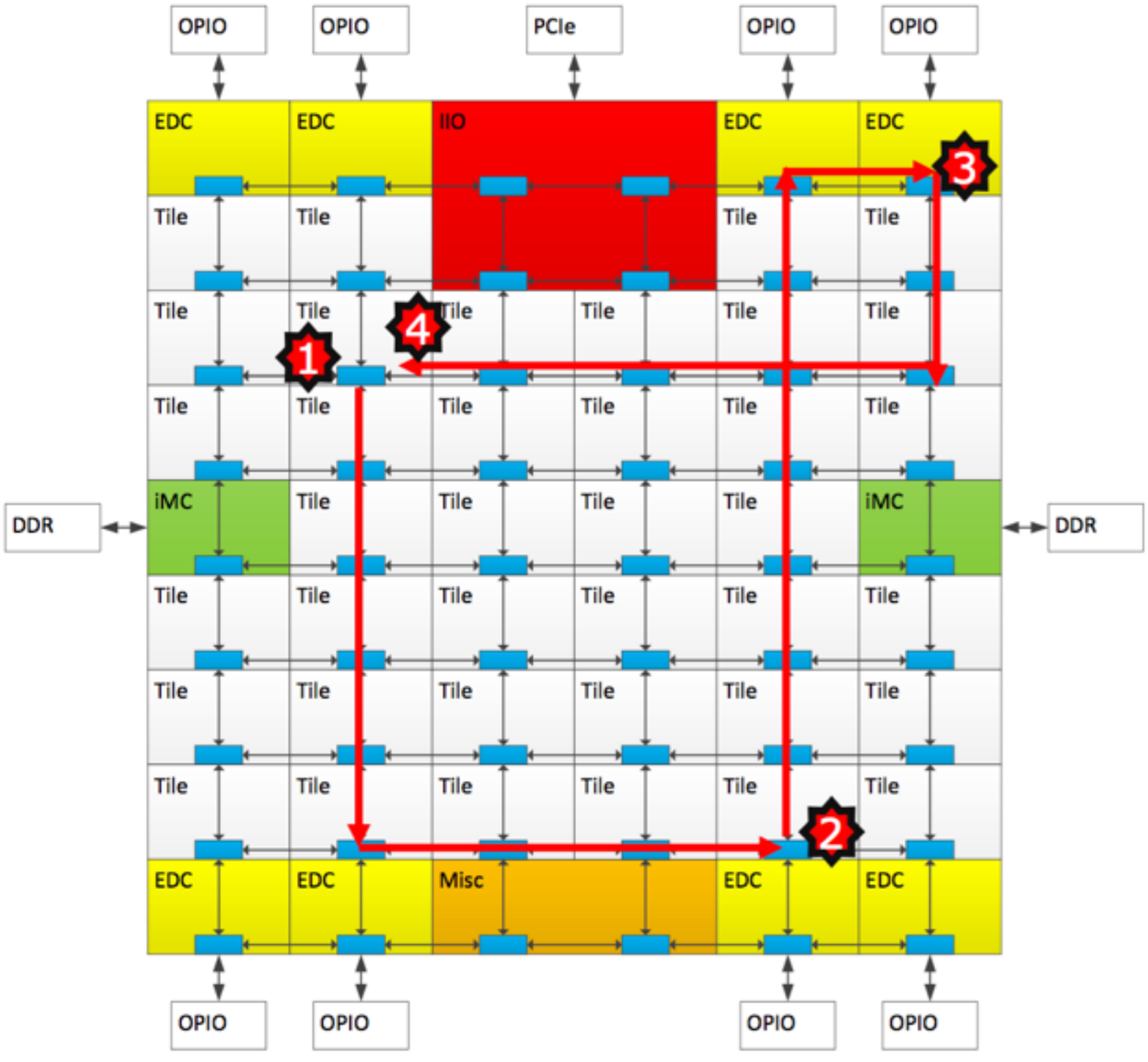


- Hybrid
  - Combination of the above two
    - E.g., 8 GB in cache + 8 GB in Flat Mode



https://anl.app.box.com/v/IXPUG2016-presentation-44

Cluster mode: All-to-All



Address uniformly hashed across all distributed directories

Typical Read L2 miss

1. L2 miss encountered

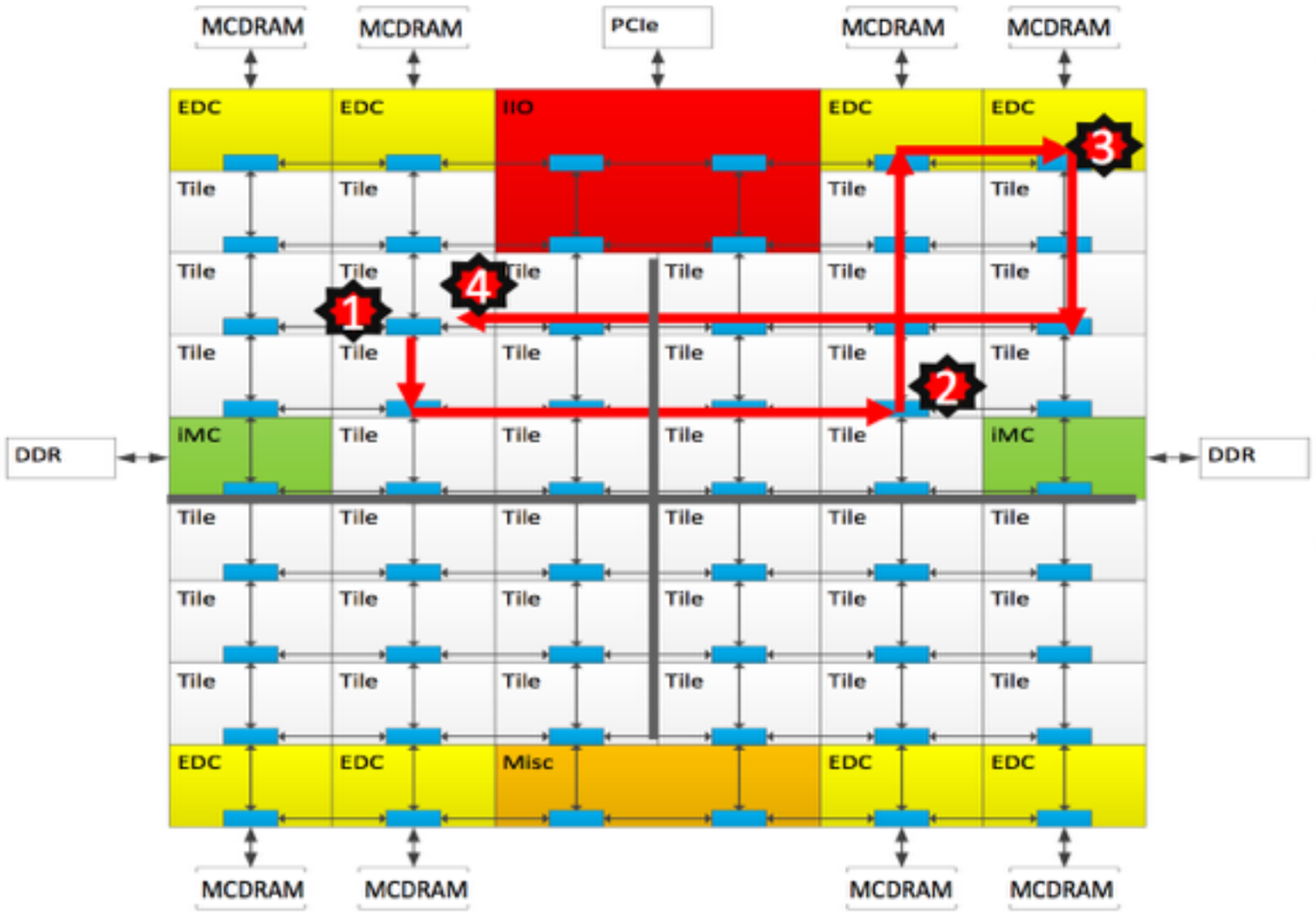
2. Send request to the distributed directory

