Evaluation of Knight Landing High Bandwidth Memory for HPC Workloads

Solmaz Salehian
Oakland University
Rochester, MI 48309
ssalehian@oakland.edu

Yonghong Yan
University of South Carolina
Columbia, SC 29201
yanyh@cse.sc.edu

ABSTRACT

The Intel Knight Landing (KNL) manycore chip includes 3D-stacked memory named MCDRAM, also known as High Bandwidth Memory (HBM) for parallel applications that needs to scale to high thread count. In this paper, we provide a quantitative study of the KNL for HPC proxy applications including Lulesh, HPCG, AMG, and Hotspot when using DDR4 and MCDRAM. The results indicate that HBM significantly improves the performance of memory intensive applications for as many as three times better than DDR4 in HPCG, and Lulesh and HPCG for as many as 40% and 200%. For the selected compute intensive applications, the performance advantage of MCDRAM over DDR4 varies from 2% to 28%. We also observed that the cross-points, where MCDRAM starts outperforming DDR4, are around 8 to 16 threads.

KEYWORDS

High Bandwidth Memory, Memory Intensive, Compute Intensive, High Performance Computing

ACM Reference format:
Solmaz Salehian and Yonghong Yan. 2017. Evaluation of Knight Landing High Bandwidth Memory for HPC Workloads. In *Proceedings of IA^3'17: Seventh Workshop on Irregular Applications: Architectures and Algorithms, Denver, CO, USA, November 12–17, 2017 (IA^3'17),* 4 pages.
https://doi.org/10.1145/3149704.3149766

1 INTRODUCTION

The past decade has seen dramatically increased complexity of computer systems, including the increase of parallelism and the adoption of heterogeneous and accelerator architectures. Adding on that recently, memory systems are becoming more complex, e.g. the use of 3D-stacked memory and NVRAM. 3D-stacked DRAMs [8], which connect DRAM dies vertically with through silicon vias (TSVs), increase memory density and the amount of interface channels. Two 3D-stacked DRAM standards and products, Hybrid Memory Cube (HMC) and High Bandwidth Memory (HBM), are already available in commercial products, including the use of HBM in AMD's Fiji GPUs, Nvidia's Pascal GPUs, and Intel's Knights Landing (KNL) manycore architecture.

IA^3'17, November 12–17, 2017, Denver, CO, USA

ACM ISBN 978-1-4503-5136-2/17/11…$15.00
https://doi.org/10.1145/3149704.3149766

It has been well discussed the 3D-stack memory is unable to eliminate memory wall [11]. Particularly, for the latency challenge, 3D-stacked DRAMs may increase latency due to circuit complexity. Our latency evaluation on the latest 64-core KNL shows that the idle latencies of HBM (named as Multi-Channel DRAM (MCDRAM)) and DDR4 are 168.6ns and 140.8ns. The latency tests for other KNL memory and clustering modes show the similar results between the MCDRAM and DDR4 [1]. In general, HBM's latency is about 20% higher than the DDR4 latency in KNL [2].

For bandwidth, 3D-stack memory is designed to sustain much higher memory bandwidth than conventional 2D-stacked DRAMs. It is thus generally agreed that 3D-stacked memory would benefit highly parallel memory and data intensive application on manycore architectures. In this paper, we evaluate the performance impact of using KNL MCDRAM for several HPC applications, including Lulesh, HPCG, AMG, and HotSpot. Our study shows that MCDRAM improves the performance for memory intensive applications (HPCG and Lulesh) for as many as of 40%and 200%. For compute intensive applications, the performance advantage of MCDRAM over DDR4 varies according to the applications and problem sizes, from 2% to 25%. We also observed that the cross-points as the number of threads when MCDRAM starts outperforming DDR4 are around 8 to 16 threads for most applications.

This paper is organized as follows. Section introduces KNL memory and cluster modes of its MCDRAM and DDR4 memory. Section 3 presents the performance results using STREAM benchmark. Section 4 presents our evaluation using HPC workloads. Section 5 concludes the paper.

2 KNL MEMORIES

Intel Knights Landing (KNL) [4] manycore processor is the second-generation Intel Xeon Phi product. It has up to 36 physical tiles each with two x86 cores and two vector processing units. A KNL machine has two types of memory: MCDRAM, which is a version of HBM and DDR4 memory. MCDRAM in KNL totals 16GB capacity and 450GB/s bandwidth. DDR4 is a high-capacity memory with maximum 384GB capacity and 90GB/s bandwidth.

