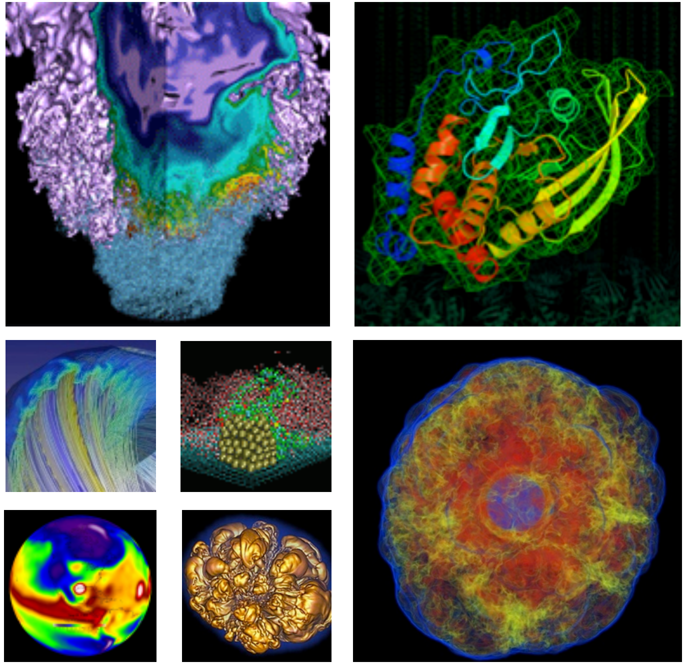
Understanding the IO Performance Gap Between Cori KNL and Haswell



Jialin Liu, Quincey Koziol, Houjun Tang, Francois Tessier, Wahid Bhimji, Brandon Cook, Brian Austin, Suren Byna, Bhupender Thakur, Glenn Lockwood, Jack Deslippe, Prabhat

NERSC

May 10 2017

Cray User Group Meeting, Seattle, 2017

U.S. DEPARTMENT OF ENERGY | Office of Science

BERKELEY LAB

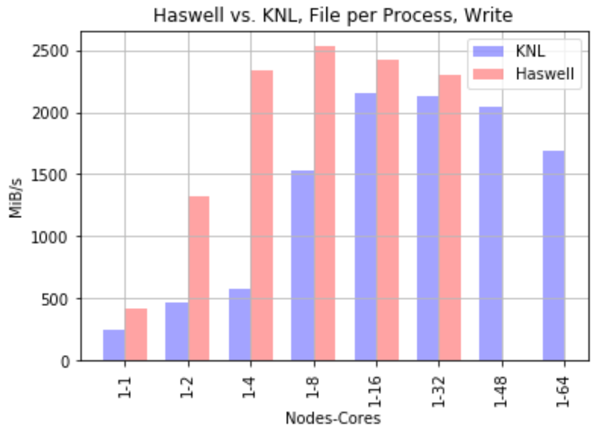
Background: New Architectures

- Cori Phase II was installed in 2016
  - Intel Xeon Phi 2nd generation
  - Knights Landing (KNL)

	KNL	Haswell
CPU	1.4GHz	2.3GHz
Memory	96 G DDR4, 16G HBM	128 G DDR4
Cache(L1, L2, L3)	64K, 1M	64K, 256K, 40M
Node	68 core, single socket	32 core, two socket
Capacity	9688 nodes	2388 nodes

Problem: Performance Gap

- Different Architectures, Performance Difference Detected
  - NESAP program at NERSC, focuses on computation performance [1]
  - How about **IO**? ~2X gap, **why**?



1.52X

IOR File Per Process

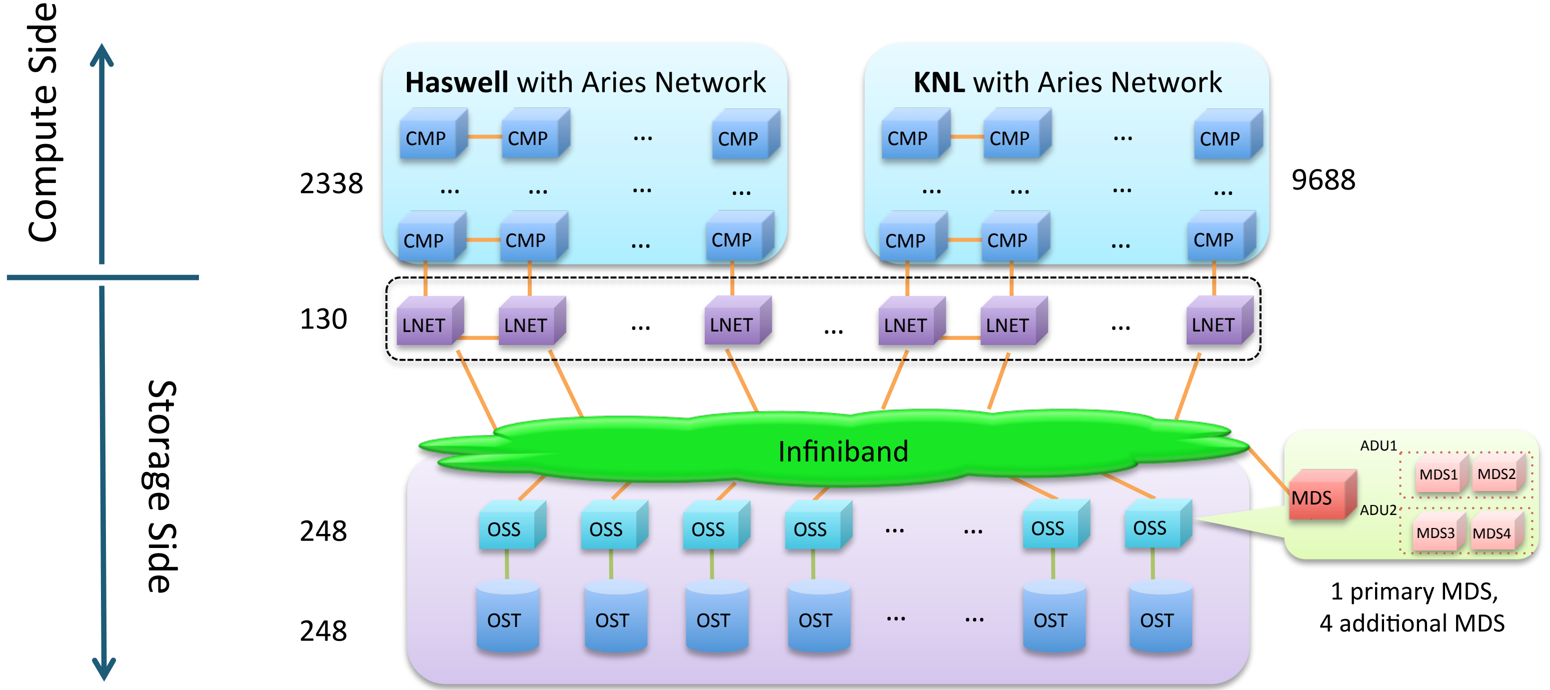
- 1G per process, 1 to 32 processes, **POSIX IO**

[1] T. Barnes, etc. Evaluating and Optimizing the NERSC Workload on Knights Landing. PMBS 2016

Mysterious IO Gap

- IO is typically slowed down by disk, not CPU
  - CPU is faster enough than disk, and has relatively smaller impact to the IO performance
- But IO stack underneath Haswell and KNL is same
  - Cray Sonexion 2000 Lustre appliance
  - Cray Aries with Dragonfly topology

IO Stack



Compute Side
Storage Side
Haswell with Aries Network
KNL with Aries Network
CMP
2338
9688
130
LNET
Infiniband
248
OSS
248
OST
MDS
ADU1
ADU2
MDS1
MDS2
MDS3
MDS4
1 primary MDS,
4 additional MDS

IO Benchmarks

- DD
- IOR
- HDF5

DD

- **dd**: Simple, commonly used in testing disk bandwidth
  - Copy a file, converting and formatting according to the operands.
  - *`dd if=input of=output`*
- **IO**:
  - *`dd_copy(){`*
  - *`posix_read();`*
  - *`posix_write();}`*
- **512** bytes by default
  - 1M block size is used in our test
  - 10000 count