3. Miss in the directory. Forward to memory

4. Memory sends the data to initial requestor

https://anl.app.box.com/v/IXPUG2016-presentation-3

Cluster Mode: Quadrant



Chip divided into four virtual Quadrants

Address hashed to a Directory in the same quadrant as the Memory

Affinity between the Directory and Memory

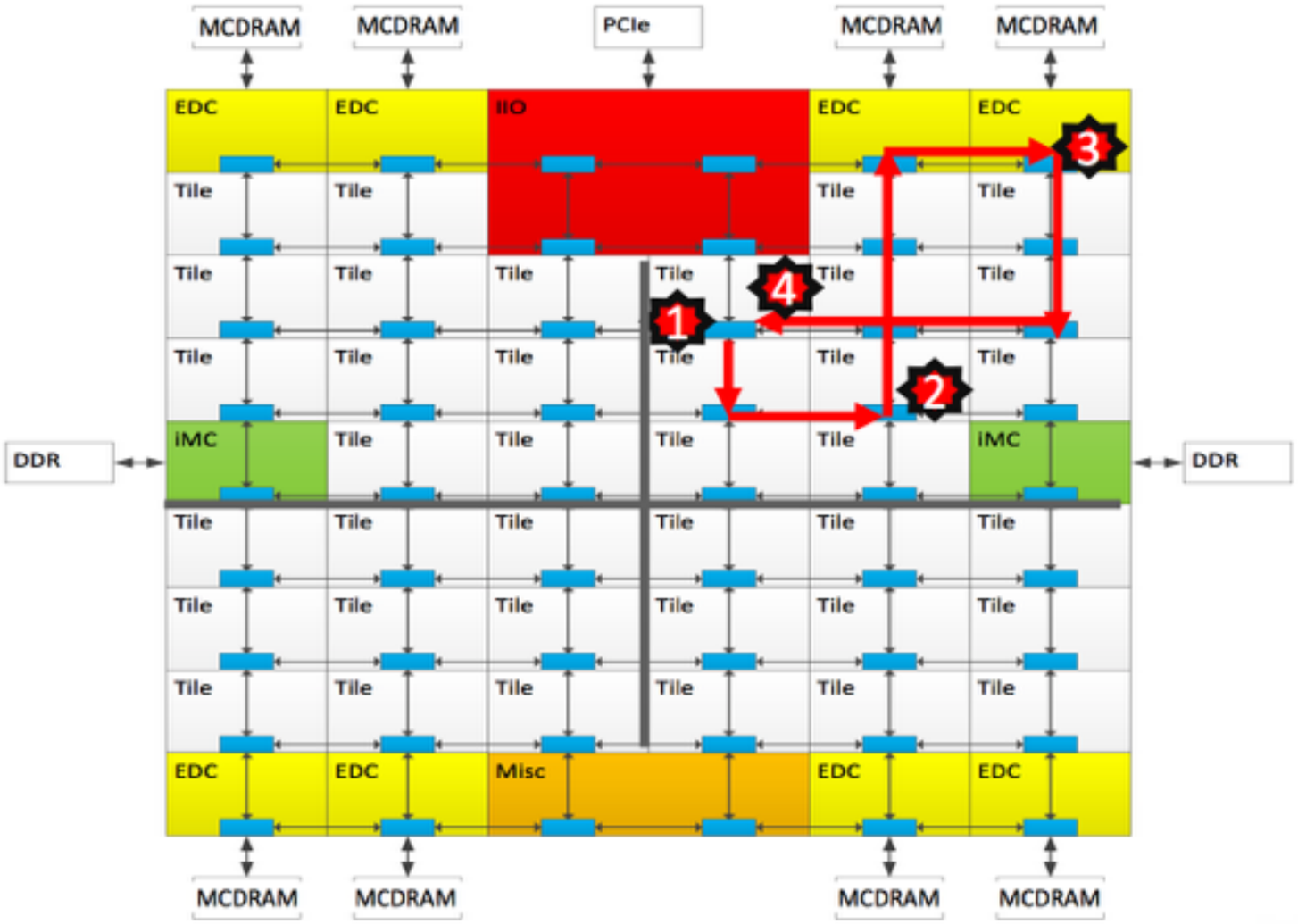
Lower latency and higher BW than all-to-all. SW Transparent.

1) L2 miss, 2) Directory access, 3) Memory access, 4) Data return

15

https://anl.app.box.com/v/IXPUG2016-presentation-3

Cluster Mode: Sub-NUMA Clustering (SNC)



1) L2 miss, 2) Directory access, 3) Memory access, 4) Data return

Each Quadrant (Cluster) exposed as a separate NUMA domain to OS.

Looks analogous to 4-Socket Xeon

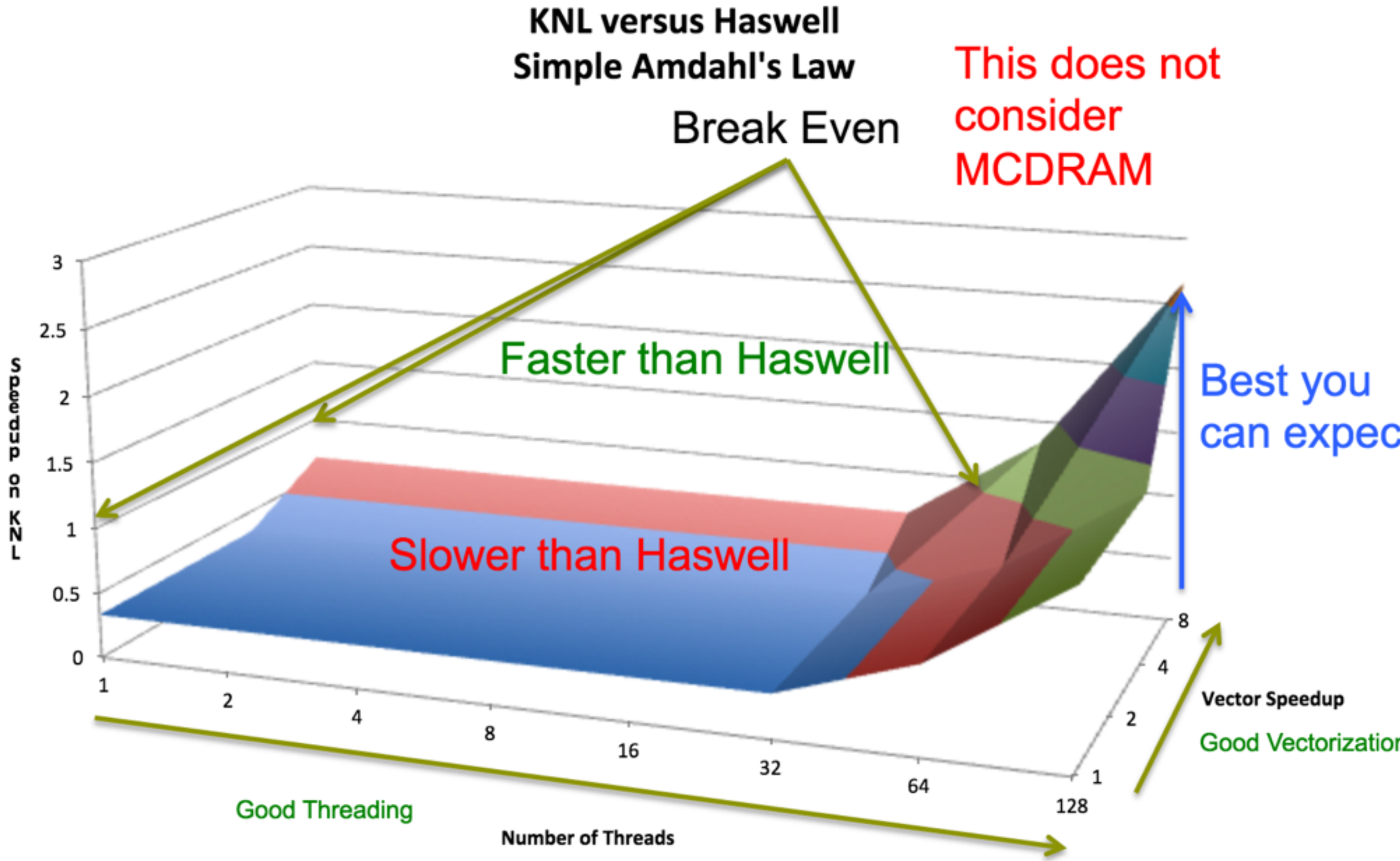
Affinity between Tile, Directory and Memory

