Article

Power and Performance Evaluation of Memory-Intensive Applications †

Kaiqiang Zhang [1], Dongyang Ou [1], Congfeng Jiang [1,*], Yeliang Qiu [1] and Longchuan Yan [2]

1 School of Computer Science and Technology, Hangzhou Dianzi University, Hangzhou 310018, China; zkqcs@hdu.edu.cn (K.Z.); oudongyang@hdu.edu.cn (D.O.); qiuyeliang@hdu.edu.cn (Y.Q.)
2 Information and Communication Corporation, State Grid Corporation of China, Ltd., Beijing 100053, China; lcyan@sgcc.com.cn
* Correspondence: cjiang@hdu.edu.cn; Tel.: +86-5718-6919-113

† This paper is an extended version of our paper published in 2018 First BenchCouncil International Symposium on Benchmarking, Measuring, and Optimizing (Bench 2018), Seattle, WA, USA, 10–13 December 2018; LNCS 11459; pp. 189–201.

Abstract: In terms of power and energy consumption, DRAMs play a key role in a modern server system as well as processors. Although power-aware scheduling is based on the proportion of energy between DRAM and other components, when running memory-intensive applications, the energy consumption of the whole server system will be significantly affected by the non-energy proportion of DRAM. Furthermore, modern servers usually use NUMA architecture to replace the original SMP architecture to increase its memory bandwidth. It is of great significance to study the energy efficiency of these two different memory architectures. Therefore, in order to explore the power consumption characteristics of servers under memory-intensive workload, this paper evaluates the power consumption and performance of memory-intensive applications in different generations of real rack servers. Through analysis, we find that: (1) Workload intensity and concurrent execution threads affects server power consumption, but a fully utilized memory system may not necessarily bring good energy efficiency indicators. (2) Even if the memory system is not fully utilized, the memory capacity of each processor core has a significant impact on application performance and server power consumption. (3) When running memory-intensive applications, memory utilization is not always a good indicator of server power consumption. (4) The reasonable use of the NUMA architecture will improve the memory energy efficiency significantly. The experimental results show that reasonable use of NUMA architecture can improve memory efficiency by 16% compared with SMP architecture, while unreasonable use of NUMA architecture reduces memory efficiency by 13%. The findings we present in this paper provide useful insights and guidance for system designers and data center operators to help them in energy-efficiency-aware job scheduling and energy conservation.

Keywords: energy efficiency; memory system; memory-intensive computing; energy proportionality

Citation: Zhang, K.; Ou, D.; Jiang, C.; Qiu, Y.; Yan, L. Power and Performance Evaluation of Memory-Intensive Applications . *Energies* **2021**, *14*, 4089. https://doi.org/10.3390/en14144089

Academic Editor: Sergio Saponara

Received: 5 May 2021
Accepted: 30 June 2021
Published: 6 July 2021

1. Introduction

In recent years, memory-based computing is one of the alternative methods to solve many emerging workloads that are constrained by high data-access costs. Efficient data storage and analysis is one of the critical challenges in the big data paradigm [1–7]. Therefore, the processor-centric computing is transforming to memory-centric computing. Although a single 12 TB memory server has appeared on the IDC market today, in many cases, the data to be processed has exceeded the memory capacity of the server [8]. In addition, application-level scalability is limited by memory capacity and communication latency. A common solution is data parallelization, which divides the dataset into smaller subsets to accommodate memory capacity and parallelization acceleration. More specifically, data flows into and out of the processor in parallel like sparks for rapid analysis [2,4,9,10]. It is also possible to introduce new memory hierarchies, such as 3-D memory stacking, to improve bandwidth, energy efficiency and scalability [11–19]. How to reduce data movement and energy consumption as much as possible also depends on these new storage technologies in large-scale storage systems. In addition, ref [3,20] also proposed memory-based data calculation, which reduces the energy consumption of the server system by performing calculations in the memory module. In [21], the author believes that there is no universally accepted solution to provide efficient distributed shared memory.