2.1 KNL MCDRAM Memory Modes

The MCDRAM can be configured in three different modes: 1) Cache mode, in which MCDRAM acts as directed mapped L3 cache for DDR4; 2) Flat mode, in which MCDRAM is treated the same as DDR4, but in a different NUMA domain; and 3) Hybrid mode, which

[1] John D. McCalpin, Memory Latency on the Intel Xeon Phi x200 "Knights Landing" processor from http://sites.utexas.edu/jdm4372/tag/memory-latency
[2] Tested using Intel Memory Latency Checker v3.3 from https://software.intel.com/en-us/articles/intelr-memory-latency-checker.

is a combination of cache and flat modes. Some MCDRAM is configured as addressable memory as a new NUMA domain and the other is configured as cache.

2.2 KNL Memory Cluster Modes

Cluster mode refers to the way that memory requests originating from cores reach the memory channel. There are three kinds of cluster mode in KNL: 1) All-to-All, 2) Quadrant mode, 3) Sub-NUMA Cluster (SNC). In all-to-all cluster mode, memory requests can be originated from any core to any memory channel. In quadrant mode, which is the default mode when all channels have the same capacity and number of DIMMs, the mesh network and cores titles are virtually divided into four identical sections. The quadrant mode generally results in less mesh traffic as compared to all-to-all mode and hence can provide better performance.

For the results reported in this paper, we used a KNL 7210 processor @ 1.30GHz configured with the quadrant clustering mode. we used MCDRAM flat mode in order to take full advantage of MCDRAM, The processor has 36 tiles connected by a 2D Mesh interconnect. It has 64 cores with up to 4 threads per core, 16 GB of MCDRAM and 68 GB of DDR4. The host operating system is CentOS 6.7 Linux with kernel version 2.6.32-573.12.1.el6.x86 64. For compiling, Intel ICC has been used with -O2 optimization option.

3 EVALUATION OF BANDWIDTH SCALING USING STREAM BENCHMARK

Our first evaluation used STREAM benchmark [10] to measure the memory bandwidth and its scalability on the KNL machine. It consists of four operations: Copy, Scale, Sum and Traid. Table 1 shows the execution time of the four functions of STREAM using DDR4 and MCDRAM for problems size of 15 Million (M). MCDRAM starts outperforming DDR4 for both Copy and Scale functions from 8 threads onward, and for Sum and Traid, this cross-point is at 16 threads. Before these points, DDR4 shows a bit better performance in terms of execution time. This can be explained by the lower latency of DDR4 than that of MCDRAM before bandwidth becomes the dominating factor of the performance. These points are also the number of threads when DDR4 bandwidths are saturated. The bandwidth trends shown in Figure 1 to Figure 4 verify this. Since 16 threads, DDR4 reaches the bandwidth saturation point and bandwidth levels off after that. MCDRAM delivers better performance by adding more threads after 10 threads. The best performance by MCDRAM for all functions are achieved when the number of threads is 256, which is about 5 times better than using DDR4. It is also worth to mention that hyperthreading (from 64 to 256 threads) does not help the performance much for this workload.

4 APPLICATION EVALUATION

In this section, we evaluate the performance impacts of KNL MCDRAM for memory and compute intensive applications. The performance of memory intensive applications may converge to the saturation point determined by the memory bandwidth. Thus, they can take advantage of HBM to achieve better performance.

4.1 Memory Intensive Applications

HPCG: High Performance Conjugate Gradients (HPCG) is a benchmark created as a complement to High Performance LINPACK (HPL) Benchmark for measuring the performance of and ranking HPC systems [7]. The main difference between HPL and HPCG refers to their main computational kernels, which is matrix multiplication in HPL, and matrix vector multiplication in HPCG [9]. Thus, the ratio of memory access to FLOP in HPCG is significantly higher than HPL [2]. HPCG also exhibits the same irregular access to memory and fine-grain recursive computations[1].

HPCG has been evaluated for three problem sizes (24, 120, 256) across different number of threads. As shown in Table 2, DDR4 and MCDRAM perform very closely for the small numbers of threads, but DDR4 still shows a bit better performance rather than MCDRAM due to the lower latency of DDR4. However, from 16 threads and more, MCDRAM outperforms DDR4, and this fact is more obvious in large problem sizes (120 and 256). The best performance by MCDRAM for problem size 120 is achieved at 64 threads, which is about two times faster than DDR4. For problem size 256, MCDRAM also performs around three times better of using DDR4 at 64 threads. HPCG performance has similar trends as STREAM and the saturation points determined by the DDR4 bandwidth remains constant across different problem sizes at 16 threads.