Local communication. Lowest latency of all modes.

SW needs to NUMA optimize to get benefit.

https://anl.app.box.com/v/IXPUG2016-presentation-3

Performance potential



https://anl.app.box.com/v/IXPUG2016-presentation-3

Machines studied

The aibuild machine at CERN
Kernel: 2.6.32-573.18.1.el6.x86_64
Model: Intel(R) Xeon(R) CPU E5-2650 v2 @ 2.60GHz
Cores: 16 Ivy-bridge cores (32 HT)

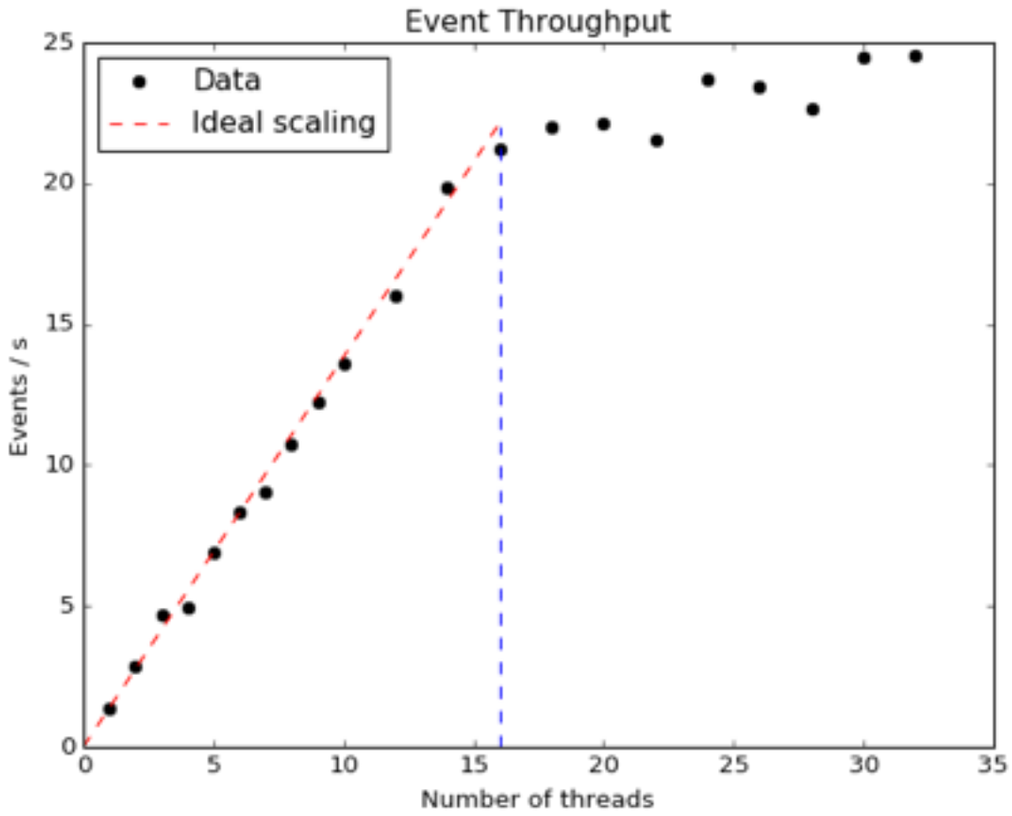
NERSC's Cori Phase 1: http://www.nersc.gov/users/computational-systems/cori/cori-phase-i/
Nodes: 1,630
Kernel: 3.12.51-52.31.1_1.0600.9146-cray_ari_c
Model: (R) Xeon(R) CPU E5-2698 v3 @ 2.30GHz
Cores: 32 Haswell cores (64 HT)

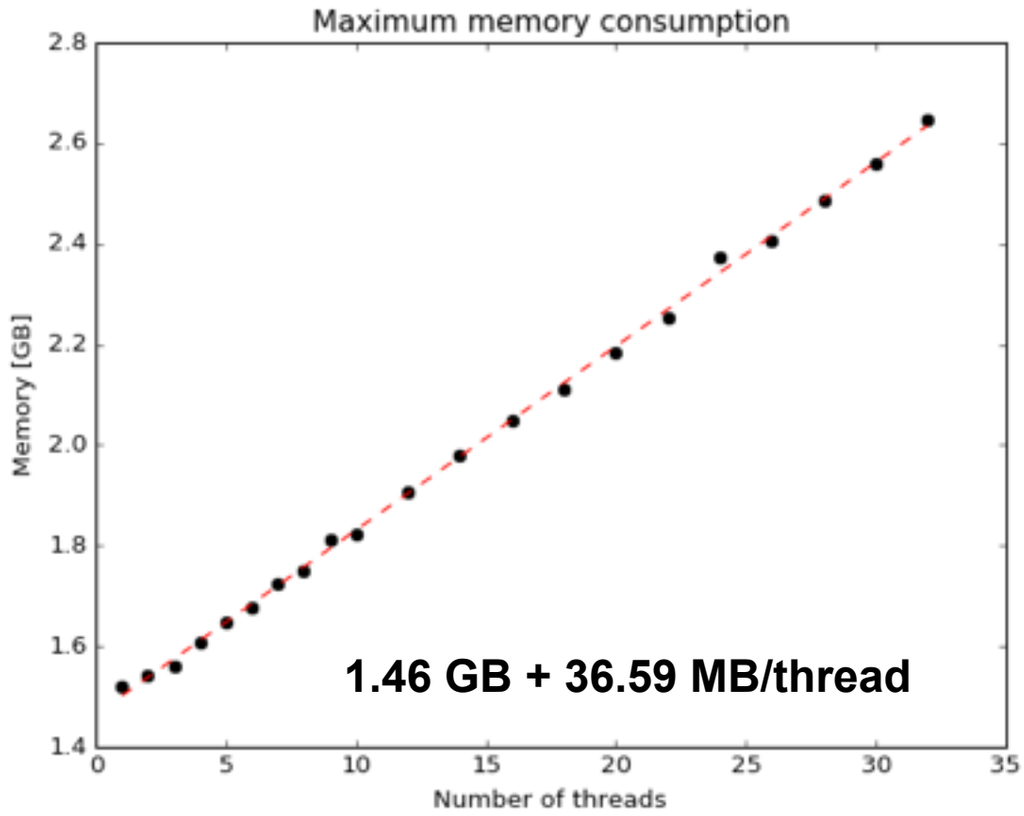
NERSC's Carl KNL cluster
Kernel: 3.12.49-11.1.xppsl_1.3.3.151-default
Model: Intel(R) Xeon Phi(TM) CPU 7210 @000000 1.30GHz
Cores: 64 (256 HT)
Configuration: quad+flat or quad+cache

