Parallel Computing: Technology Trends
I. Foster et al. (Eds.)

doi:10.3233/APC200044

Exploring High Bandwidth Memory for PET Image Reconstruction

Dai YANG, Tilman KÜSTNER, Rami AL-RIHAWI, and Martin SCHULZ
{*d.yang, tilman.kuestner, martin.w.j.schulz*}*@tum.de*
Chair of Computer Architecture and Parallel Systems
Technical University of Munich

Abstract. Memory bandwidth plays an essential role in high performance computing. Its impact on system performance is evident when running applications with a low arithmetic intensity. Therefore, high bandwidth memory is on the agenda of many vendors. However, depending on the memory architecture, other optimizations are required to exploit the performance gain from high bandwidth memory technology. In this paper, we present our optimizations for the Maximum Likelihood Expectation-Maximization (MLEM) algorithm, a method for positron emission tomography (PET) image reconstruction, with a sparse matrix-vector (SpMV) kernel. The results show significant improvement in performance when executing the code on an Intel Xeon Phi processor with MCDRAM when compared to multi-channel DRAM. We further identify that the latency of the MCDRAM becomes a new limiting factor, requiring further optimization. Ultimately, after implementing cache-blocking optimization, we achieved a total memory bandwidth of up to 180 GB/s for the SpMV operation.

Keywords. Intel Xeon Phi, MCDRAM, Sparse Matrix-Vector Multiplication, Maximum Likelihood Expectation-Maximization, Positron Emission Tomography

1. Introduction

The Intel Xeon Phi product family, based on the Intel Many-Integrated-Core (MIC) architecture, targets high-performance computing (HPC) applications [26]. Especially the Knights Landing (KNL) microarchitecture provides a large number of cores, achieving a high aggregated performance and a high performance per watt ratio [5]. The package also includes 3D-stacked DRAM, which provides a high memory bandwidth. The KNL may be a discontinued product, but the design of both utilizing many-core architecture and high bandwidth memory remains important for modern HPC systems. Other notable products featuring a high bandwidth memory architecture include both NVIDIA and AMD's graphics cards.

Positron Emission Tomography (PET) is a medical imaging modality with clinical value for the detection, staging, and monitoring of many diseases. It is a functional imaging technique, as it allows the observation of metabolic processes. A radioactive tracer is injected into the patient or subject. The tracer

undergoes beta decay, emitting positrons, which annihilated with electrons, creating two 511 keV gamma photons traveling in opposite directions. The scanner consists of a ring of detectors, with scintillator crystals and photodiodes. When two detectors each record a photon within a certain time window, an annihilation event is assumed somewhere along the line connecting the detectors, called the Line of Response (LOR). In reality, we have a Tube of Response (TOR), as two detectors can detect events not only from a line but from a larger, roughly polyhedral portion of the three-dimensional space inside the scanner tube, called Field of View (FOV). The number of detected events influences the quality of the measurement, while the covered area of the field of view by LORs affects the achievable resolution. The resolution is usually better at the center than at the edges of the field of view. For image reconstruction, the field of view is divided into a three-dimensional grid, where each grid cell is called a voxel.

In this paper we use the small animal PET scanner MADPET-II as an example (see Figure 1 (left)). The scanner has a unique design consisting of two concentric rings of detectors, which increases sensitivity while preserving spatial resolution [15]. To obtain an image from the scanner, the detector output – called list-mode *sinogram* – needs to be reconstructed using a system matrix.

There is a number of image reconstruction algorithms used in medical imaging. The algorithms used in our MADPET-II is Maximum Likelihood Expectation-Maximization (MLEM) [24]. A detailed mathematical description of the physical processes involved in tomography systems, such as the attenuation and scattering of photons in the body, is presented by Vazquaz et al. [29].