For computing-intensive applications, data processing can be easily accelerated using many core processors or GPUs. However, for memory-intensive applications, the application performance is highly correlated with memory capacity and bandwidth. DRAM is an important source of server power consumption, especially when the server is running memory-intensive applications. Although power capping and thermal throttling of processors are well investigated and commonly used in data centers, fine-grained and scalable power-aware adaptation of memory systems is still an open problem. Current energy-efficiency-aware scheduling assumes that DRAM is also energy proportional like processors. However, the non-energy proportionality of DRAM significantly affects the energy consumption of the whole server system, especially for memory-intensive applications. However, the non-energy proportionality of DRAM causes the power consumption model to be unable to accurately represent the energy consumption of the server, which affects the energy consumption of the entire server system, especially for memory-intensive applications. Therefore, a full understanding of the server energy ratio under memory-intensive workloads can help better place workloads and increase the energy savings of data center hybrid resource scheduling [22,23]. For example, the memory of each core will also change when the scale of the system increases, which will affect the performance of the application and the cost of the overall system.

In response to the continuous increase in energy consumption caused by the continuous expansion of the data center, the industry standards organization has developed the SPECpower_ssj2008 [24] (referred to as SPECpower in the rest of this article) benchmarks to evaluate the energy efficiency of servers. SPECpower is widely used to characterize the energy efficiency of the system at different utilization levels. Mainstream server vendors submit the SPECpower test results to SPEC and provide them online after passing the review.

However, the SPECpower benchmark will not stress memory system well, because it is a server-side Java benchmark. In Table 1, we list the per-core memory statistics (MPC, the ratio of installed memory capacity to installed processor cores) for 658 servers released before 2020.

Table 1. Memory pre core statistics of published servers with SPECpower_ssj results.

memory per core(GB/core)	0.06	0.25	0.48	0.5	0.57	0.66	0.85	0.88	1.0	1.2	1.3	1.45	1.5	1.6	1.71	1.77	2.0
count	1	1	1	2	1	15	1	2	153	3	32	7	68	4	20	13	145
memory per core(GB/core)	2.25	2.28	2.5	2.6	2.9	3.0	3.2	3.4	3.5	4.0	4.3	5.3	6.0	6.8	8.0	10.6	16.0
count	1	1	1	9	6	37	1	78	1	33	1	4	1	2	7	4	2

It can be observed that in all the 658 SPECpower results published, there are only 13 servers with a single-core memory greater than or equal to 8 GB/core, and only 2 servers with a single-core memory of up to 16 GB/core. Most servers have less than 4 GB/core.

Assume we have a single-node server equipped with 8 Xeon 8260 CPUs (one of the most common CPU on the market in 2019Q4) and 8 TB memory (the maximum memory capacity supported by the processor), the memory per core is 42.6 GB/core. If the processor has fewer cores as most usual configurations of 2 or 4 sockets per node, the memory per core will be significantly greater than 42.6 GB/core. Therefore, from this perspective, the SPECpower results cannot be an ideal and reliable source for an energy efficiency study of large memory systems. This motivates us to investigate the energy efficiency of servers with large memory installation.

In this paper, we use the STREAM benchmark to test three rack servers with different workload intensities to study the energy efficiency of large memory servers running memory-intensive applications. Since the SPECpower benchmark cannot stress memory system well, we use the STREAM benchmark to check how the performance and power changes with different memory stress levels (multi-threads) or racks or modules. If we can find that the memory system has different energy efficiency (or proportionality patterns) from the CPU, at least under different workload types, we can provide some insights for workload placement in clouds or big data analytics scenarios. Our experiments show that hardware configuration can significantly affect server energy efficiency for memory-intensive applications. The findings we presented in this paper provide useful insights and guidance for system designers and data center operators to achieve energy-efficiency-aware job scheduling and energy conservation.