Carl also has "bin1" nodes, which should be the same as Cori Phase 2:
Kernel: 3.12.49-11.1.xppsl_1.3.3.151-default
Model: Intel(R) Xeon Phi(TM) CPU 7250 @ 1.40GHz
Cores: 68 (272 HT)
Configuration: quad-flat or quad-cache

Intel Endeavour KNL cluster (external dev queue)
Kernel: 3.10.0-229.20.1.el6.x86_64.knl2
Model: Intel(R) Xeon Phi(TM) CPU 7210 @ 1.30GHz
Cores: 64 (256 HT)
Configuration: quad+flat or quad+cache

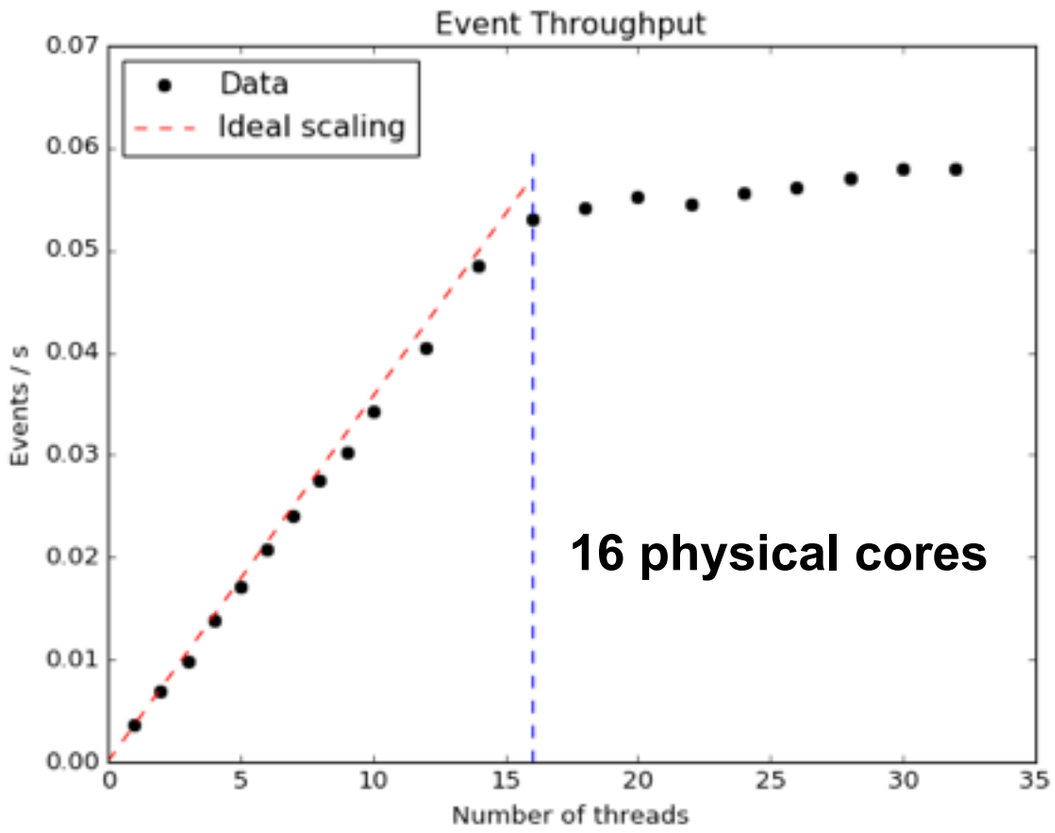
Scaling on a Xeon - single-muon sample

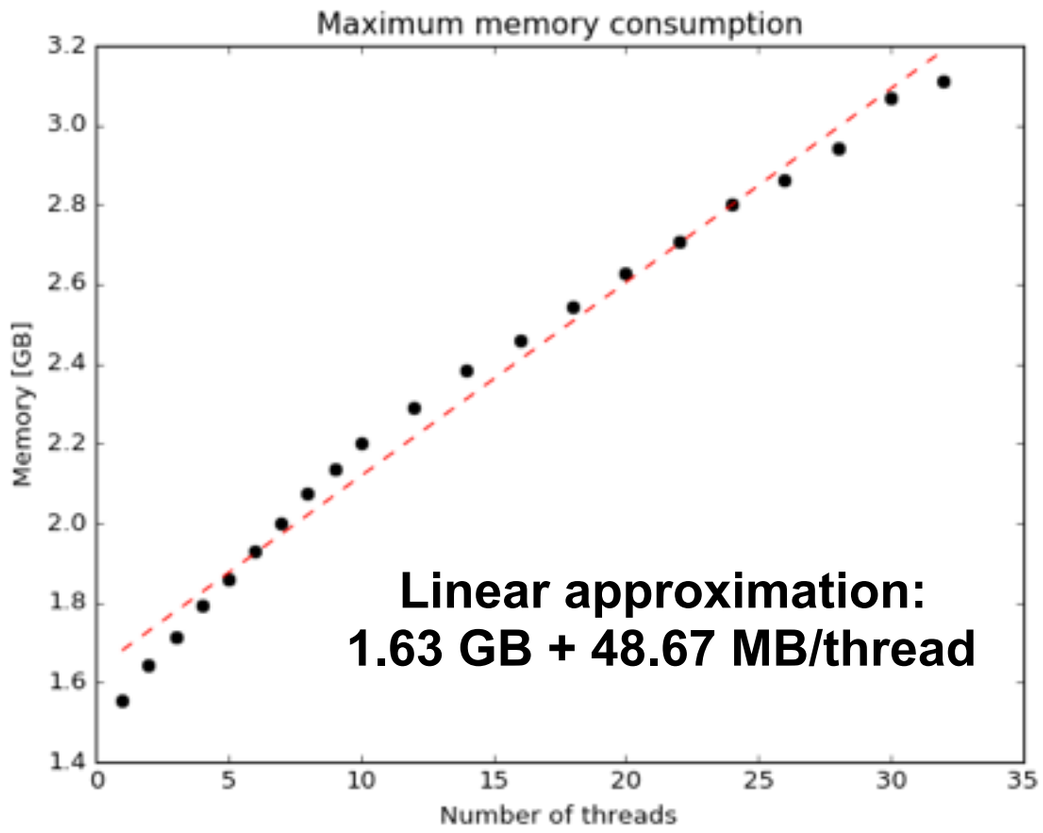




- As with the Ztautau sample, the scaling with the single-muon sample is excellent up to the physical number of cores
- The memory scaling is also good again