The main contributions of this paper are:

- We present an optimized MLEM implementation for modern many-core systems.
- We show the impact of increased memory bandwidth on Sparse Matrix-Vector (SpMV) operation performance in MLEM by benchmarking our implementation on an Intel Xeon Phi (KNL) based system.
- We identify the role of latency and propose future optimization recommendations for MLEM and SpMV codes in general.

2. Intel Xeon Phi Knight's Landing

Applications can be classified by the limitation on their performance into three categories, namely compute bound, (memory) latency bound, and (memory) bandwidth bound. To improve the performance of a compute bound application, the utilization of more cores is sufficient. To improve the performance of a memory (bandwidth) bound applications, two memory technologies with a higher memory bandwidth are developed: the *High Bandwidth Memory*(HBM) and the *Hybrid Memory Cube*(HMC). Both of the technologies are based on 3D-stacking of the classic DRAM dies. These memory modules are physically installed onto the processor package. However, stacking of the DRAM chips requires a significantly higher amount of wiring and controlling logic, which results in a higher latency of the memory. On the Intel Xeon Phi processor with Knight's Landing (KNL) ar-

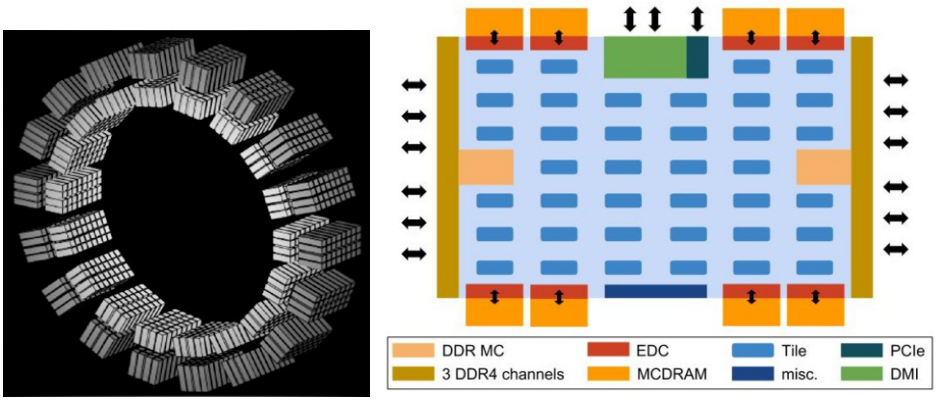

Figure 1. (Left): Illustration of the small animal PET scanner [14]. (Right): Diagram demonstrating an overview of the KNL Architecture.

chitecture, an HMC-based high bandwidth memory called Multi-Channel DRAM (MCDRAM) is embedded in the processor. Finally, to improve the performance of memory (latency) bound application, advanced optimization techniques such as cache-blocking and prefetching can be used.

The Xeon Phi Knights Landing (KNL) is part of the Xeon Phi family of processors, which is based on the Intel MIC architecture. The MIC architecture integrates many x86 processor cores with vectorization support to deliver massive parallelism. It is designed for high-performance computing applications [7].

In general, the KNL chip consists of tiles, MCDRAM, DRAM, and I/O, as shown in Figure 1. Each tile consists of two simple out-of-order cores that are derived from Intel Atom cores (based on the Silvermont microarchitecture). Each core supports up to 4 (hyper-)threads per core - running at 1.3 to 1.5 GHz. They are equipped with a 32KB L1 data cache and a 32KB L1 instruction cache. A 1MB L2 cache is shared among a single tile. The MCDRAM consists of 8x2GB blocks connected to different memory controllers located on different regions of the processor. The two DDR memory controllers support six channels with a bandwidth of up to 90 GB per second. The MCDRAM on KNL processors can be used in three different modes, namely Cache, Flat, and Hybrid. An overview is given below. In this paper, we used the flat mode to control MCDRAM usage.

- **Cache**: MCDRAM can be used as the Last Level Cache (LLC).
- **Flat**: The entire MCDRAM memory is added to the address space extending the space of the existing DDR4 Memory. In this mode, it is possible to allocate memory in the MCDRAM explicitly.
- **Hybrid**: It allows the MCDRAM to be partitioned into a part-cache and part-flat configuration by specifying a ratio between the two, either 75% - 25%, 50% - 50%, or 25% - 75%.

