

NEXTGenIO: Next Generation I/O for HPC

Michèle Weiland

m.weiland@epcc.ed.ac.uk

@micheleweiland





Optane DCPMM

- Intel Optane DC Persistent Memory, based on 3D XPoint device technology
 - Hosted in the DIMM slots, additional memory controller
- Byte addressable connected to the memory bus like normal DRAM
 - Cache Coherent, Load /Store Accesses
 - Ability to do DMA & RDMA
- 128/256/512GB per module, 2 socket system up to 6TB
- High Endurance with 5 years lifetime assuming maximum write bandwidth
 - ~500 Petabyte written per module (flash devices ~500-700 Terabyte)
- Storage I/O operations become fast persistent memory operations
 - No Paging for storage I/O
 - No Context Switching for storage I/O
 - No Interrupts for storage I/O
 - No Kernel Code Running for storage I/O









- System built by Fujitsu using bespoke motherboard
- 34 compute nodes
- Node configuration
 - Dual socket, 2 x 24-core Intel Xeon Platinum8260M CPUs
 - 192GB DDR4 DRAM (12 x 16GB)
 - 3TB DCPMM (12 x 256GB)
- Omni-Path interconnect
 - Dual rail
- 270TB Lustre file system
- Total memory capacity
 - 6.5TB DRAM
 - 100TB NVRAM





Basic throughput & latency

DDR4 DRAM

- ~70ns idle latency
- ~18.7GB/s read
- ~8.9GB/s write

DCPMM NVRAM

- ~176ns idle latency (2.5x)
- ~8.0GB/s read (2.4x)
- ~1.5GB/s write (6x)

- Values are *per device*
- Bandwidth between sockets (UPI) ~83,2 GB/s bidirectional

The measured idle latencies (in ns) reported by Intel Memory Latency Checker





Platform modes

2LM

- NVRAM is transparent and **not** persistent
- All DRAM is used as L4 cache for the memory mode space
- Use application as normal, no changes required

1LM

- NVRAM needs to be addressed directly
- Persistent
- Various namespaces options
 - e.g. fsdax, devdax

Memory mode:

Size of all memory mode space in all DCPMMs == the size of the OS main memory \rightarrow can only be used in 2LM platform mode

App Direct mode:

App Direct space can be accessed via memory mapped operations \rightarrow can be used in both 1LM and 2LM platform mode





System setup

- DCPMM devices are socket local
- Memory space has 192GB Level 4 cache
 - OS main memory is 3TB per node
- App Direct space configured as fsdax (ext4) namespace
 - fsdax == filesystem dax
 - /mnt/pmem_fsdax{0,1}





STREAM benchmark

https://github.com/NGIOproject/benchmarks

- Synthetic application designed to measure the achievable memory bandwidth
- Standard STREAM for DRAM and Memory mode
- Modified STREAM for App Direct
 - Initial memory allocation replaced with a call to pmem_map_file & memory offset calculations
 - Add "data persist" instructions to ensure data is fully stored on the DCPMM
- Configuration
 - DRAM and App Direct: array size 19MB per process (2.7GB total)
 - Memory mode: array sizes 19MB and 4GB per process (768GB total)



STREAM Triad

- Theoretical peak (DRAM only) is 210GB/s
- Configuration: 48 MPI processes per node, 10 nodes
- App Direct DCPMM bandwidth is ~5x less than DRAM
- Memory mode bandwidth for small size is roughly same as DRAM
 - This does not fulfil the "array size should be at least 4x the size of the sum of all the last-level caches used in the run" rule

Mode	Min BW (GB/s)	Med BW (GB/s)	Max BW (GB/s)
App Direct (DRAM only)	174	180	183
App Direct (DCPMM only)	47	48	49
Memory mode (19MB array size)	144	146	147
Memory mode (4GB array size)	27	28	28



Memory intensive application





CASTEP

- Materials modelling application
- Needs scale to satisfy memory demands
- BUT poor scaling behaviour due to high volume of all-to-all communications
 - Poor parallel efficiency CPUs underutilised