Scaling on a Xeon - ttbar sample

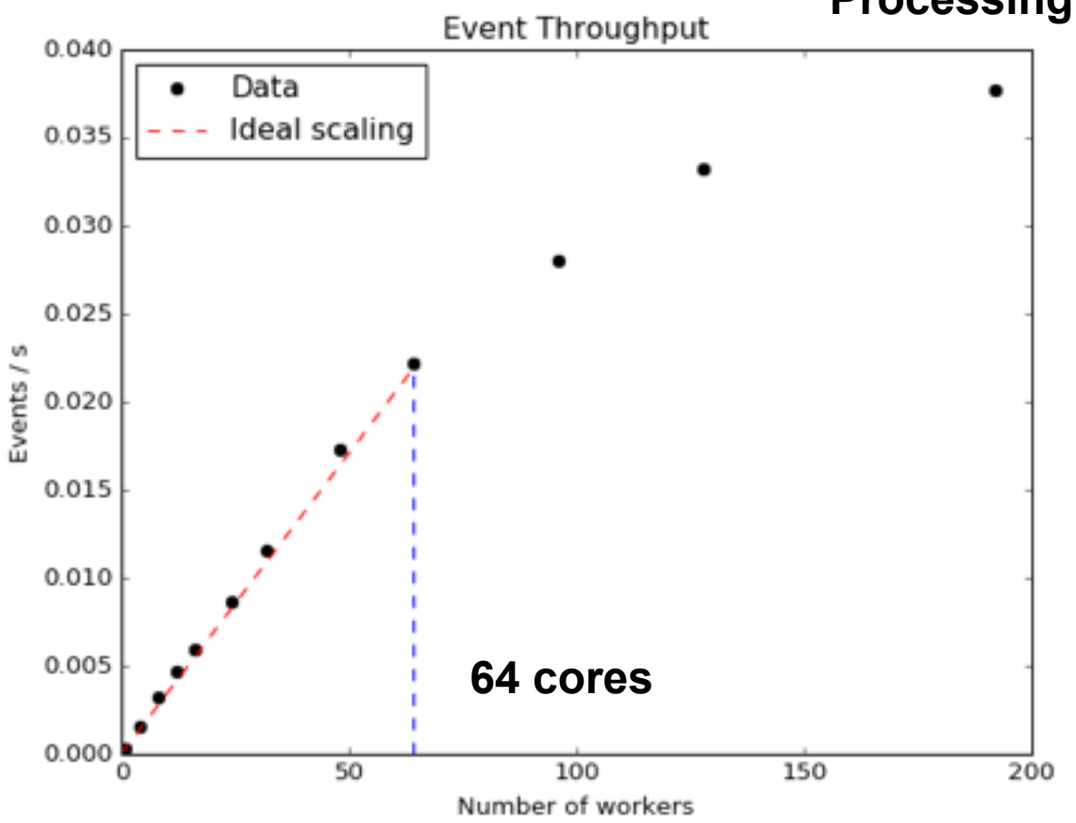


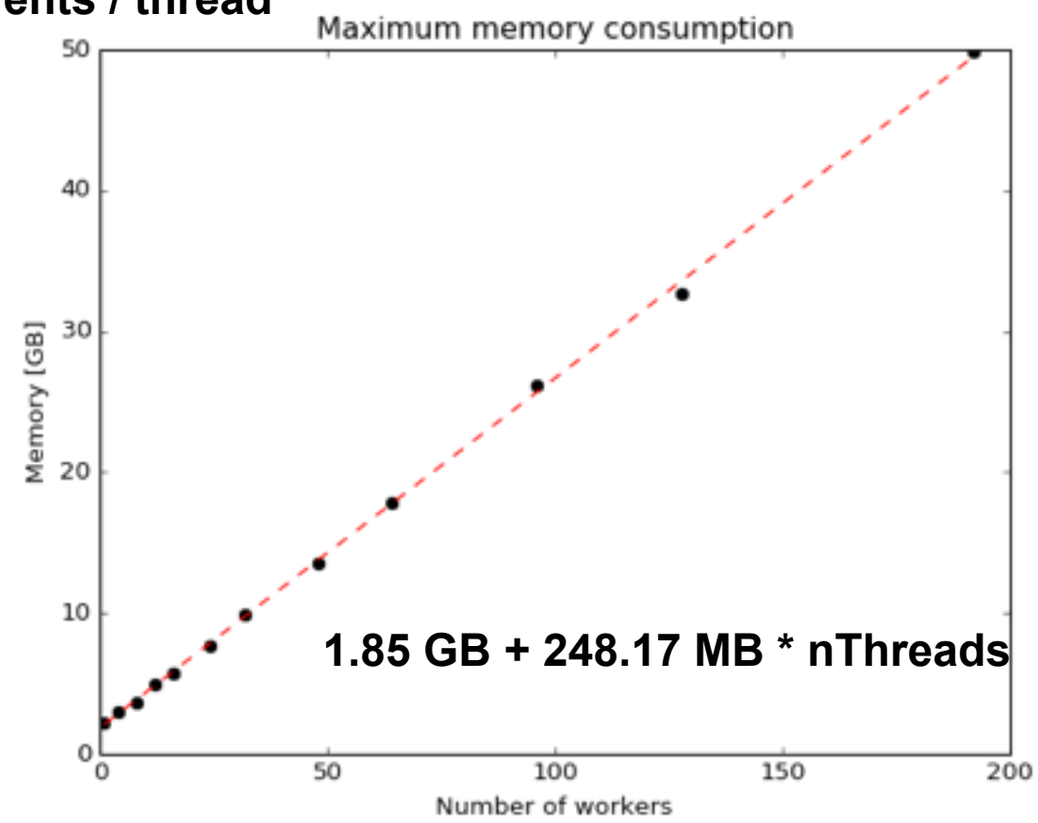


- Event throughput scales very well up to the physical number of cores, and plateaus quite abruptly in hyper-threading regime
- Memory scales nicely, showing excellent savings from sharing across threads

Multi-process scaling on Xeon Phi - Zττ sample

Processing 10 events / thread





- Processes were not pinned to cores, here, so core migration and context switching probably impact throughput
- AthenaMP shows memory savings, but per-thread increase is ~5 times larger than pure multi-threading

*Intel Xeon Phi CPU 7210 @ 1.30GHz

Xeon vs. Xeon Phi performance (single-muon)

Threads	Xeon throughput [events/s]	Xeon Phi throughput [events/s]	Throughput ratio (KNL slowdown)
1	1.38	0.24	5.78
4	4.88	0.91	5.39
6	8.24	1.47	5.62
8	10.54	1.78	5.91
12	15.17	2.76	5.50
16	20.50	3.68	5.57
24	22.51	5.16	4.36
32	24.24	7.24	3.35

=> Per-core performance is about 5.5 times worse on KNL than on the Ivy-bridge Xeon

Can we understand why?