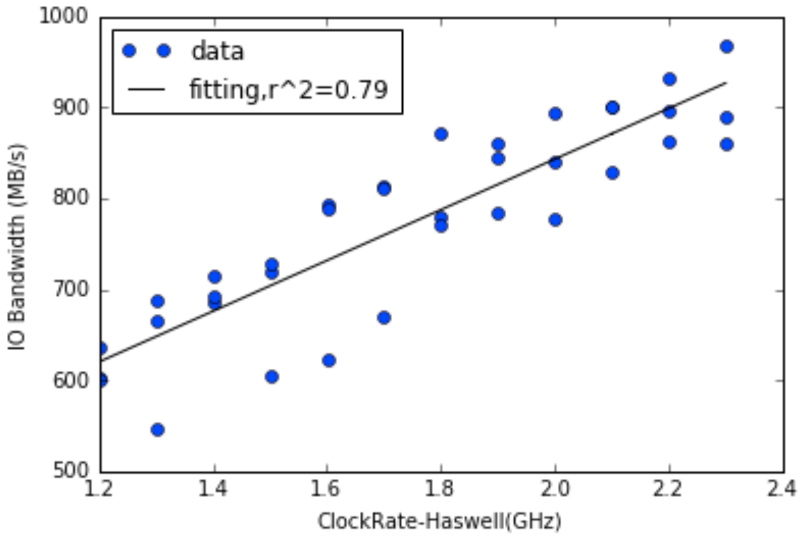
IOR

- **IOR**: HPC IO Benchmark for measuring peak performance
- **IO**:
  - File Per Process (FPP)
  - Single Shared File (SSF)
- Flexible Configurations:
  - Transfer/block/segment size
  - Fsync/dsync/direct IO
  - POSIX/MPIIO/HDF5

HDF5

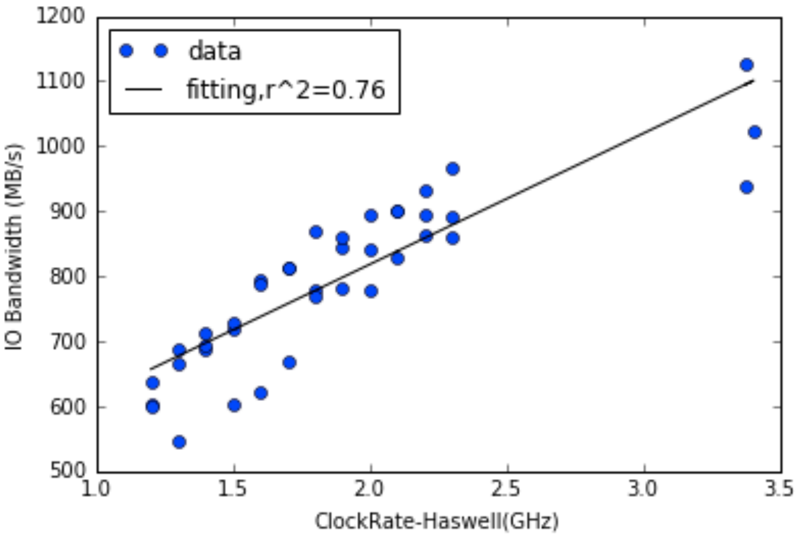
- **HDF5**: IO middleware used in many HPC applications
  - Construct logical access pattern
  - MPI Independent/Collective IO
- **MPIIO**:
  - *H5Dwrite() and H5Dread()*
  - *Collective buffer*
- Customized IO Benchmarks

Varying CPU Frequencies: Haswell



Without Turbo Mode

- $r^2 = 0.79$

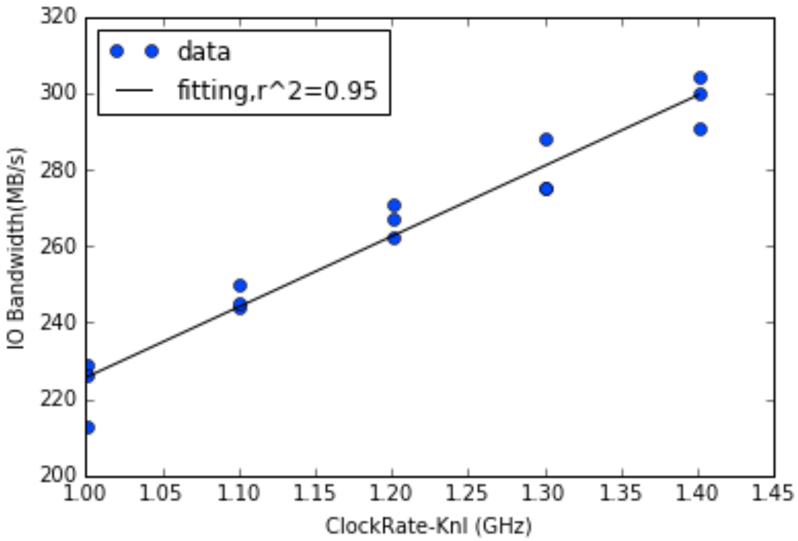


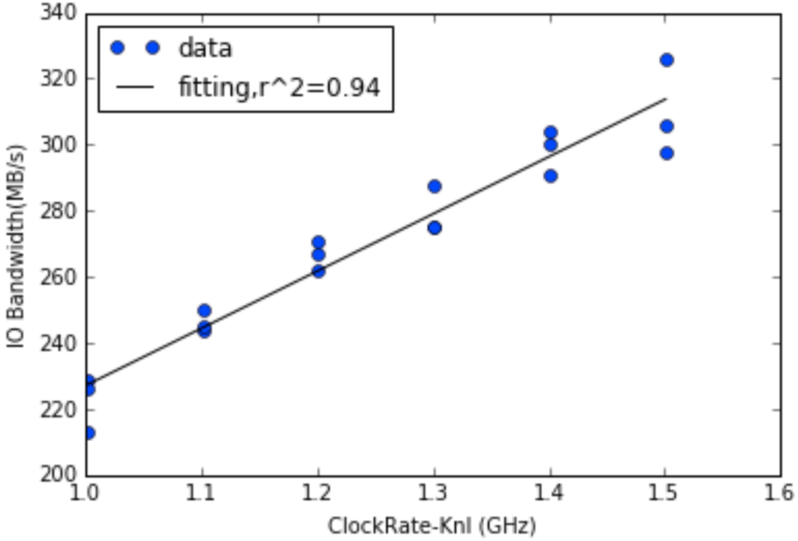
With Turbo Mode

- $r^2 = 0.76$

Haswell, Single Core, 10G write

Varying CPU Frequencies: KNL





Without Turbo Mode

- $r^2 = 0.95$

With Turbo Mode

- $r^2 = 0.94$

KNL, Single Core, 10G write

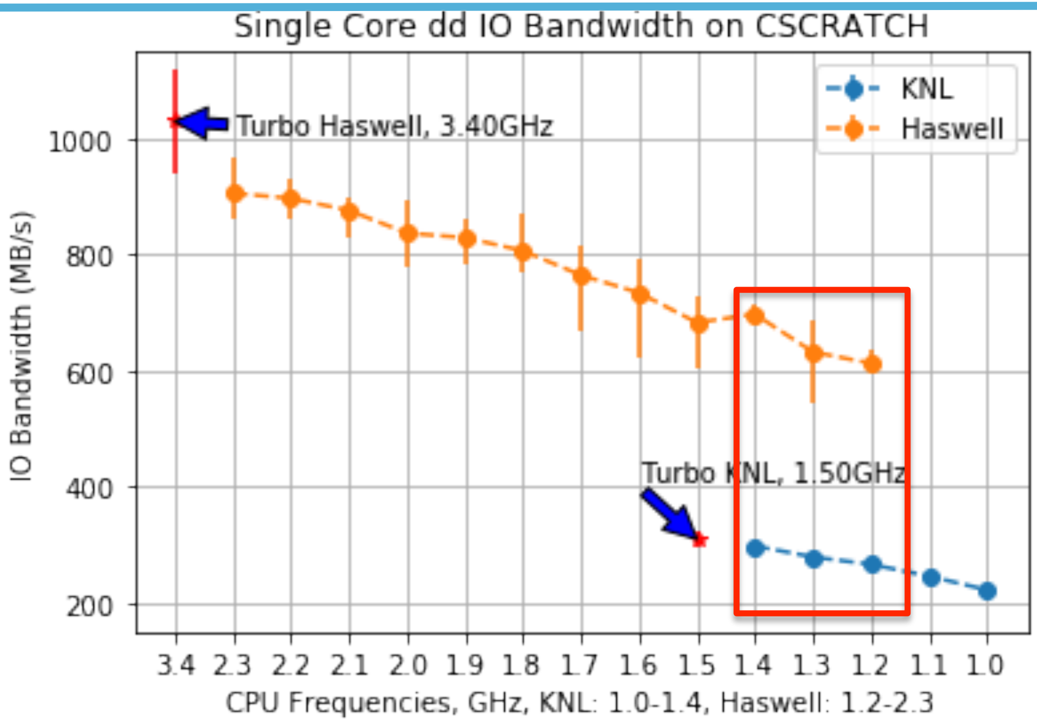
How Well is the Fitting

Partition	Haswell	KNL
r^2	0.79	0.95
intercept (MB/s)	286.11	41.28

- IO ~ CPU Frequency
- Single Core IO = f (CPU frequency, other), if IO fits in page buffer
  - **$r^2_{haswell} < r^2_{knl}$** : Complex Haswell chip; Wider range of CPU frequencies, more pipelined KNL chip
  - **intercept >> 0**: Page Cache.