Lulesh: Livermore Unstructured Lagrangian Explicit Shock Hydrodynamics (LULESH) is a proxy-app which performs a hydrodynamics stencil calculation by using both MPI and OpenMP [6]. Hydrodynamics describes the relative motion of materials caused by a force. Lulesh partitions the problem domain into a collection of elements and estimates the hydrodynamics equations discretely [6]. Generally, the compute intensive part of this application is still considered massive, but it also demands high memory access which has an irregular pattern. It can take advantage of MCDRAM.

Based on the performance results in Table 3, it is clear that for small problem size (20) the margin between DDR4 and MCDRAM is small, less than 10%. For large problem sizes (70 and 100), MCDRAM outperforms DDR4 when the number of threads reaches 16. For the input data size of 70, MCDRAM achieve 57% better performance than using DDR4 at 64 threads, and this improvement for problem size 100 is around 47%. After these points MCDRAM also creates less parallelism overhead. The saturation point of DDR4 for Lulesh varies based on the problem size. For large problem sizes, it stands at 16 threads. From the results, it is also obvious that for small problem size this application achieves better performance by adding more threads until 16 cores, while for larger problem sizes, increasing number of threads until 64 is still worth and can provide better execution time. Thus, this application exhibits certain weak scalability character.

4.2 Computation Intensive Applications

AMG: AMG is an Algebraic Multigrid (AMG) solver for linear systems for unstructured grids [5]. The memory access pattern for this application is irregular. The default problem of AMG which is a Laplace type problem and demands more memory access in comparison with other problems of AMG, has been evaluated. The results in Table 4 indicate that AMG is not impacted much by the

Table 1: STREAM performance (s) for DDR4 and MCDRAM. The results in red color are the cross-point of the number of threads when MCDRAM outperforms DDR4. The results in bold are the best performance achieved.

	Number of threads								
Memory/problem size/function	1	2	4	8	16	32	64	128	256
DDR4-PS=15M-Copy	0.435397	0.220136	0.112345	0.059609	0.038772	0.037819	0.038212	0.039263	0.039532
MCDRAM_PS=15M-Copy	0.457037	0.229367	0.115001	0.05775	0.028958	0.015616	0.009761	0.00996	**0.008285**
DDR4-PS=15M-Scale	0.440568	0.222475	0.113633	0.059729	0.038788	0.037891	0.038142	0.039427	0.040249
MCDRAM_PS=15M-Scale	0.453367	0.227237	0.113996	0.057183	0.028733	0.015667	0.009532	0.010174	**0.007274**
DDR4-PS=15M-Add	0.475274	0.238583	0.12065	0.067228	0.051013	0.050519	0.051147	0.0524	0.053464
MCDRAM_PS=15M-Add	0.525253	0.264652	0.134484	0.069455	0.039043	0.026416	0.015315	0.014271	**0.010627**
DDR4-PS=15M-Triad	0.48141	0.244698	0.125668	0.069466	0.051278	0.050505	0.051214	0.052698	0.052827
MCDRAM_PS=15M-Traid	0.543858	0.274373	0.140221	0.073043	0.043038	0.02887	0.015856	0.01447	**0.010696**

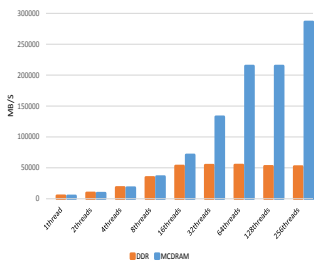

Figure 1: Bandwidth usage of Copy function for DDR4 and MCDRAM.

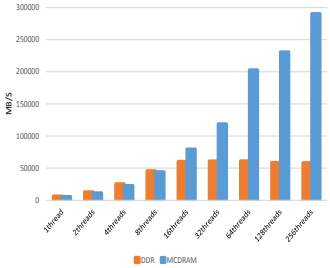

Figure 2: Bandwidth usage of Sum function for DDR4 and MCDRAM.

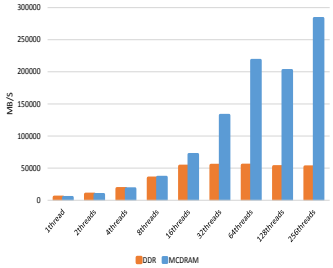

Figure 3: Bandwidth usage of Scale function for DDR4 and MCDRAM.