Application hotspots

- **Hotspots on a Haswell machine** (Zμμ sample, single worker thread):

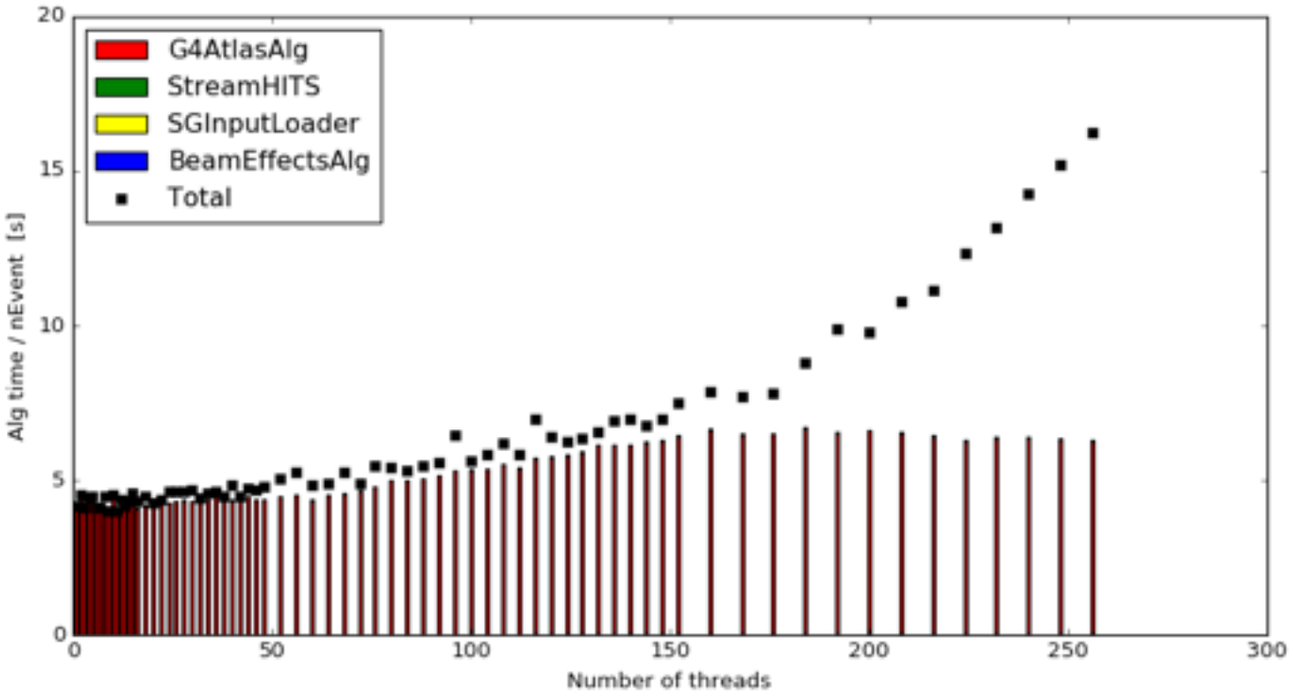
Function	Clockticks	Instructions Retired	CPI Rate	Front-End Bound(%)	Bad Speculation(%)	Back-End Bound(%)
G4PhysicsVector::Value	132,160,198,240	107,000,160,500	1.235	0.164	9.0%	0.557
__sin_avx	129,540,194,310	71,740,107,610	1.806	0.344	26.9%	0.197
sincos	118,360,177,540	44,420,066,630	2.665	0.414	27.6%	0.170
__cos_avx	117,680,176,520	25,380,038,070	4.637	0.459	24.2%	0.203
LArWheelCalculator_Impl::DistanceCalculatorSaggingOff::	110,760,166,140	196,640,294,960	0.563	0.051	8.8%	0.355
__ieee754_log_avx	75,140,112,710	36,560,054,840	2.055	0.338	22.5%	0.283
__ieee754_atan2_avx	72,300,108,450	56,620,084,930	1.277	0.373	21.3%	0.185
G4Navigator::LocateGlobalPointAndSetup	67,680,101,520	49,380,074,070	1.371	0.313	6.2%	0.471
LArWheelCalculator::parameterized_sin	67,460,101,190	92,560,138,840	0.729	0.076	24.9%	0.284
MagField::AtlasFieldSvc::getField	58,240,087,360	57,800,086,700	1.008	0.144	19.5%	0.472

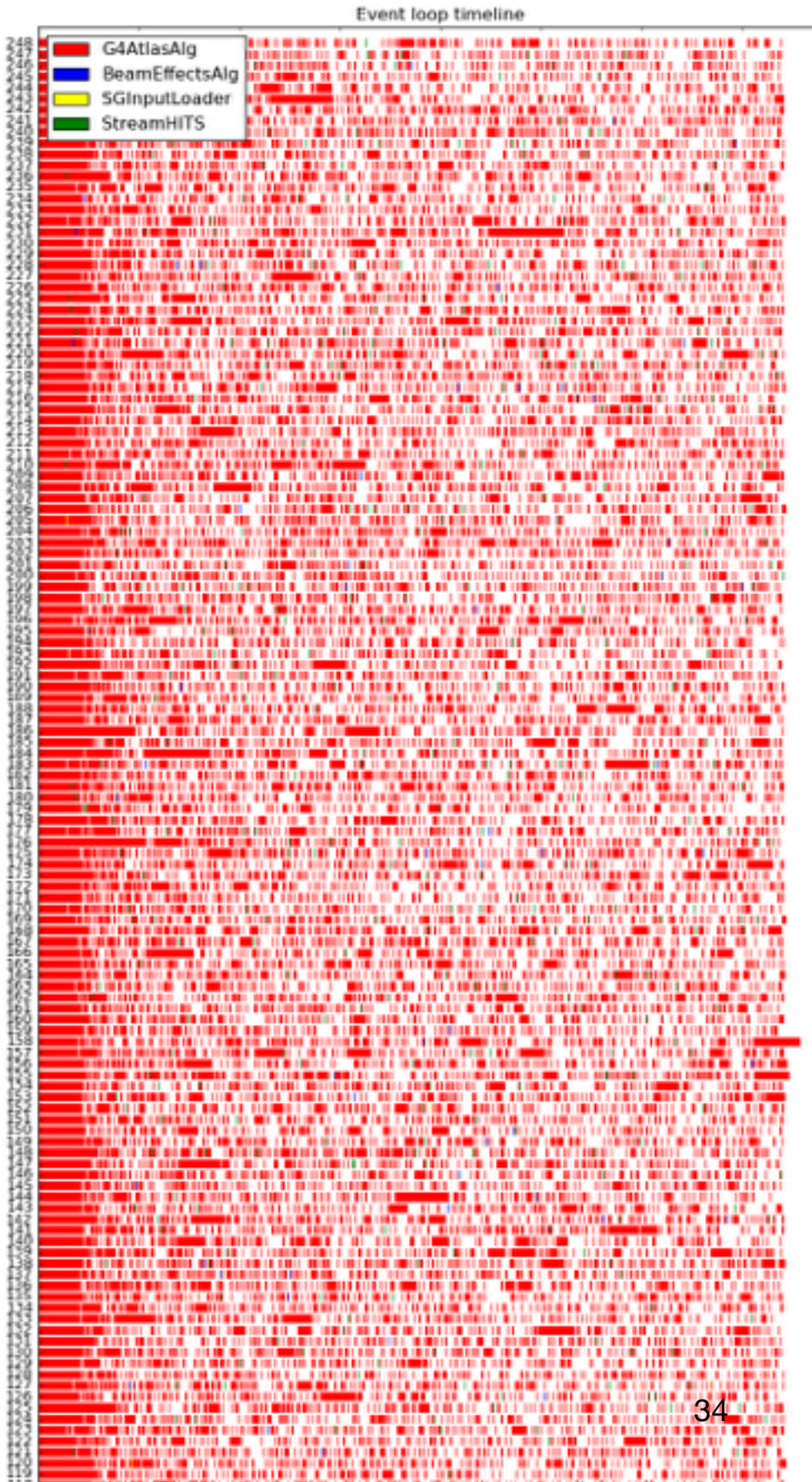
- **Hotspots on a KNL machine** (same config):