Note that, the IO can fit in the page buffer well

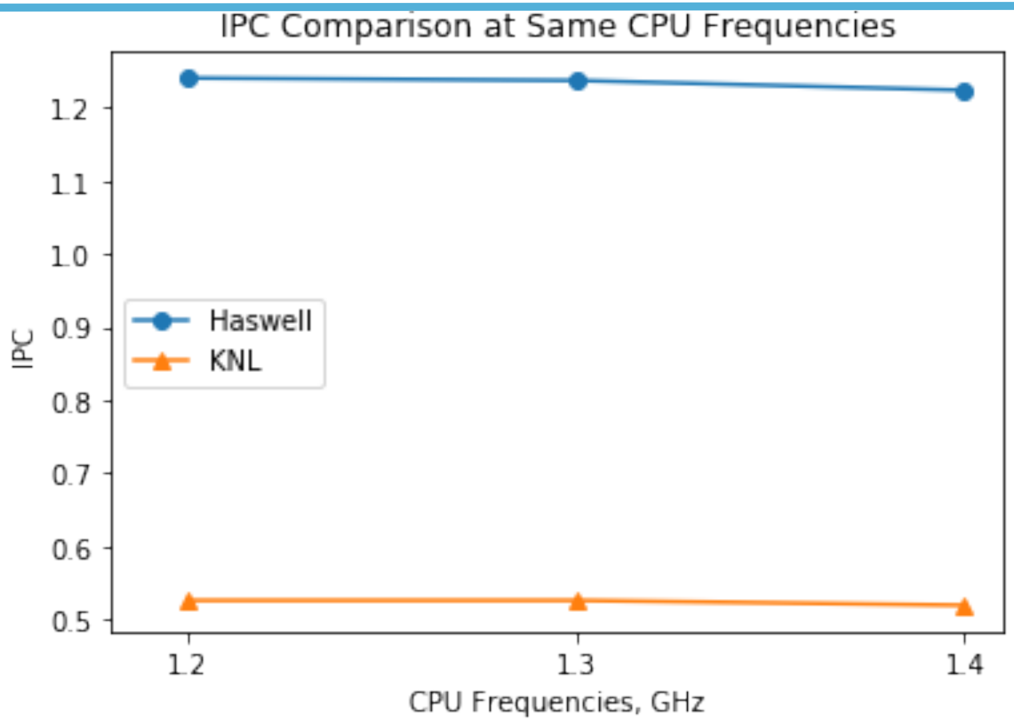
Haswell vs KNL



- Bandwidth Ratio Haswell / KNL = **2.30** (at same CPU freq)

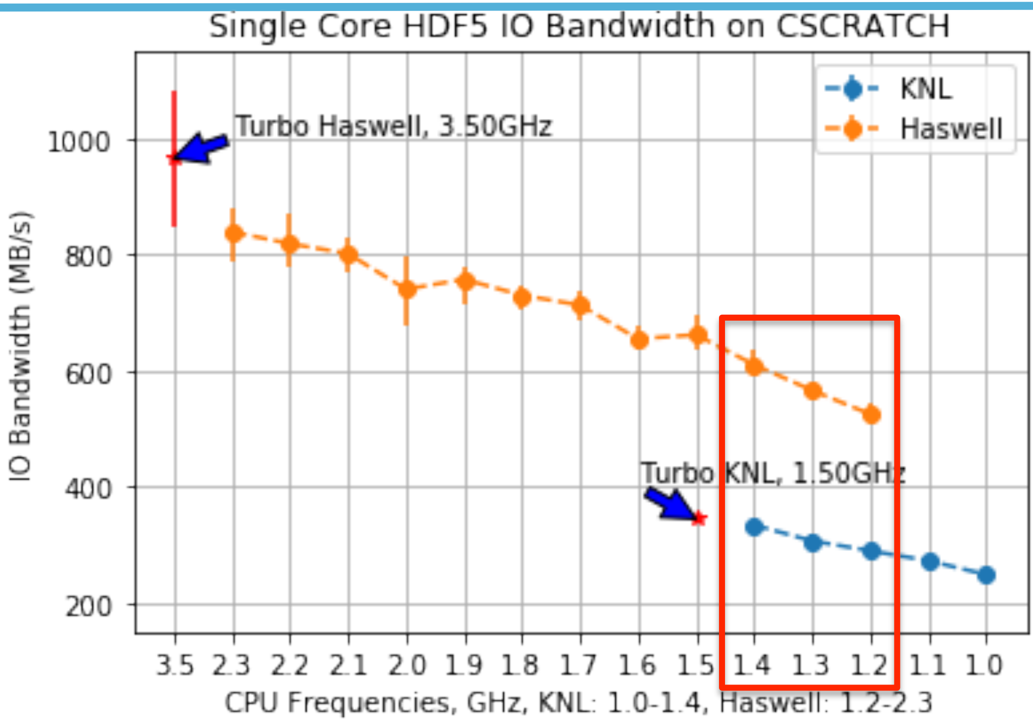
  = **3.46** (Turbo)

Other Facts



- $IPC_{haswell} / IPC_{knl}$ = **2.23** (average)
- $IPC_{haswell} / IPC_{knl}$ = **2.35** (at same cpu freq)

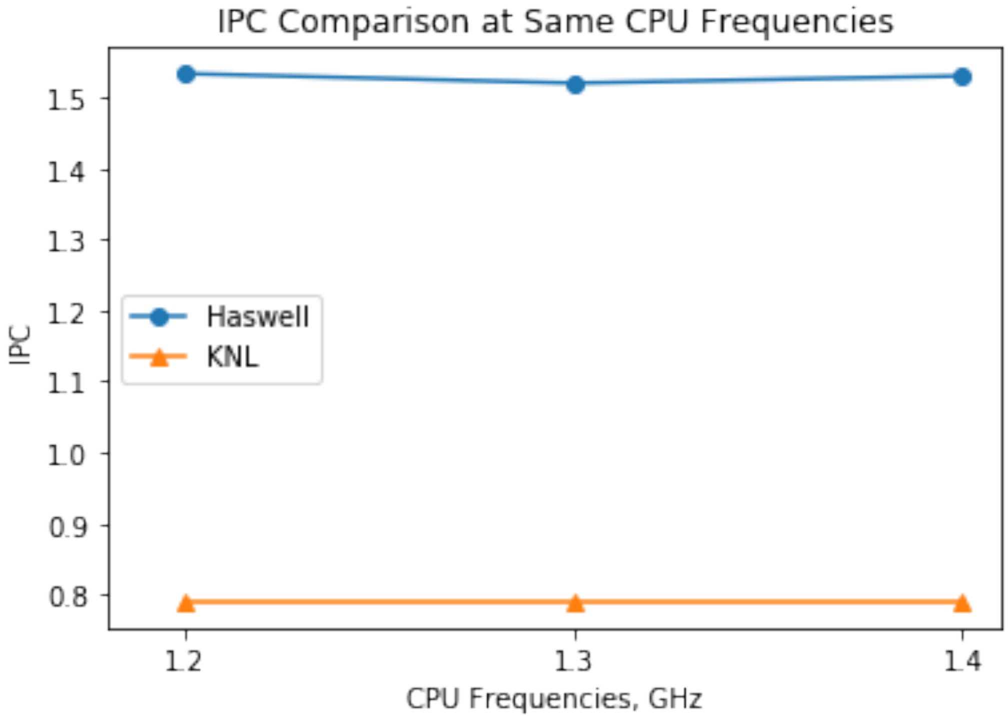
Similar Result with HDF5 Parallel IO



- $IO_{haswell} / IO_{knl}$ = **1.83** (at same CPU freq)