Number	MPI procs	Average mem	Total mem
of nodes		per node (GBs)	(TBs)
20	720 (36 MPI per node)	103.21	2.01





CASTEP baseline, no DCPMM

• DNA test case

- Application's own estimates of memory requirements and parallel efficiency
- Measurement of wallclock time for 3 SCF loop iterations

Nodes	MPI x OpenMP	Active cores	Memory estimate per process	Runtime	Overall parallel efficiency
34	48x1	1632	5,486.6 MB	OOM	N/A
24	36x1	864	6,476.0 MB	3,959.01s	23%
24	24x2	1152	7,491.2 MB	4,598.74s	28%
20	36x1	720	7,034.9 MB	3,311.88s	33%
20	24x2	960	8,320.0 MB	4,377.15s	36%
18	36x1	648	7,355.5 MB	3,587.94s	34%
18	24x2	864	8,843.5 MB	4,814.94s	36%





More memory capacity

- DCPMM can provide more memory per core in two ways
 - 2LM Memory mode transparent
 - 1LM App Direct mode using *libvmmalloc* (PMDK library)
- *libvmmalloc* intercepts calls to dynamic memory allocations (e.g. malloc, memalign or free) and replaces them with persistent memory allocations
 - 1. Set the size of the non-volatile memory pool (VMMALLOC_POOL_SIZE) and the location of the pool (VMMALLOC_POOL_DIR)
 - 2. The path to the non-volatile memory pool points to a directory that is created in /mnt/pmem_fsdax{0,1}
 - 3. Preload *libvmmalloc* using LD_PRELOAD and launch the application as normal



Tracing with a smaller test case

- TiN test case fits on single node, memory high watermark ~19GB
- 4 scenarios (top to bottom)
 - 1. DRAM only
 - 2. Memory mode



4. App Direct + *libvmmalloc*, compute on socket 0 and DCPMM on socket 1







Tracing with a smaller test case (2)



Mode	Runtime (s)	Slowdown
App Direct (DRAM only)	442.30	-
Memory mode	484.90	1.10x
App Direct + libvmmalloc local	468.89	1.06x
App Direct + libvmmalloc remote	599.70	1.36x

Mode	CPI	Stall cycles	Load instructions from DCPMM
App Direct (DRAM only)	1.17	1.62E+13	0
Memory mode	1.25	1.87E+13	3.81E+08
App Direct + libvmmalloc local	1.25	1.84E+13	6.87E+08
App Direct + libvmmalloc remote	1.57	2.45E+13	5.66E+08



DNA test case

Number of nodes	MPI procs	Average mem per node (GBs)	Total mem (TBs)
1	48 (48 MPI per node)	1,735.40	1.69
2	96 (48 MPI per node)	882.16	1.72
4	192 (48 MPI per node)	452.10	1.77
20	720 (36 MPI per node)	103.21	2.01

DNA test case now fits on a single node

Mode	Nodes	MPI x OpenMP (per node)	Hyperthreading (Y/N)	Runtime (seconds)
App Direct (DRAM only)	20	36x1	N	3,311.92
App Direct (DRAM only)	20	24x2	Ν	4,377.20
App Direct with libvmmalloc	20	36x1	Ν	3,491.59
App Direct with libvmmalloc	20	24x2	Ν	4,523.66
Memory mode	4	36x1	Ν	13,773.95
Memory mode	4	24x2	Ν	16752.95
Memory mode	4	48x1	Ν	14,127.05

20% throughput improvement





I/O intensive application





OpenFOAM

- v1812 built with Intel 19.0.3.199 compilers & MPI library
- Test case: open wheel race car geometry 90 million cells





Performance of default setup



I/O characteristics for this case



Using fsdax directly

- Taking advantage of OpenFOAM's I/O strategy use fsdax namespace
 - Write once, read never
 - Write locally
- Copy test case data to DCPMM
 - Use fsdax on both sockets!
- Exploit MPMD for good data locality