In addition to the MCDRAM configuration, there are several ways to configure the memory subsystem: All-to-all (A2A), Quadrant (quad), Hemisphere (HEMI), Sub-NUMA Clustering 4 (SNC4) and Sub-NUMA Clustering 2 (SNC2).

Detailed descriptions of these cluster modes are given by Jeffers et al. [9] and Sodani et al. [26]. A short overview of the differences between the cluster modes used in this paper is presented through the following scenario:

- All-to-All (A2A): In this cluster mode, memory addresses are uniformly distributed across all Tile Directories (TDs) plus the memory (MCDRAM and DDR) is set to UMA.
- Sub-NUMA Clustering 4 (SNC4): The memory subsystem divides the tiles into 4 clusters resulting in separate cache-coherent clusters.

3. The Maximum Likelihood Expectation-Maximization (MLEM) Algorithm

One widely used iterative reconstruction method for emission tomography is the Maximum Likelihood (ML) reconstruction using the Expectation-Maximization (EM) algorithm, which was proposed by Shepp et al. [24] in 1982. The algorithm uses the iteration scheme given in (1), where N is the number of voxels; M is the number of detector pairs; f is the 3D image that is reconstructed; A is the system matrix of size $M \times N$, which describes the geometrical and physical properties of the scanner; g is the measured list-mode sinogram of size M and q is the iteration number. The algorithm is based on the probability matrix $A = a_{ij}$, where each element represents the probability of a gamma photon discharge from a voxel j being recorded by a given pair of detectors i.

$$f_j^{(q+1)} = \frac{f_j^q}{\sum_{l=1}^{N} a_{lj}} \sum_{i=1}^{N} \left(a_{ij} \left(\frac{g_i}{\sum_{k=1}^{M} a_{ik} f_k^q}\right)\right) \tag{1}$$

The algorithm starts with an initial estimate, a grey image. Then, in each iteration, it executes the following steps:

- **Forward projection**: $h = Af$. Project the current approximation of the image into the detector space.
- **Correlation**: $c_i = \frac{g_i}{h_i}$. Correlate the projection to the actual measurement.
- **Backward projection**: $u = A^T c$. Project the correlation factor back into image space by multiplying with the transposed matrix.
- **Update image**: $f_j^{q+1} = \frac{f_j^q}{n_j} u_j$. Update the image with the back-projected correlation factor and apply a normalization n.

The runtime of the algorithm is dominated by the two sparse matrix-vector operations, forward and backward projection. Note that we do not need to create and store the transposed matrix A^T, as the backward projection can be computed as $u^T = c^T A$.

The system matrix describes the geometrical and physical properties of the scanner. For MADPET-II, the field of view is divided into a grid of $140 \times 140 \times 40$ voxel in x-, y- and z-dimension, respectively. The 1152 detectors result in 664,128 unique detector pairs or lines of response. The matrix was generated by the Detector Response Function (DRF) model [10,27]. The matrix is stored in Compressed Sparse Row (CSR) format using single-precision floating-point numbers. (For a list of commonly used formats see Barrett et al. [1]).

For parallelization, the matrix is partitioned into blocks of rows with approximately the same number of non-zero elements per block. This results in good, albeit not perfect load balancing. A more fine-grained approach, which cuts elements within one row, is possible, but would result in additional management or copying overhead.

4. First Optimization for KNL Architecture and MCDRAM

Our existing MLEM code uses both OpenMP and MPI. In order to achieve the best performance for KNL, we have optimized our MLEM implementation with the following steps:

- We make **all memory allocations on the MCDRAM** by using the `memkind` library.
- We rewrite the matrix loading part to support the special memory allocation in the MCDRAM. In particular, a set of OpenMP threads are created prior to memory allocation, and the matrix is directly copied into the corresponding memory by each thread during initialization. This ensures memory first touch for all threads. We further enforce thread reuse and thread pinning during kernel execution.
- We add `#pragma unroll` and `#pragma ivdep` into the kernel to assist **auto-vectorization** for SIMD execution using the AVX512 units of the processor.