  = **3.06** (Turbo)

IPC Statistics



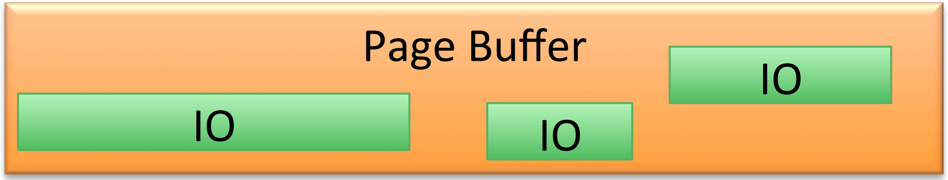
- $IPC_{haswell} / IPC_{knl}$ = **1.88** (average)
- $IPC_{haswell} / IPC_{knl}$ = **1.93** (at same cpu freq)

Same KNL IO Issue Confirmed at ANL

- dd write 10G to scratch
- IO Bandwidth: **262**MB/s, 1.398GHz (Turbo), AN
  306MB/s, NERSC

- CPU 7230 @ 1.30GHz 64 cores ANL
- CPU 7250 @ 1.40GHz 68 cores NERSC

Summary I for Single Core IO Performance

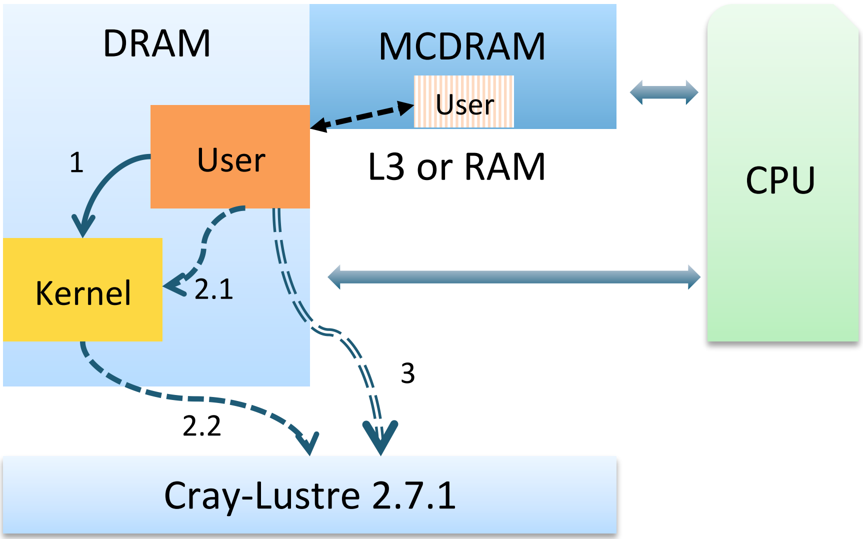


- **IO ~ F(CPU Frequency)**
  - Haswell: $r^2 = 0.79$ (DD), $r^2 =$ **0.89** (HDF5)
  - KNL: $r^2 = 0.95$ (DD), $r^2 =$ **0.96** (HDF5)
- KNL / Haswell
  - HDF5: IPC Ratio= 51%, IO BW Ratio = 55 %
  - DD: IPC Ratio=44%, IO BW Ratio = 43 %
- Turbo Mode (Default)
  - IO BW Ratio (KNL/Haswell) = **32%** (HDF5), **29%** (DD)

Node Local IO Path Deep Dive

- With the Same CPU Frequencies
  - What is the difference in the two node's IO path?

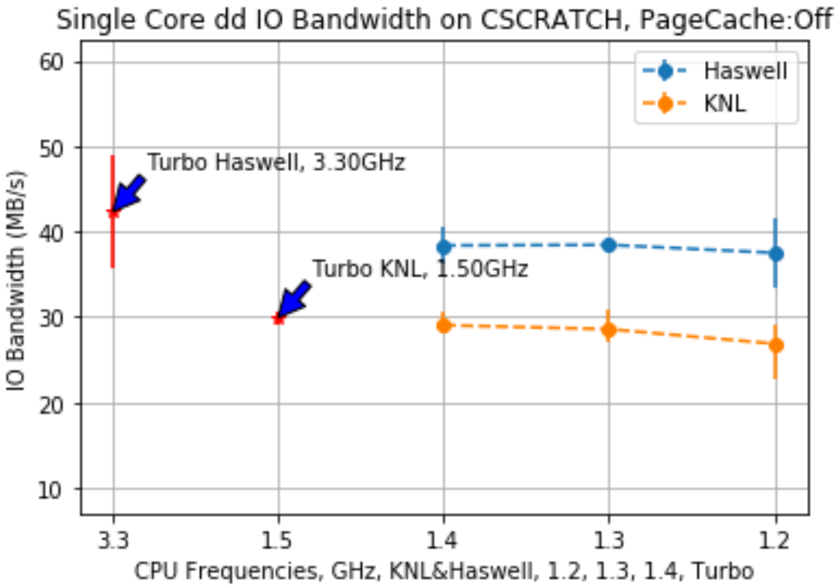
IO Path in Different IO Modes



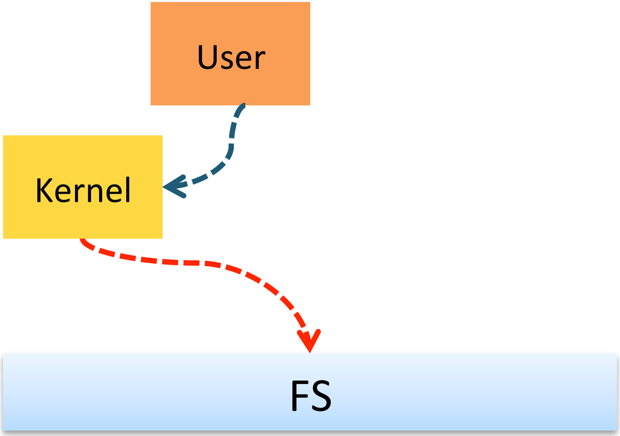
- 1 Buffer IO (Default)
  - User->Kernel, *memcpy()*
- 2 Sync IO
  - 2.1 User->Kernel, *memcpy()*
  - 2.2 Memory->Lustre, *buffer_io()*
- 3 Direct IO
  - User->Lustre, *direct_io()*

- MCDRAM in Cache Mode or Flat Mode
- IO is
  1. Buffer IO
  2. Sync IO
  3. Direct IO