<pre>cd /mnt/pmem_fsdax0</pre>			
time srun –n 4 –N 4	cp -fr	/home/nx01/nx01/mweiland/caseDir	
<pre>cd /mnt/pmem_fsdax1</pre>			
time srun –n 4 –N 4	cp -fr	/home/nx01/nx01/mweiland/caseDir	-

time mpirun -ppn 48 -n 24 simpleFoam -case /mnt/pmem_fsdax0/caseDir -parallel : \
-n 24 simpleFoam -case /mnt/pmem_fsdax1/caseDir -parallel : \
-n 24 simpleFoam -case /mnt/pmem_fsdax1/caseDir -parallel : \
-n 24 simpleFoam -case /mnt/pmem_fsdax1/caseDir -parallel : \
-n 24 simpleFoam -case /mnt/pmem_fsdax0/caseDir -parallel : \
-n 24 simpleFoam -case /mnt/pmem_fsdax1/caseDir -parallel : \
-n 24 simpleFoam -case /mnt/pmem_fsdax1/c

Total number of files created: Write interval 1: 192 * 100 * 8 = 153,600

4 node experiments

Total number of directories created: N * T * 3 Write interval 1: 192 * 100 * 3 = 57,600





Total number of files created: Write interval 1: 960 * 100 * 8 = 768,000

20 node experiment

Total number of directories created: N * T * 3Write interval 1:960 * 100 * 3 = 256,000



Performance stability



Achieved I/O performance

- Difference in time per timestep between "no IO" and "write interval X" can be entirely attributed to data writes and associated metadata operations
- On 20 nodes for "write interval 1":
 - Lustre 29.77s 1.47s = 28.28s 7.5GB / 28.28s = 0.256GB/s
 - fsdax 2.23s 1.47s = 0.76s 7.5GB / 0.76s = 9.87GB/s

Ensemble experiments

- Nobody runs on empty systems
- Single isolated experiments are also rare
- For a more realistic view, we compare performance of five 4-node experiments running concurrently
 - Using a total of 20 nodes, but independent simulations
 - More representative of real, busy environments
 - Plus: same number of metadata operations, but 5 times the amount of data written
 - 7.5GB per timestep *per simulation*

Ensemble performance



Time per timestep: single run vs ensemble

	Lustre		fsc	lax	
	Single run (4 nodes)	Ensemble average (5 x 4 nodes)	Single run (4 nodes)	Ensemble average (5 x 4 nodes)	
write interval 100	7.680	7.674	7.700	7.700	
write interval 10	7.890	7.816	7.870	7.844	
write interval 1	15.060	58.97	9.500	9.322	
	f 7.5GB per ste	p			
Lustre 7.5GB / (15.06-7.0 37.5GB / (58.97-7	68) = 1.01GB/s 7.67) = 0.73GB/s	37.	5GB per step		
fsdax 7.5GB / (9.5-7.7) = 37.5GB/ (9.3-7.7)	sdax 7.5GB / (9.5-7.7) = 4.17GB/s 37.5GB/ (9.3-7.7) = 23.12GB/s				

Conclusions

- DCPMM is a new memory technology that offers unprecedented opportunities both as memory & storage device
 - Through byte-addressability, high capacity, low latency (and persistence)
- CASTEP in Memory mode \rightarrow increased throughput
- OpenFOAM in App Direct mode \rightarrow performance hit of I/O negated
- Many avenues for research still left to be explored



Michèle Weiland, Holger Brunst, Tiago Quintino, Nick Johnson, Olivier Iffrig, Simon Smart, Christian Herold, Antonino Bonanni, Adrian Jackson, and Mark Parsons. 2019. **An Early Evaluation of Intel's Optane DC Persistent Memory Module and its Impact on High-Performance Scientific Applications**. In The International Conference for High Performance Com- puting, Networking, Storage, and Analysis (SC'19), November 17–22, 2019, Denver, CO, USA. https://doi.org/10. 1145/3295500.3356159





NEXTGenIO has received funding from the European Union's Horizon 2020 Research and Innovation programme under Grant Agreement no. 671951