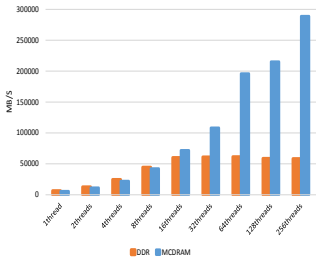

Figure 4: Bandwidth usage of Traid function for DDR4 and MCDRAM.

Table 2: HPCG performance for DDR4 and MCDRAM based on different problem sizes

	Number of threads									
Memory/Problem size	1	2	4	8	16	32	64	128	192	256
DDR4-PS=24	0.413	0.246578	0.170208	0.136	0.121	0.121	0.143	0.193	0.236	0.29
MCDRAM-PS=24	0.424	0.252088	0.172699	0.137	**0.119**	0.12	0.141	0.192	0.236	0.254
DDR4-PS=120	47.51	25.391	12.929	6.91	3.87	2.72	2.42	2.69	2.96	3.22
MCDRAM-PS=120	49.25	26.2325	13.3637	7.11	3.78	2.09	**1.32**	1.34	1.705	1.86
DDR4-PS=256	530.548	294.781	145.211	71.48	37.42	27.1	25.75	27.27	28.34	31.31
MCDRAM-PS=256	560.835	310.951	152.326	73.12	35.91	18.88	**8.54**	8.54	8.8	9.009

Table 3: Lulesh performance for DDR4 and MCDRAM based on different problem sizes

	Number of threads									
Memory/Problem size	1	2	4	8	16	32	64	128	192	256
DDR4-PS=20	26.98	16.16	9.6	6.4	5.1	5.27	6.49	7.87	10.73	15.32
MCDRAM-PS=20	27	16.04	9.55	6.3	**4.97**	4.97	5.57	8.1	10.95	14.17
DDR4-PS=70	3763.85	1935.76	1013.7	554.02	324.97	213.18	193.28	277.17	377.24	469.56
MCDRAM-PS=70	3803.33	1957.34	1029.87	557.63	313.92	184.05	**122.94**	147.96	184.58	205.29
DDR4-PS=100	15979.59	8747.46	4253.11	2269.75	1304.7	877.96	727.55	959.73	1245.7	1539.96
MCDRAM-PS=100	16142.62	8310.88	4307.5	2304.18	1280.26	772.95	**495.16**	527.95	648.36	748.3

Table 4: AMG performance for DDR4 and MCDRAM based on different problem sizes

	Number of threads									
Memory/Problem size	1	2	4	8	16	32	64	128	192	256
DDR4-PS=8	0.5743	0.5812	0.2628	0.2404	0.2809	0.3178	1.1006	1.9945	3.1980	4.6404
MCDRAM-PS=8	0.5812	0.3712	0.2738	**0.2310**	0.2754	0.3177	1.0903	2.0824	3.4203	4.3952
DDR4-PS=25	20.7608	12.6096	8.7321	7.3952	7.3918	7.9169	11.0711	18.8593	25.2377	31.4963
MCDRAM-PS=25	21.1714	12.9787	9.1912	7.3045	**7.1013**	7.2919	10.8136	18.6789	24.9322	31.8620
DDR4-PS=32	46.3767	27.8609	19.3458	15.8239	15.8775	17.4901	24.1028	35.6903	45.5660	55.1687
MCDRAM-PS=32	47.2552	28.4761	19.3595	17.6864	**15.3934**	15.6847	23.0124	34.0167	43.8763	53.1282

Table 5: Hotspot performance for DDR4 and MCDRAM based on different problem sizes

	Number of threads									
Memory/Problem size	1	2	4	8	16	32	64	128	192	256
DDR4-PS=64	0.0781	0.0396	0.0421	0.0409	0.0349	0.0405	0.0529	0.0763	0.0995	0.1390
MCDRAM-PS=64	0.0611	0.0406	**0.0326**	0.0374	0.0368	0.0428	0.0538	0.0798	0.1019	0.1382
DDR4-PS=1024	0.8844	0.4641	0.2481	0.1433	0.0966	0.0770	0.0756	0.0988	0.1313	0.1618
MCDRAM-PS=1024	0.8615	0.4445	0.2398	0.1475	0.0949	0.0750	**0.0714**	0.0954	0.1210	0.153164
DDR4-PS=8192	30.7353	15.9062	7.9791	4.0918	2.2283	1.2724	0.8157	0.8564	1.0689	1.2723
MCDRAM-PS=8192	30.9131	16.0073	8.0585	4.0764	2.0582	1.1213	**0.6792**	0.7256	0.8498	0.806057

memory types. MCDRAM only delivers about 2%-4% better performance than DDR4. For small input dataset (8), the best performance by MCDRAM is achieved when the number of threads is 8 and it also has less parallelism overhead, whereas for larger problem size it is at 16 threads. The cross-point is between 8 and 16 threads.