Sync IO

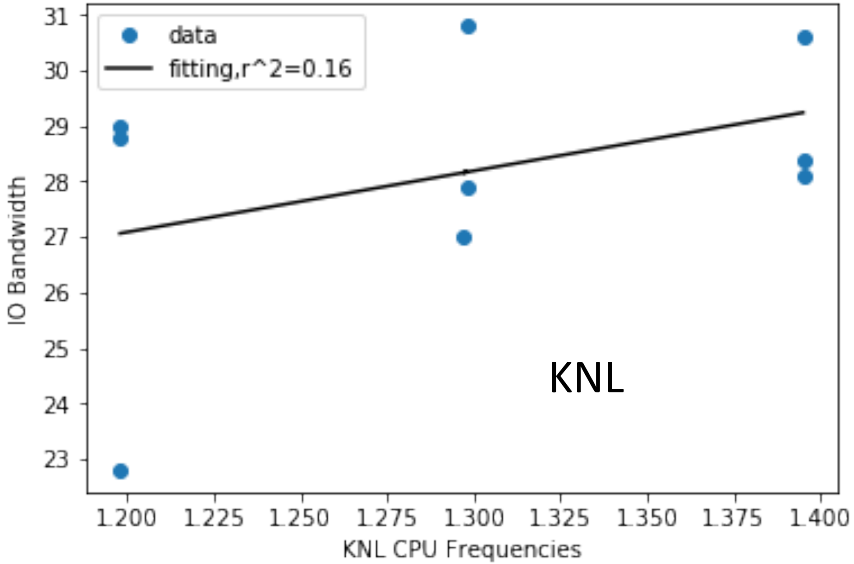


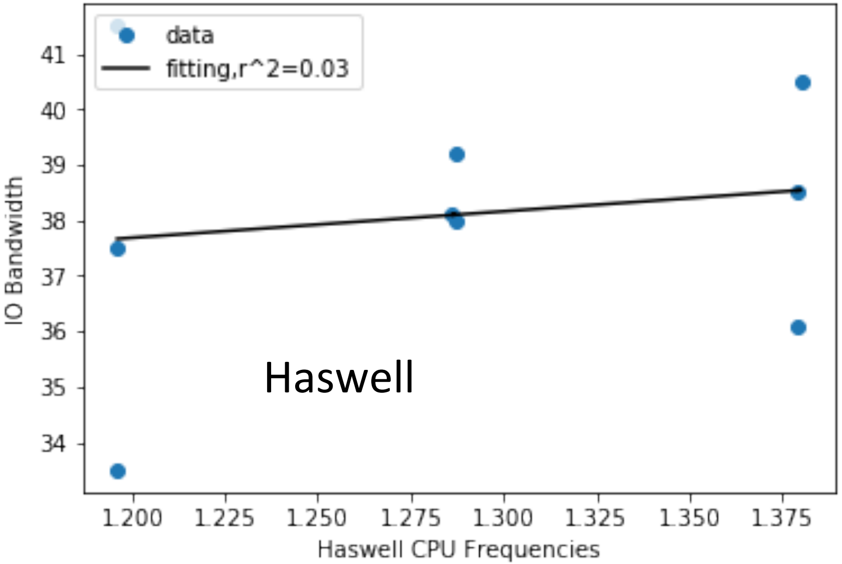
X is 1.4Ghz, 1.3 Ghz, 1.2Ghz, each is repeated 3 times



dd oflag=dsync if=/dev/zero of=$SCRATCH/1.txt bs=1m count=10000

Sync IO: CPU Impact diminishes





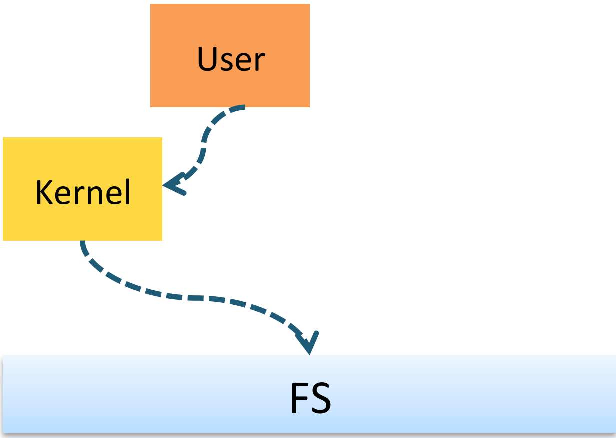
r-square

- 0.89-0.96 (Buffered IO)
- 0.03-0.16 (Sync IO)

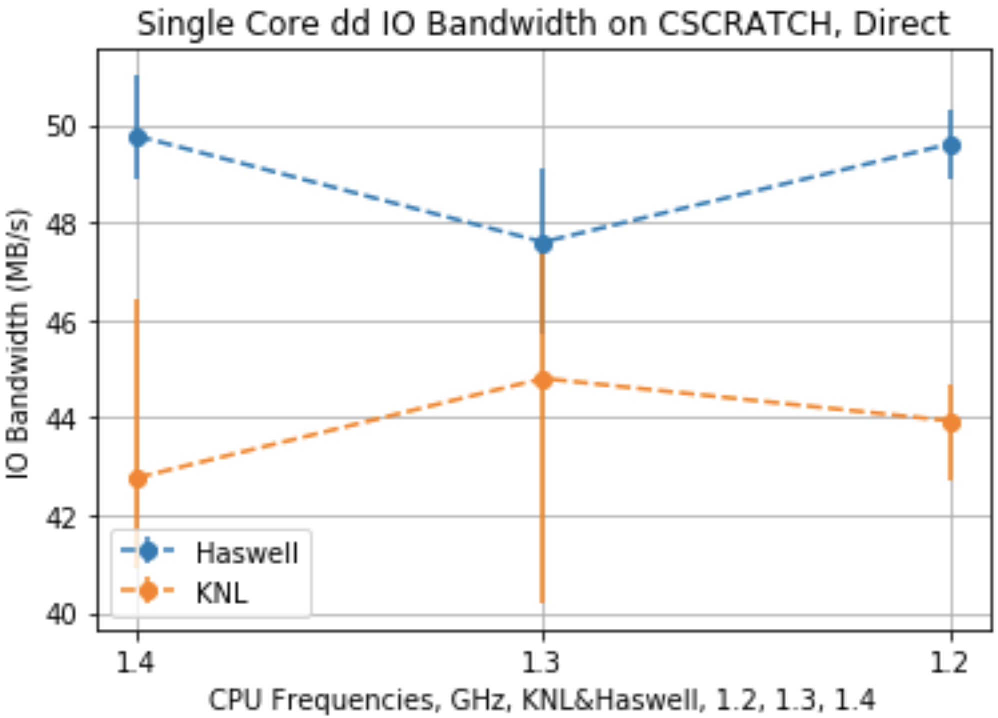
Sync IO: KNL is closer to Haswell

	Haswell	KNL-DRAM	KNL-MCDRAM
STDEV	0.55	0.46	2.22
AVERAGE	45.21	31.23	30.20
MEDIAN	45.35	31.47	31.13
KNL/Haswell		69%	67%

- *10G write with sync IO*
- *CPU 1.4, 1.3, 1.2 GHz*
- *Each test repeated 3 times*
- *Average*
- *User space memory is in DRAM or MCDRAM, set by numactl m=0 or 1*
- *Kernel space memory is unknown in case of MCDRAM as first priority memory, i.e., m=1*

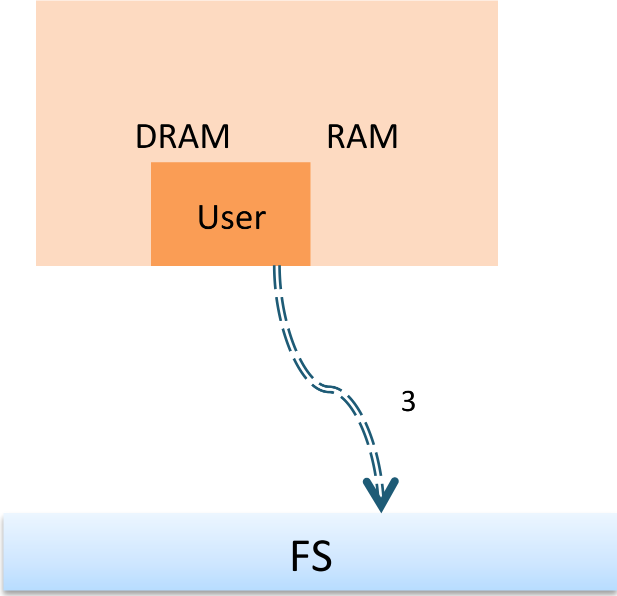


Page Cache Off, Direct IO



Page Cache Off, Direct IO

	Haswell	KNL-DRAM	KNL-MCDRAM
STDEV	1.21	1.02	1.15
AVERAGE	48.99	43.83	46.78
MEDIAN	49.60	43.93	46.40
KNL/Haswell		90%	96%