To summarize, we improved the data loading process to support high bandwidth memory and ensure locality. In addition, we enabled and assisted the auto-vectorization to improve instruction-level data parallelization.

5. Evaluation

To show the influence of MCDRAM on the execution time of MLEM, we compile and run our code on CoolMUC-III, which is built of 148 compute nodes. Each node consists of one Intel Xeon Phi 7210F (Knight's Landing, KNL) processor with 64 cores, 256 threads, 96 GB of main memory and 16 GB of on-chip MCDRAM. The memory subsystem is configured for Sub-NUMA clustering (*SNC4*) mode and all-to-all (*A2A*) modes. The MCDRAM is configured to be *flat* addressable. According to Intel, maximum bandwidth of 490GB/s for the MCDRAM and 90GB/s for the DDR RAM can be achieved on this processor [19].

To investigate the effect of high bandwidth memory, we have run our MLEM code with three different memory configurations: *A2A, DDR-A2A, and SNC4*, where *DDR-A2A* represents the result of the native execution on the DDR4 main memory. The MCDRAM itself is also set to flat mode. As mentioned in Section 4, we use the *memkind* library and its `hbw_alloc` to explicitly allocate memory on the MCDRAM. For experiments on the DDR-4 memory, a standard `malloc` is used. We run setup runs ten times. During each run, the algorithm records the iteration time for the forward projections and backward projections, as well as the total iteration time. The first iteration is disregarded as we consider it

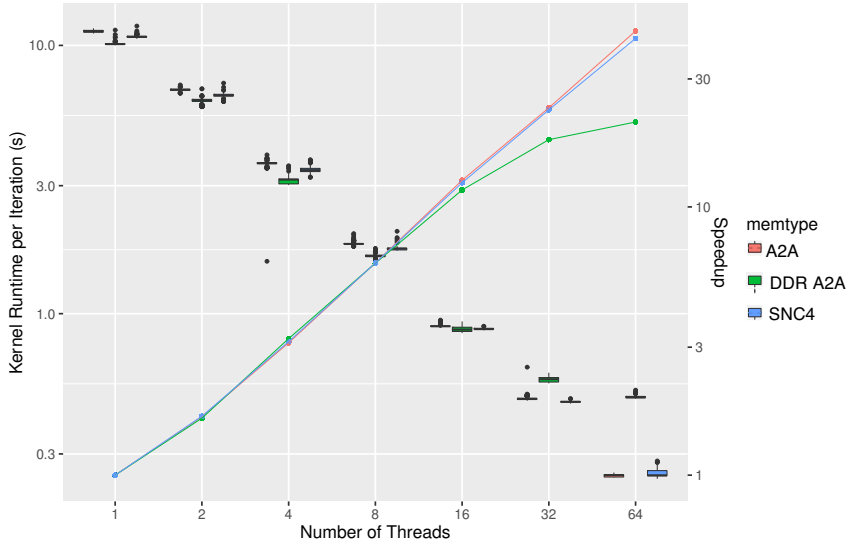

Figure 2. Runtime Comparison of MLEM on CoolMUC-III. The bars (from left to right in each cluster) represent *A2A*, *DDR-A2A*, and *SNC4*.

as warm-up time. The total kernel runtime per iteration, its speedup, and the memory bandwidth for the forward projection are assessed. In this paper, we analyze the forward projection for evaluation because the backward projection requires random access into memory space, which is significantly slower than streaming as required by the forward projection. The results of our experiments are summarized in Figure 2 and 3.