Function	Clockticks	Instructions Retired	CPI Rate	Front-End Bound(%)	Bad Speculation(%)	Back-End Bound(%)
G4PhysicsVector::Value	333,820,500,730	82,760,124,140	4.034	0.567	2.8%	0.270
LArWheelCalculator::parameterized_sin	257,140,385,710	189,920,284,880	1.354	0.456	0.3%	0.125
LArWheelCalculator_Impl::DistanceCalculatorSaggingOff::	235,160,352,740	127,340,191,010	1.847	0.161	4.1%	0.443
G4Navigator::LocateGlobalPointAndSetup	194,320,291,480	42,160,063,240	4.609	0.620	1.2%	0.246
__tls_get_addr	193,620,290,430	49,360,074,040	3.923	0.638	1.4%	0.207
G4SteppingManager::DefinePhysicalStepLength	189,300,283,950	64,240,096,360	2.947	0.658	2.2%	0.150
__sin_avx	174,540,261,810	58,140,087,210	3.002	0.441	1.3%	0.354
G4Navigator::ComputeStep	166,540,249,810	41,240,061,860	4.038	0.767	0.3%	0.093
MagField::AtlasFieldSvc::getField	158,660,237,990	55,860,083,790	2.840	0.451	1.9%	0.333
BFieldCache::getB	156,520,234,780	106,760,160,140	1.466	0.159	0.7%	0.501

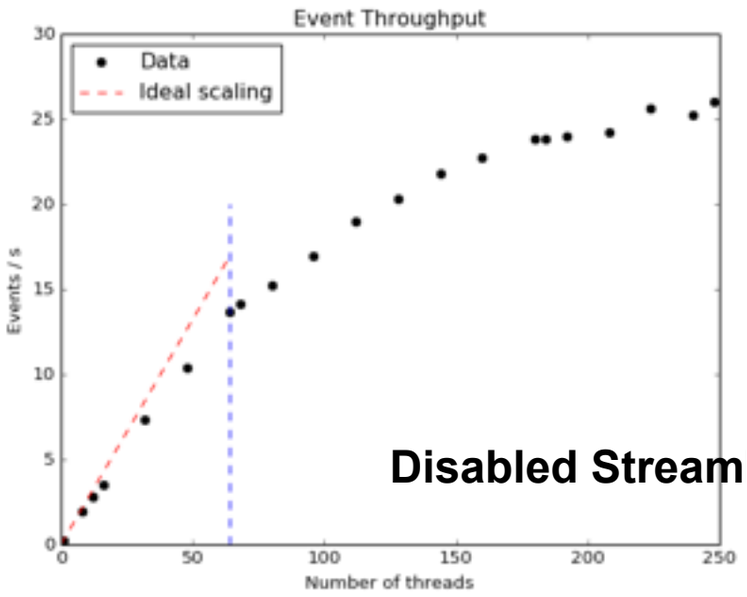
- The lists are fairly similar
  - The KNL slowdown doesn't seem to be due to any particular piece of code, but rather a global slowdown of the entire codebase

Single-mu scaling issue



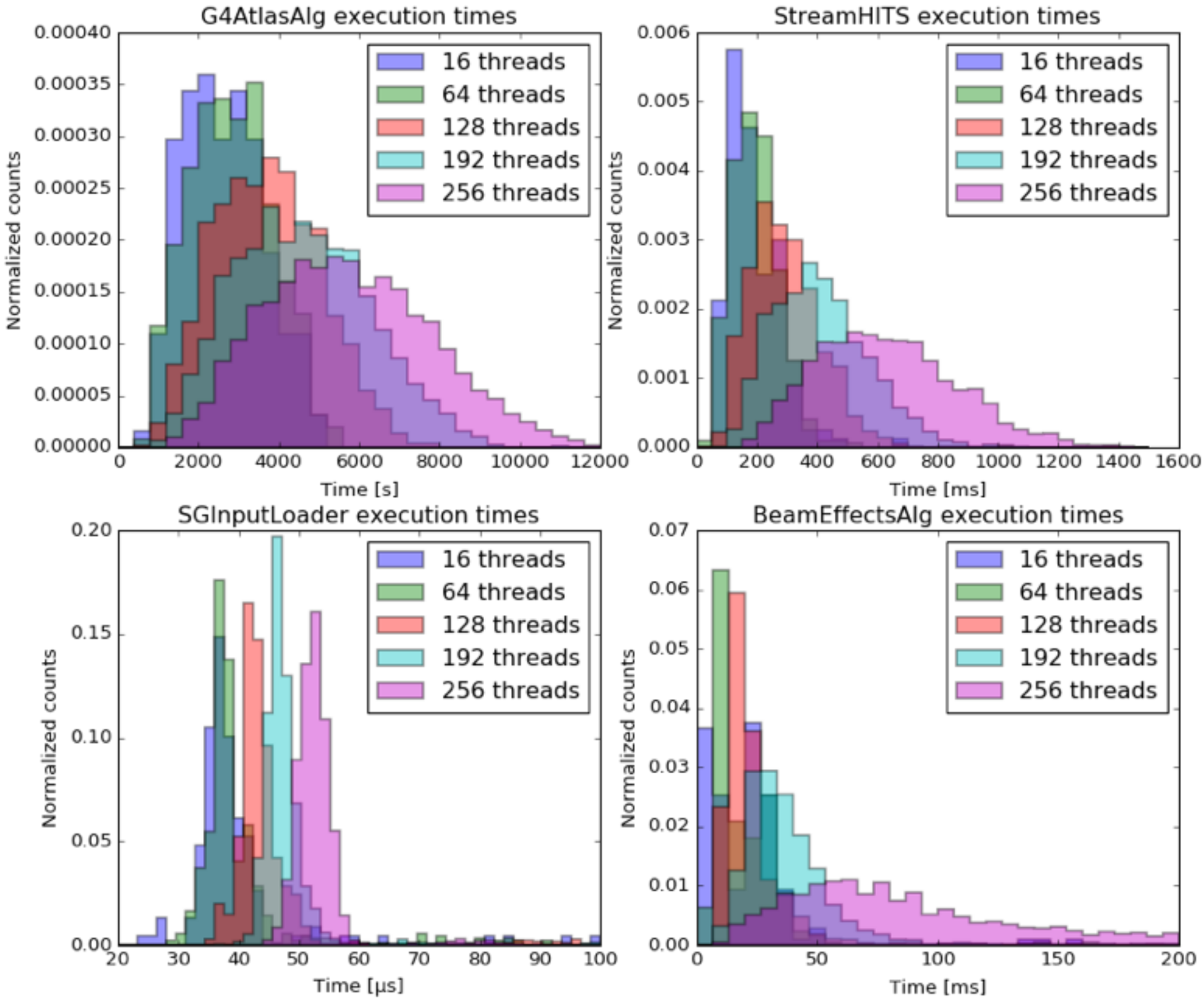


- Not accounted for in algorithmic time!
- The culprit?
  - Serialized I/O algorithms, particularly the StreamHITS
- Verified by disabling output stream