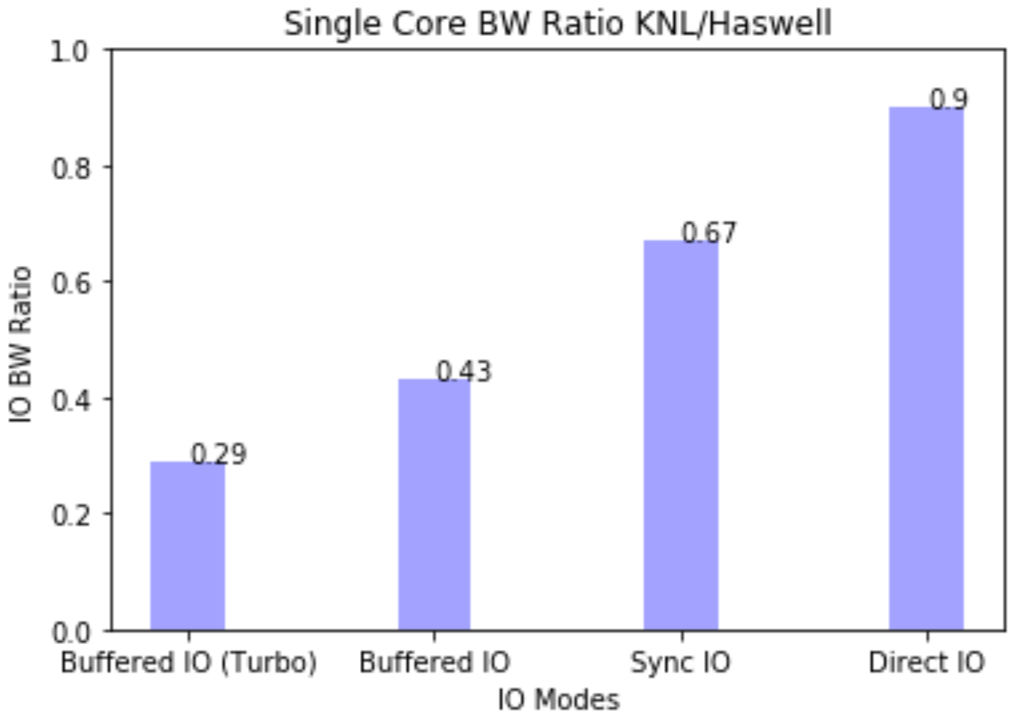


- *10G write with direct IO*
- *CPU 1.4, 1.3, 1.2 GHz*
- *Each test repeated 3 times*
- *Average*
- *Bypassing kernel space buffer*
- *User space memory is either using DRAM or MCDRAM, controlled by numactl m=0 for DRAM, 1 for MCDRAM*

Summary II for Single Core IO Performance

- IO ~ CPU Frequency: CPU Impact diminishes
  - Haswell r-square: 0.79 (page cache on) --->> 0.03 (page cache off)
  - KNL r-square: 0.95 (on) --->> 0.16 (off)
- IO BW at same CPU Frequencies
  - DRAM: Sync IO 67%, Direct IO 90% (KNL/Haswell)
  - MCDRAM: Sync IO 69%, Direct IO 96%
- Turbo Mode (Default)
  - DRAM: Sync IO 65%, Direct IO 78%
  - MCDRAM: Sync IO 73%, Direct IO 88%

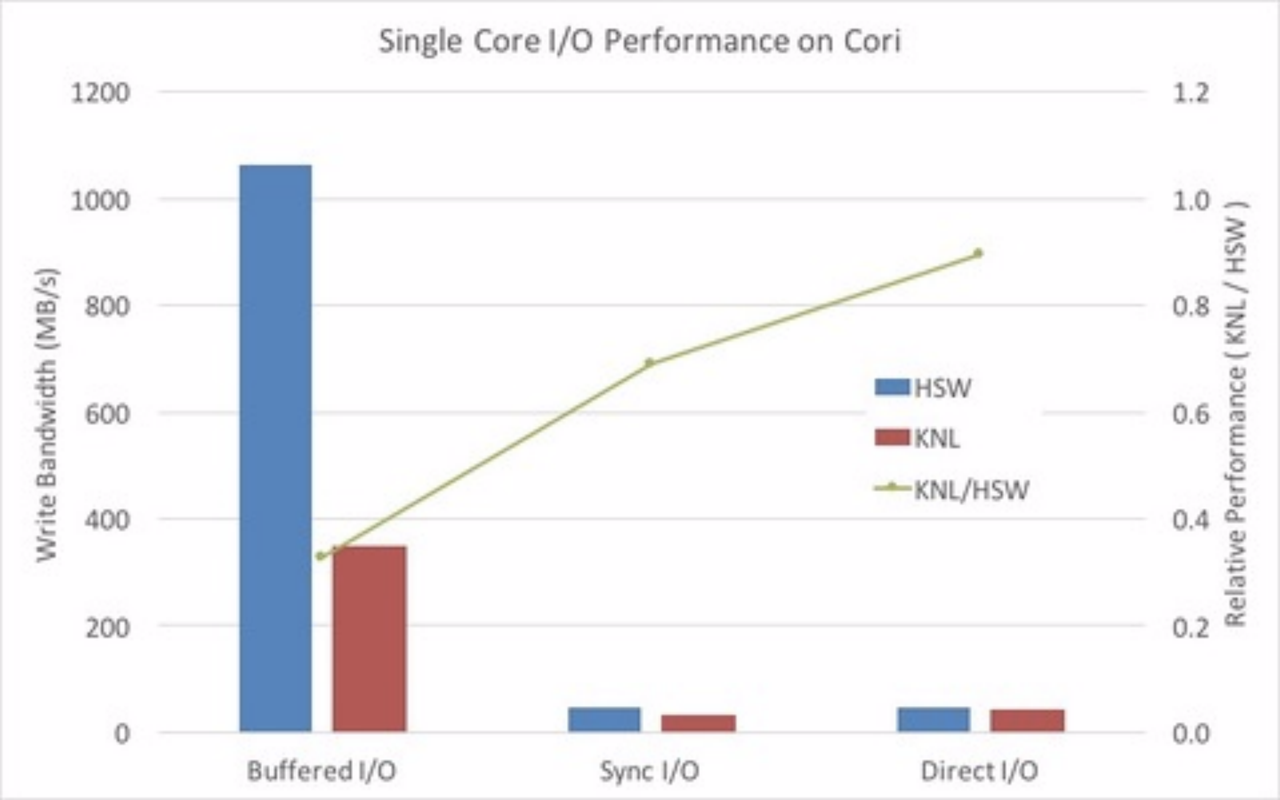
Summary II for Single Core IO Performance



Ratio=KNL/Haswell

Note that the absolute performance number is not revealed in this plot, **Buffered IO** typically deliver **10X performance speedup in write**

Summary II for Single Core IO Performance

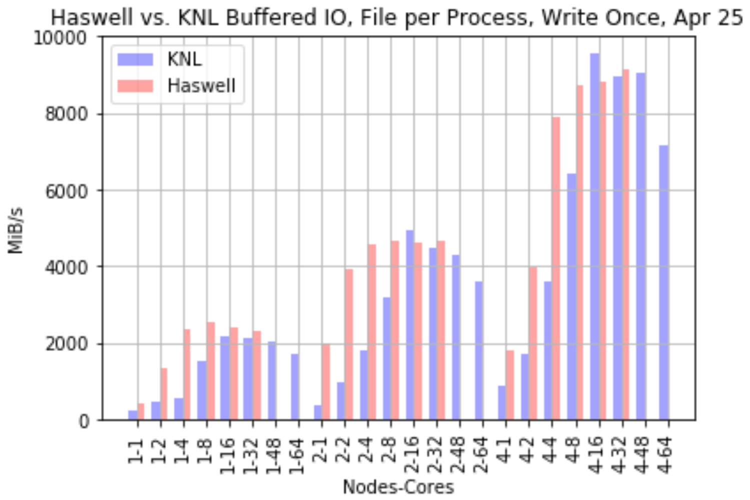


Parallel IO

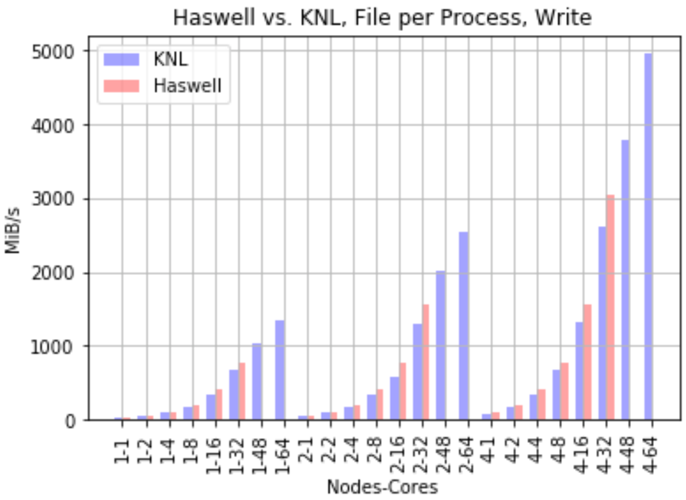
- Parallelism
  - More threads on KNL
  - Internal parallelism, Check Intel's new Lustre optimization LUG17
- Network, Inter-node Communication Latency
  - MPIIO
- Node Local Collective Buffer Size
  - Collective IO