Figure 2 shows the comparison of runtime per iteration in seconds on CoolMUC-III using different memory configurations/modes. Best performance is achieved using the *A2A* setting on the MCDRAM. The *SNC4* setup follows with a slightly higher runtime. The fastest runtime with 64 threads on *A2A* mode is around 0.24s per iteration. We achieve a maximum speedup of about 50x, showing almost perfect scaling behavior. The runtime of the *DDR-A2A* version is significantly higher when using 32 or 64 OMP threads. In addition, the speedup curve indicates a typical saturating line that shows reducing speedup with increasing numbers of threads, indicating that the memory bandwidth limit is hit.

However, our speedup curve indicates a near linear increase of speedup in relation to the number of threads for executions on the MCDRAM. This shows that the code is capable of exploiting the higher memory bandwidth.

Figure 3 shows the corresponding bandwidth achieved for the forward projection, which contributes up to ˜50% to the runtime per iteration. The best memory bandwidth is achieved using the *A2A* mode on MCDRAM, which reflects the result from Figure 2. The slightly higher bandwidth on *A2A* mode over *SNC4* mode is also found in other research, such as stated by Ramos et al. [20]. However, the difference between A2A and *SNC4* is not significant. The maximum bandwidth at 76GB/s on the DDR-4 memory is close to the maximum of 90GB/s with stream benchmark. The bandwidth observed on the MCDRAM is double as high as on the DDR-4, with a maximum of appox. 150GB/s. Although we are

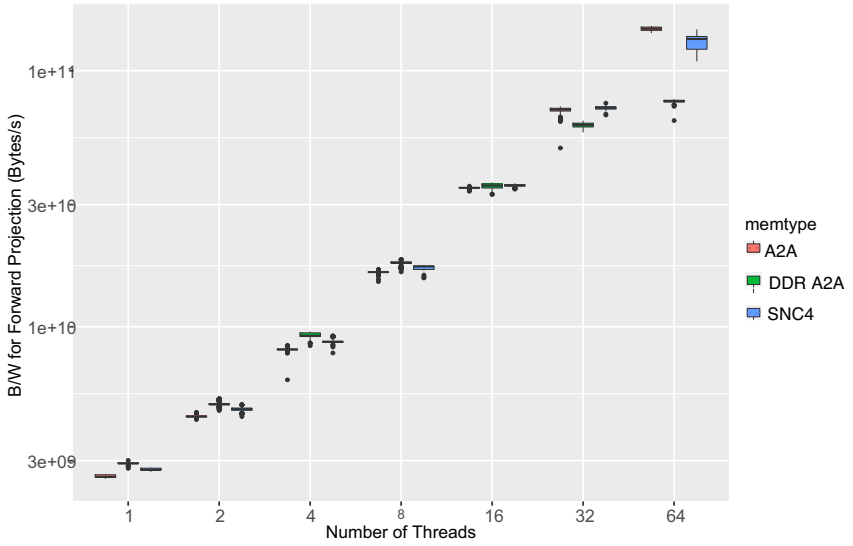

Figure 3. Memory Bandwidth Utilization Comparison of MLEM on CoolMUC-III. The bars (from left to right in each cluster) represent *A2A*, *DDR-A2A*, and *SNC4*.

able to exploit the higher bandwidth of the MCDRAM, we are not able to utilize the full memory bandwidth limit of 490GB/s on MCDRAM. We assume that the cause of this huge gap is the small L2 cache size, which makes it impossible to store the entire working vector within the cache, requiring more frequent load-/store operations. Combined with a high latency introduced by the MCDRAM, the kernels suffer from the frequent load/store operations. Similar results can also be found in related research by Saule et al. [23]. Anecdotal evidence collected by enforcing software prefetching by using `-qopt-prefetch` shows slightly increased performance, which reflects the memory latency boundary.

Further optimization featuring the implementation of a cache-blocking scheme for the working vector is implemented to reduce latency impact. This way, we are able to achieve a bandwidth of approx. 180GB/s on the MCDRAM with the *A2A* configuration.