Hotspot: Hotspot [3] is able to take advantage of MCDRAM because the memory access is not contiguous for this algorithm. It demands more bandwidth because of large amount of cache misses. As shown in Table 5, Hotspot for small problem size (64) achieves best performance with MCDRAM and created less parallelism overhead at 4 threads with almost 28% improvement comparing to DDR4. While for larger problem sizes, this happens at 64 threads, and showing around 6% better performance for problem size 1024 and 20% for 8192. However, this margin for memory intensive applications is higher than 45% for large input dataset. This application is not able to scale well and after 16 number of threads it shows a downward trend for problem size 64. For larger problem sizes increasing the number of threads until 64 can help to performance, but after that hyperthreading does not help either.

5 CONCLUSION

In this work, we evaluated performance of compute and memory intensive applications for both DDR4 and MCDRAM memory in Intel Knight Landing processor. For compute intensive applications, the performance advantage of MCDRAM over DDR4 varies according to the applications and problem sizes, which range from 2% to 28%. For memory intensive applications when the input data size is big enough, MCDRAM is able to deliver much better performance than DDR4, 57% for Lulesh and 200% for HPCG. The results also show that the cross-points as the number of threads, when MCDRAM starts outperforming DDR4 for memory intensive applications, are around 8 to 16 threads.

ACKNOWLEDGMENT

This material is based upon work supported by the National Science Foundation under Grant No. 1652732.

REFERENCES

[1] Jack Dongarra, Michael A Heroux, and Piotr Luszczek. 2016. High-performance conjugate-gradient benchmark: A new metric for ranking high-performance computing systems. *The International Journal of High Performance Computing Applications* 30, 1 (2016), 3–10.

[2] Georgios Goumas, Kornilios Kourtis, Nikos Anastopoulos, Vasileios Karakasis, and Nectarios Koziris. 2008. Understanding the performance of sparse matrix-vector multiplication. In *Parallel, Distributed and Network-Based Processing, 2008. PDP 2008. 16th Euromicro Conference on*. IEEE, 283–292.

[3] Wei Huang, Shougata Ghosh, Sivakumar Velusamy, Karthik Sankaranarayanan, Kevin Skadron, and Mircea R Stan. 2006. HotSpot: A compact thermal modeling methodology for early-stage VLSI design. *IEEE Transactions on Very Large Scale Integration (VLSI) Systems* 14, 5 (2006), 501–513.

[4] James Jeffers, James Reinders, and Avinash Sodani. 2016. *Intel Xeon Phi Processor High Performance Programming: Knights Landing Edition*. Morgan Kaufmann.

[5] Lawrence Livermore National Lab. 2013. Algebraic Aultigrid (AMG). (2013). https://codesign.llnl.gov/amg2013.php.

[6] Lawrence Livermore National Lab. 2013. Livermore Unstructured Lagrangian Explicit Shock Hydrodynamics (LULESH). (2013). https://codesign.llnl.gov/lulesh.php.

[7] Innovative Computing Laboratory. 2015. High Performance Conjugate Gradients (HPCG) Benchmark. (2015). https://www.hpcg-benchmark.org/.

[8] Gabriel H. Loh. 2008. 3D-Stacked Memory Architectures for Multi-core Processors. In *Proceedings of the 35th Annual International Symposium on Computer Architecture (ISCA '08)*. IEEE Computer Society, Washington, DC, USA, 453–464. https://doi.org/10.1109/ISCA.2008.15

[9] Vladimir Marjanović, José Gracia, and Colin W Glass. 2014. Performance modeling of the HPCG benchmark. In *International Workshop on Performance Modeling, Benchmarking and Simulation of High Performance Computer Systems*. Springer, 172–192.

[10] John D. McCalpin. 1991-2007. STREAM: Sustainable Memory Bandwidth in High Performance Computers. (1991-2007).

[11] Milan Radulovic, Darko Zivanovic, Daniel Ruiz, Bronis R. de Supinski, Sally A. McKee, Petar Radojković, and Eduard Ayguadé. 2015. Another Trip to the Wall: How Much Will Stacked DRAM Benefit HPC?. In *Proceedings of the 2015 International Symposium on Memory Systems (MEMSYS '15)*. ACM, New York, NY, USA, 31–36. https://doi.org/10.1145/2818950.2818955