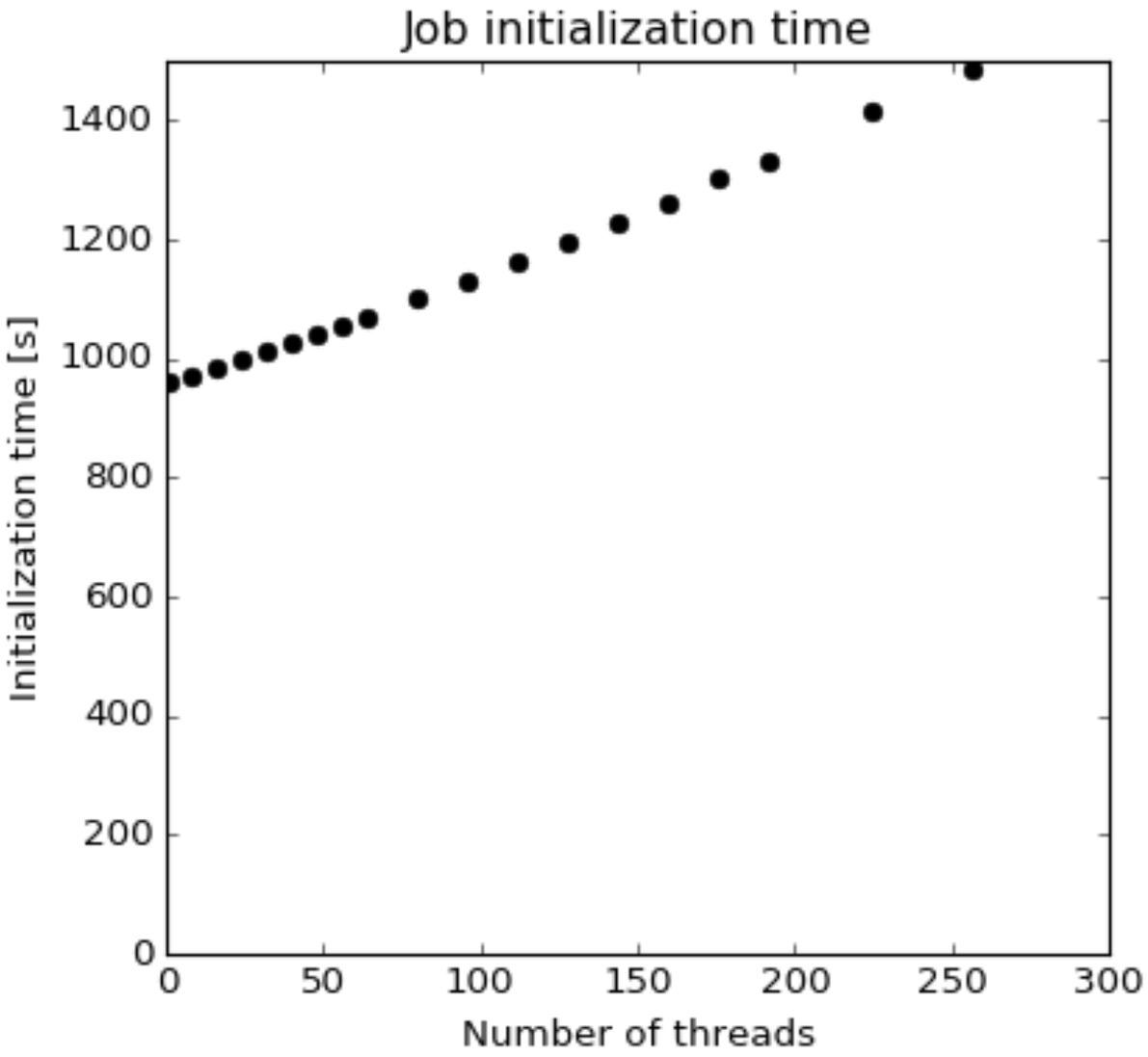


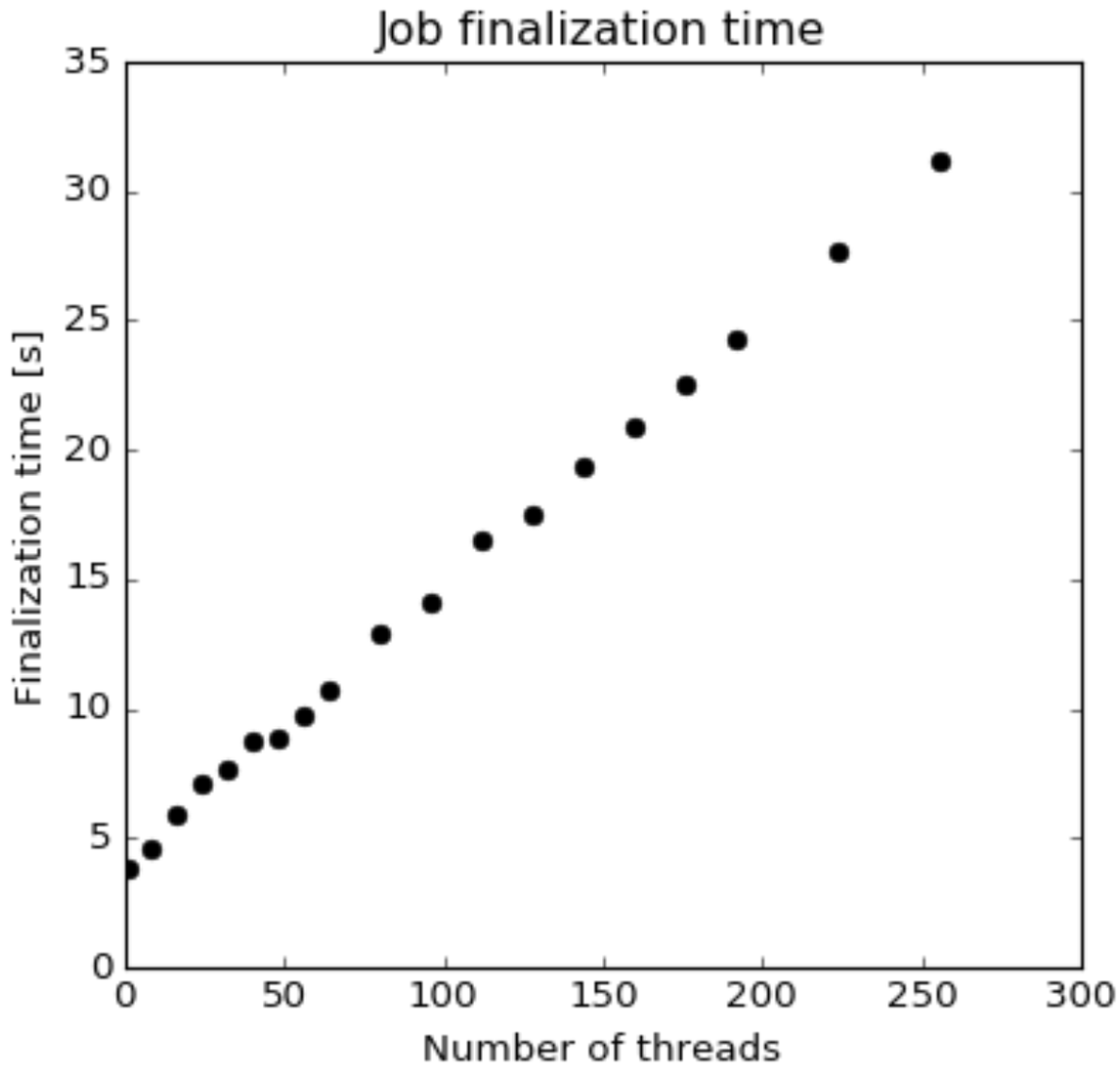
Disabled StreamHITS

Algorithm execution times on KNL (Ztautau)



Initialization and finalization times on KNL (Ztautau)





Memory consumption on KNL (Ztautau)