6. Related Work

Speeding up iterative emission tomography image reconstruction, such as MLEM, has been an important research topic. Improvements have been made on the algorithmic side [6, 11, 21], as well as on the implementation side [3]. Work has also been done porting the MLEM algorithm to distributed GPU clusters [3, 16].

Lui et al. [12] investigated the performance of sparse matrix-vector multiplication (SpMV) on the Knights Corner (KNC), the predecessor of the Knights Landing architecture. They are able to reach 90% of the device's peak memory bandwidth by using a specialized data structure. Bell and Garland [2] show techniques on how to implement SpMV on GPUs (which typically include high band-

width memory nowadays), resulting in a good performance in several sparsity classes.

As KNL offers a number of configuration possibilities, it is especially important to review the work done in this field, with respect to cluster modes, memory modes, and thread affinity.

Rosales et al. [22] investigated the effect of cluster modes on the performance of HPC applications. The paper uses mini applications (MiniFE [4], MiniMD [4], and LBS3D [30]) to observe the performance differences in *A2A* and *QUAD* cluster modes running 1 to 256 threads. The results from MiniFE and MiniMD, using DRAM or MCDRAM, show that *A2A* mode scales comparable to or better than the quad mode. On the other hand, the results from LBS3D when using MCDRAM show varying performance behavior. When using DRAM with *A2A* mode, the code performs better or similar to *quad* mode. Moreover, *A2A* mode scales slightly better than *QUAD* mode when using MCDRAM and significantly better when using DRAM. Ultimately, the effects of the cluster modes seem to be dependent on the application.

Smith et al. [25] compared the effect of the MCDRAM memory modes, *flat* and *cache*, on the performance of sparse tensor factorization, using several datasets. The results reveal that both *flat* and *cache* mode perform identically when the dataset fits into MCDRAM; otherwise *flat* mode performs better than *cache* mode. Peng et al. [18] thoroughly investigated the effects of memory modes across several applications and benchmarks. The paper shows the performance of XSBench [28] over a range of problem sizes, Graph500 [17] over a range of graph sizes, GUPS [13] over a range of table sizes, MiniFE [4] over a range of matrix sizes, and DGEMM [13] over a range of array sizes in flat and cache modes. The results of XSBench and DGEMM show similar performance, Graph500 and GPUs show varying performance, and MiniFE shows flat outperforms the cache mode.

Jabbie et al. [8] observed the performance of the classical elliptic test problem of the Poisson equation on KNL. The work tests two pinning techniques, scatter and balanced, over a range of processes and threads using a hybrid approach. The results show no observable difference in runtime behavior.

7. Conclusion and Future Work

In this paper, we present an implementation of a medical image reconstruction algorithm, the Maximum Likelihood Expectation-Maximization (MLEM), for Positron Emission Topography (PET), optimized for Intel's KNL architecture high memory bandwidth. We investigated the effects of the higher memory bandwidth on our MLEM and provide optimization considerations for SpMV-like code.

We show that SpMV kernels are able to exploit the higher memory bandwidth. However, the higher memory latency makes it hard to exploit the full potential of the MCDRAM on KNL. The massive parallelization combined with high memory bandwidth and latency hiding via cache-blocking provides a significant speedup of MLEM. Overall we achieve a maximum memory bandwidth of 180GB/s on the KNL processor with MCDRAM, which is a significant improve-

ment for our MLEM code.

The next steps are to optimize the matrix storage format and apply further latency hiding technologies to further speedup MLEM. We will also evaluate the efficiency of high bandwidth memory for MLEM on GPU architectures.

Acknowledgment This work is partially funded by German Federal Ministry for Education and Research under grant title 01|H16010D. Compute resource on CoolMUC-III is sponsored by Leibniz Supercomputer Centre under grant title `pr63qi`.