Multiple Core, Multiple Node IO Tests
File per Process

Write, Same IO Mode, Haswell vs KNL



Buffered Write, Haswell vs KNL

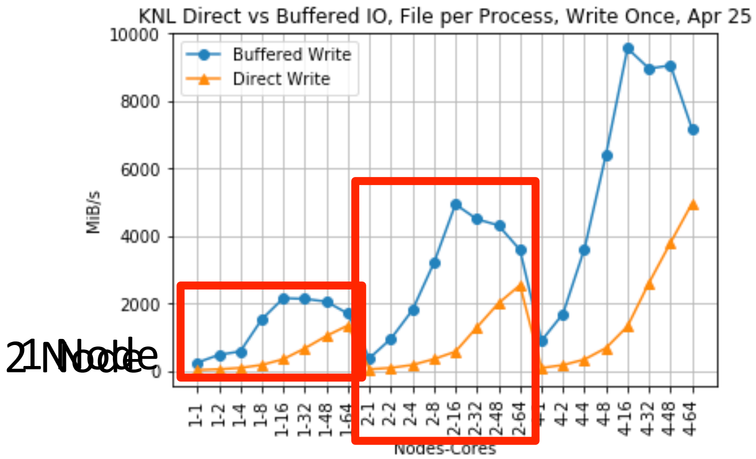


Direct Write, Haswell vs KNL

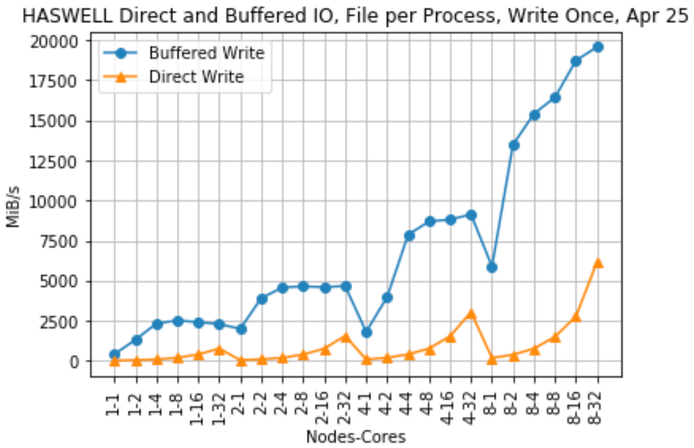
Same Number of Procs KNL/Haswell= 0.58 → 0.83 More Cores

Maximum KNL/Haswell= 0.99 → 2.25

Write: Same Node, Buffered vs Direct IO



Buffered vs. Direct Write, **KNL**



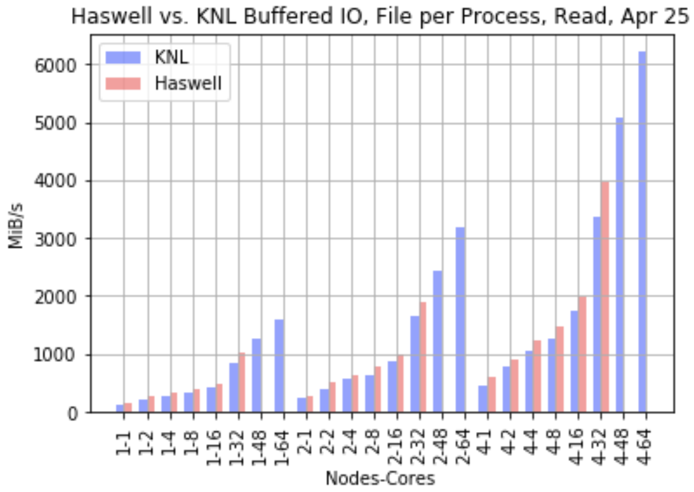
Buffered vs. Direct Write, **Haswell**

- *Direct IO is scalable*
- *KNL has less page buffer, and probably less powerful buffer management*

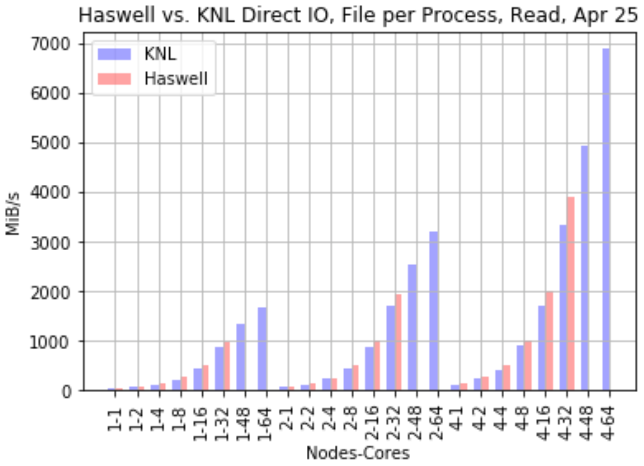
More Buffer

Read Once, Same IO Mode, Haswell vs KNL

Read 1 time



Buffered Read, Haswell vs. KNL

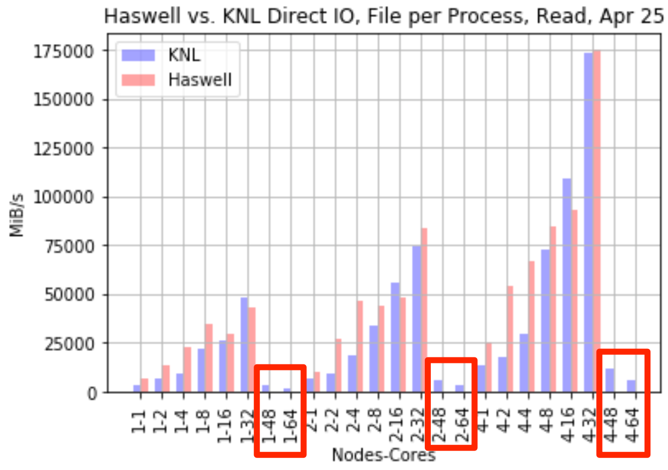


Direct Read, Haswell vs KNL

- KNL IO BW **outperforms** Haswell with **more cores** in both buffered & direct IO

Read Multiple Times

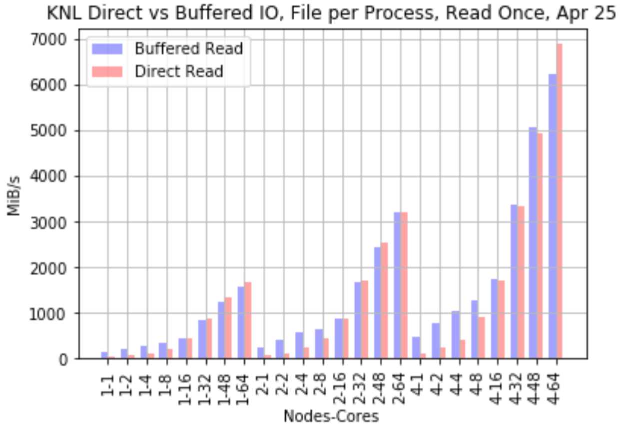
Read 3 times, Don't flush the cache explicitly



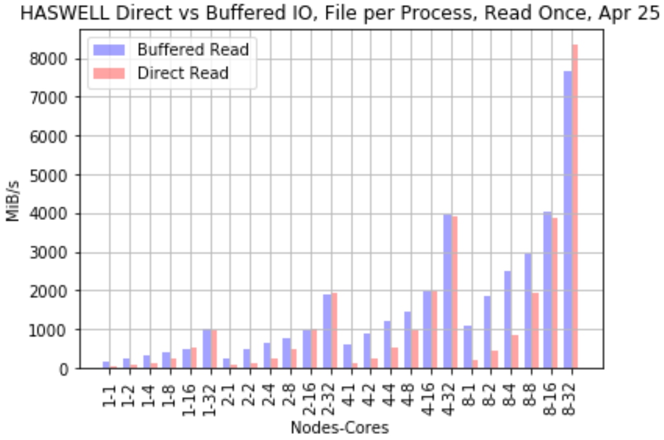
Buffered Read, Haswell vs. KNL

- KNL IO BW drops at 48-64 cores per node
  - Increase page buffer, not tried yet