The remainder of this paper is organized as follows. In Section 2, we summarize related work. In Section 3, we first describe the server energy efficiency and energy proportionality from the published SPECpower benchmark results and introduce the energy efficiency metric for servers with large memory installations. In Section 4, we provide experimental results and observations of the energy efficiency of servers with large memory under typical memory-intensive workloads. In Section 5, we characterized the energy efficiency of memory systems, including the economies of scale in memory utilization, comparison of energy efficiency between SMP and NUMA architecture, and we also derive insights on energy efficiency of memory-intensive applications. We conclude the paper and make remarks on future work in Section 6.

2. Related Work

Nowadays, owing to rapid advancement in hardware technology, the ever-increasing main memory capacity has promoted the development of memory big data management and processing [5]. Although in-memory processing moves data into memory and eliminates disk I/O bottlenecks to support interactive data analysis, in-memory systems are more susceptible to the utilization, time/space efficiency, parallelism, and concurrency control of modern CPU and memory hierarchies than disk I/O-based systems [25–29].

In many cases, memory bandwidth restricts the performance of computer systems. Besides, because of pin and power constraints of CPU packages, it is also a challenge to increase the bandwidth. To increase performance under these restrictions, we have proposed the near-DRAM Computing (NDC), near-DRAM acceleration (NDA) architectures, Processing-In-Memory (PIM), Near Data Processing (NDP), or memory-driven computing [30–34]. In [30], the authors proposed Chameleon, an NDA architecture that can be achieved without depending on 3D/2.5D-stacking technology and seamlessly integrated with large memory systems for servers. Experiment has shown that a Chameleon-based system can provide 2.13× higher geo-mean performance while consuming 34% lower geo-mean data transfer energy than a system that integrates the same accelerator logic within the processor.

Recently, new memory hierarchies like 3-D memory stacking are proposed for promotion in energy efficiency, bandwidth, and scalability. For example, the hybrid memory cube (HMC) [35] has promised to enhance bandwidth and density and decrease power consumption for the next-generation main memory systems. Besides, to fill the gap between processors and memories, 3-D integration gives a second shot to revisit near memory computation. Active Memory Cube (AMC) [36], which has been proposed recently, contains general-purpose host processors and in-memory processors (processing lanes), which is specially designed and would be integrated in a logic layer within 3D DRAM memory. DRAM contains multiple resources called banks that can be accessed in parallel and maintain state information independently. In Commercial Off-The-Shelf (COTS) multicore platforms, banks are shared among all cores in common, even if programs running on the cores do not share memory space. In this situation, it is hard to predict memory as a result of contention in the shared banks [37]. For the sake of testing different forms of memory, various benchmarks and system have been implemented by some researchers [38–41].

In multi-core platforms, memory is shared among all processor cores. However, the gains of compute offered by multi-cores are often counteracted by degradation of performance towing of shared resources, such as main memory [42–46]. In order to efficiently use multi-core platforms, tightly binding the interference when accessing shared resources is required. For example, due to interference in the shared, a task running on one core can be delayed by other tasks running simultaneously on other cores DRAM main memory. In some cases, such memory interference delay can be large and highly variable. A tight upper bound on the worst-case memory interference in a COTS-based multi-core system was proposed by Kim [47] and he explicitly modeled the major resources in the DRAM system, including banks, buses, and the memory controller. Dirigent [48] is proposed to balance the performance of latency-critical jobs that finish sooner than required with higher system throughput. Fine time granularity QoS problems for GPUs in heterogeneous platforms [49] have been also tackled by Min et al. However, it is not common to use progress heuristics for the GPU, and the mechanism proposed is restricted to managing main memory bandwidth contention between the CPU and GPU.