References

[1] Richard Barrett, Michael W Berry, Tony F Chan, James Demmel, June Donato, Jack Dongarra, Victor Eijkhout, Roldan Pozo, Charles Romine, and Henk Van der Vorst. *Templates for the solution of linear systems: building blocks for iterative methods*, volume 43. Siam, 1994.

[2] Nathan Bell and Michael Garland. Implementing sparse matrix-vector multiplication on throughput-oriented processors. In *Proceedings of the Conference on High Performance Computing Networking, Storage and Analysis*, SC '09, pages 18:1–18:11, New York, NY, USA, 2009. ACM.

[3] Jingyu Cui, Guillem Pratx, Bowen Meng, and Craig S Levin. Distributed MLEM: An iterative tomographic image reconstruction algorithm for distributed memory architectures. volume 32, pages 957–967. IEEE, 2013.

[4] Michael A Heroux, Douglas W Doerfler, Paul S Crozier, James M Willenbring, H Carter Edwards, Alan Williams, Mahesh Rajan, Eric R Keiter, Heidi K Thornquist, and Robert W Numrich. Improving performance via mini-applications. *Sandia National Laboratories, Tech. Rep. SAND2009-5574*, 3, 2009.

[5] Yuta Hirokawa, Taisuke Boku, Shunsuke A Sato, and Kazuhiro Yabana. Performance evaluation of large scale electron dynamics simulation under many-core cluster based on knights landing. In *Proceedings of the International Conference on High Performance Computing in Asia-Pacific Region*, pages 183–191. ACM, 2018.

[6] H Malcolm Hudson and Richard S Larkin. Accelerated image reconstruction using ordered subsets of projection data. *IEEE transactions on medical imaging*, 13:601–609, 1994.

[7] Intel Corporation. Intel® Xeon Phi™ Processors. `https://www.intel.com/content/www/us/en/products/processors/xeon-phi/xeon-phi-processors.html`. Accessed: December 2018.

[8] Ishmail A Jabbie, George Owen, and Benjamin Whiteley. Performance comparison of Intel Xeon Phi Knights Landing. *SIAM Undergraduate Research Online (SIURO)*, 10, 2017.

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

[10] Tilman Küstner, Peter Pedron, Jasmine Schirmer, Melanie Hohberg, Josef Weidendorfer, and Sibylle I. Ziegler. Fast system matrix generation using the detector response function model on Fermi GPUs. In *2010 Nuclear Science Symposium and Medical Imaging Conference*, 2010.

[11] Robert M Lewitt and Samuel Matej. Overview of methods for image reconstruction from projections in emission computed tomography. volume 91, pages 1588–1611. IEEE, 2003.

[12] Xing Liu, Mikhail Smelyanskiy, Edmond Chow, and Pradeep Dubey. Efficient sparse matrix-vector multiplication on x86-based many-core processors. In *Proceedings of the 27th International ACM Conference on International Conference on Supercomputing*, ICS '13, pages 273–282, New York, NY, USA, 2013. ACM.

[13] Piotr Luszczek, Jack J Dongarra, David Koester, Rolf Rabenseifner, Bob Lucas, Jeremy Kepner, John McCalpin, David Bailey, and Daisuke Takahashi. Introduction to the HPC challenge benchmark suite. 2005.

[14] David P McElroy, Mark Van Hoose, Wendelin Pimpl, V. Spanoudaki, Torben Schüler, and Sibylle Ziegler. A true singles list-mode data acquisition system for a small animal PET scanner with independent crystal readout. *Physics in Medicine & Biology*, 50:3323, 2005.

[15] David P McElroy, Wendelin Pimpl, Marcis Djelassi, Bernd J Pichler, M Rafecas, T Schuler, and SI Ziegler. First results from MADPET-II: a novel detector and readout system for high resolution small animal PET. In *Nuclear Science Symposium Conference Record, 2003 IEEE*, volume 3, pages 2043–2047. IEEE, 2003.