Read Once, Same Node, Buffered vs Direct IO



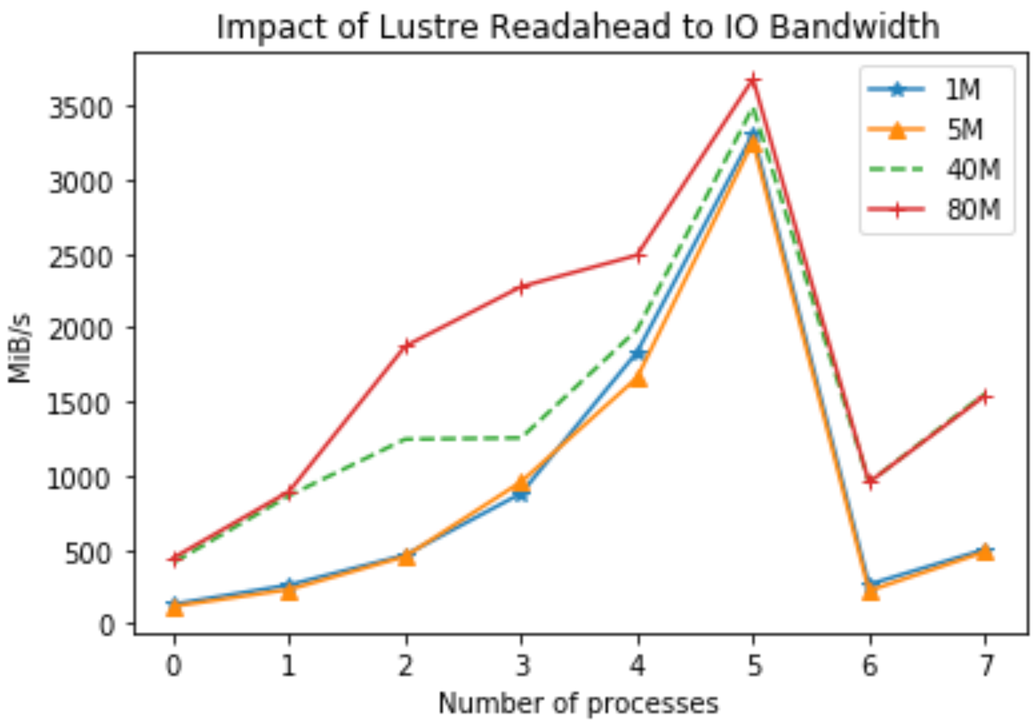
Buffered Read vs. Direct Read, KNL



Buffered vs Direct Read, Haswell

- Direct Read reaches and outperforms Buffered Read
  - Lustre readahead benefit reduces as memcopy cost increases

Lustre Read-ahead to Read Performance



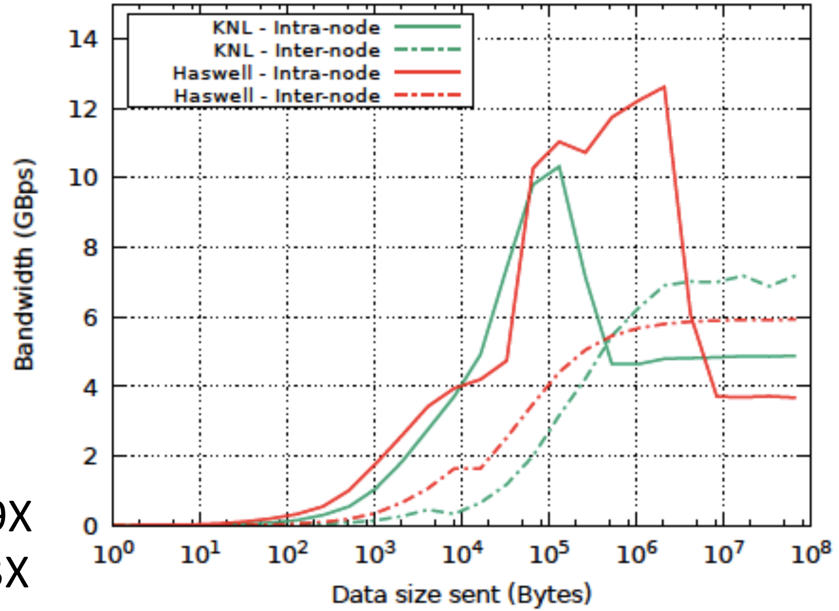
Summary III Multi-Node/Core File Per Process

- Write
  - KNL/Haswell 0.58 -> 0.99 (32 processes to 64 processes)
  - Direct IO: Scalable, can reach Buffered IO
  - *More page buffer for better buffered IO performance*
- Read
  - KNL outperforms Haswell with more cores in both buffered/ direct IO, with read once IO pattern
  - KNL drops due to page buffer limit when read multiple times
  - Lustre read-ahead is a factor
  - Direct IO outperforms buffered IO with large one-time read

Multiple Core, Multiple Node IO Tests
Single Shared File

LATENCY IN μs BASED ON THE SENDING OF A 0 B MESSAGE

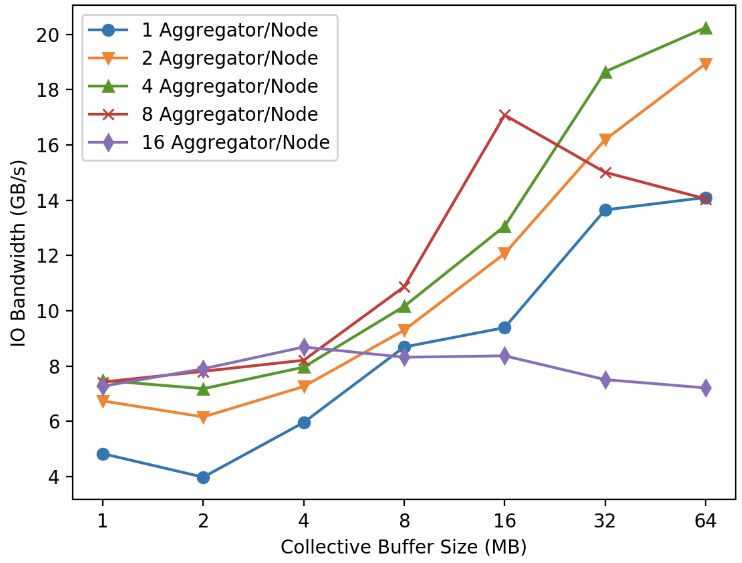
	Haswell	KNL
Inter-node	1.97	4.92
Intra-node	0.23	0.63



- With 0 Byte, Haswell/KNL Inter-node BW 2.49X
  Intra-node BW 2.73X
- KNL outperforms Haswell with larger message size in inter-node comm
- Larger buffer size

LATENCY IN μs BASED ON THE SENDING OF A 0 B MESSAGE

	Haswell	KNL
Inter-node	1.97	4.92
Intra-node	0.23	0.63



- With 0 Byte, Haswell/KNL Inter-node BW 2.49X
  Intra-node BW 2.73X
- KNL outperforms Haswell with larger message size in inter-node comm
- Larger buffer size

Conclusion

- CPU Frequency
  - Main factor
  - IO scales with CPU when IO can fit into page buffer
- Page Buffer
  - KNL is close to Haswell with direct IO
  - Page buffer management is slower on KNL
  - Page buffer benefits generally, e.g., write, multi-read
  - Direct IO can be better than buffered IO with large one-time read
- Many Cores
  - KNL could outperform Haswell with more cores in FPP read once.
  - Direct IO is much more scalable than buffered IO
- Network, Collective Buffer and Others
  - KNL has larger inter-node latency than Haswell
  - Increasing buffer size in MPIIO can improve IO BW

Future Work

- Page Buffer Management on KNL
  - MCDRAM as page buffer
- Cross-partition IO
  - Offload IO from KNL to Haswell
  - Shift computation from Haswell to KNL
  - Dynamic Datahub: https://github.com/NERSC/heterogeneous-IO
- Many/Heterogeneous Core IO Optimization