[16] Debasis Mitra, Hui Pan, Fares Alhassen, and Youngho Seo. Parallelization of iterative reconstruction algorithms in multiple modalities. In *2014 Ieee Nuclear Science Symposium and Medical Imaging Conference (Nss/Mic)*, pages 1–5. IEEE, 2014.

[17] Richard C Murphy, Kyle B Wheeler, Brian W Barrett, and James A Ang. Introducing the graph 500. *Cray User's Group (CUG)*, 19:45–74, 2010.

[18] Ivy Bo Peng, Roberto Gioiosa, Gokcen Kestor, Pietro Cicotti, Erwin Laure, and Stefano Markidis. Exploring the Performance Benefit of Hybrid Memory System on HPC Environments. In *Parallel and Distributed Processing Symposium Workshops (IPDPSW), 2017 IEEE International*, pages 683–692. IEEE, 2017.

[19] Karthik Raman. Optimizing memory bandwidth in knights landing on stream triad. `https://software.intel.com/en-us/articles/optimizing-memory-bandwidth-in-knights-landing-on-stream-triad`, 2016. accessed on 19. 07. 2019.

[20] Sabela Ramos and Torsten Hoefler. Capability models for manycore memory systems: A case-study with xeon phi knl. In *2017 IEEE International Parallel and Distributed Processing Symposium (IPDPS)*, pages 297–306, 2017.

[21] Andrew J Reader, Kjell Erlandsson, Maggie A Flower, and Robert J Ott. Fast accurate iterative reconstruction for low-statistics positron volume imaging. *Physics in Medicine & Biology*, 43:835, 1998.

[22] Carlos Rosales, John Cazes, Kent Milfeld, Antonio Gómez-Iglesias, Lars Koesterke, Lei Huang, and Jerome Vienne. A comparative study of application performance and scalability on the Intel Knights Landing processor. In *International Conference on High Performance Computing*, pages 307–318. Springer, 2016.

[23] Erik Saule, Kamer Kaya, and Ümit V Çatalyürek. Performance evaluation of sparse matrix multiplication kernels on intel xeon phi. In *International Conference on Parallel Processing and Applied Mathematics*, pages 559–570. Springer, 2013.

[24] Lawrence A Shepp and Yehuda Vardi. Maximum likelihood reconstruction for emission tomography. *IEEE transactions on medical imaging*, 1(2):113–122, 1982.

[25] Shaden Smith, Jongsoo Park, and George Karypis. Sparse tensor factorization on many-core processors with high-bandwidth memory. In *Parallel and Distributed Processing Symposium (IPDPS), 2017 IEEE International*, pages 1058–1067. IEEE, 2017.

[26] Avinash Sodani, Roger Gramunt, Jesus Corbal, Ho-Seop Kim, Krishna Vinod, Sundaram Chinthamani, Steven Hutsell, Rajat Agarwal, and Yen-Chen Liu. Knights landing: Second-generation intel xeon phi product. *Ieee micro*, 36:34–46, 2016.

[27] D Strul, RB Slates, M Dahlbom, Simon R Cherry, and Paul K Marsden. An improved analytical detector response function model for multilayer small-diameter pet scanners. *Physics in medicine & biology*, 48(8):979, 2003.

[28] John R Tramm, Andrew R Siegel, Tanzima Islam, and Martin Schulz. XSBench-the development and verification of a performance abstraction for Monte Carlo reactor analysis. *The Role of Reactor Physics toward a Sustainable Future (PHYSOR)*, 2014.

[29] C Vazquez, MJ Rodriguez-Alvarez, C Correcher, AJ González, F Sánchez, P Conde, and JM Benlloch. Parallelization of MLEM algorithm for PET reconstruction based on GPUs. In *Nuclear Science Symposium and Medical Imaging Conference (NSS/MIC), 2014 IEEE*, pages 1–4. IEEE, 2014.

[30] HW Zheng, Chang Shu, and Yong-Tian Chew. A lattice Boltzmann model for multiphase flows with large density ratio. *Journal of Computational Physics*, 218:353–371, 2006.