In big data analytics, in order to address the problems of limited bandwidth, energy inefficiency, and limited scalability by enabling in-memory computations using non-volatile memristor technology [50], Computation-in-Memory (CIM)-based architectures are proposed. They found that in large-scale data analytics frameworks, the CPU is the main performance bottleneck of these systems. For example, the Tachyon [51] file system outperforms in-memory HDFS by 110× for writes. In [52], the authors propose FusionFS, a distributed file system and distributed storage layer local to the compute nodes, which saves an extreme amount of data movement between compute and storage resources and is in charge of most of the I/O operations. Compared with popular file systems such as GPFS, PVFS, and HDFS, FusionFS is better. epiC [53] is a big data processing framework, which is in order to tackle the Big Data's data variety challenge. epiC introduces a general actor-like concurrent programming model, which is independent of the data-processing models and is for specifying parallel computations. In [54], the authors propose DigitalPIM, a Digital-based Processing In-Memory platform, which has abilities to accelerate fundamental big data algorithms in real time with orders of magnitude more energy efficient operation. In [55], the authors proposed power management schemes to raise the speedup of prior RRAM-based PIM from 69× to 273×, pushing the power usage from about 1 W to 10 W.

However, in [56], the authors found that naive adoption of hardware solutions does not guarantee superior performance over software solutions and point out problems in such hardware solutions that limit their performance, although hardware solutions can supply promising alternatives for realizing the full potential of in-memory systems. In [57], the authors found that if we increase the number of DRAM channels it will decrease DRAM power and improve the energy-efficiency across all applications at the same time.

In heterogeneous platforms, the CPU and the GPU are integrated into a single chip for higher throughput and energy efficiency. Its memory bandwidth is the most critically shared resource in such a single-chip heterogeneous processor (SCHP), requiring discreet management to maximize the throughput. Based on analysis of memory access characteristics, Wang et al. [58] proposed various optimization techniques and improved the overall throughput by up to 8% compared to FR-FCFS.

On modern multi-core platforms, memory bandwidth is highly variable for more memory-intensive applications. Jiang et al. [59] found that memory configuration on a virtualized platform also influences the server's power and performance. In [60], the authors proposed an efficient memory bandwidth reservation system, MemGuard, which has provided bandwidth reservation to guarantee the bandwidth for temporal isolation, utilizing the reserved bandwidth with efficient reclaim to maximal. It improves performance by sparing the best effort after satisfying each core's reserved bandwidth. In [61], the authors proposed to move computation closer to main memory, which offers an opportunity to reduce the overheads associated with data movement and they explore the potential of using 3D die stacking to move memory-intensive computations closer to memory. In [62], the authors propose OffDIMM, which can map a memory block in the address space of the OS to a subarray group or groups of DRAMs and sets a deep power-down state for the subarray group when offlining the block. However, OffDIMM decreases background power by 24 percent on average without notable performance overheads.

All in all, a processor's frequency scaling and power optimization is well investigated, but the research in memory-related power performance optimization is still insufficient. Furthermore, when in-memory computing becomes the mainstream paradigm for big data analytics, power consumption of large memory dominates. This drives us to investigate the power characteristics of servers running memory-intensive applications.

3. Notations of Server Energy Efficiency Evaluation

To drive server energy efficiency improvements, SPEC established SPECpower, which is the authoritative benchmark to measure the power and performance of computer systems in the industry. SPECpower's workload is designed to evaluate the energy efficiency and performance of small and medium-sized servers running server-side Java applications at different utilization levels. That is why SPECpower results are not ideal and reliable sources for an energy efficiency study of large memory systems are required.

However, the methodology of energy efficiency measurement and evaluation of SPECpower is still worthy of reference. Although SPECpower does not put pressure on storage components, it tests CPU, memory, cache, JVM, and other operating system components. Its detailed workload characteristics can be found in [63]. Specifically, SPECpower will report the server power consumption at different utilization levels within a set time period. SPEC has formulated very strict evaluation rules for SPECpower and requires the information of the tested system to be fully disclosed in the report. That is why 40 results published by the SPEC are marked as non-compliant and not accepted by SPEC.

We give a sample result of a server in Table 2, from the results released by SPECpower_ssj2008 in 2016, the server's memory per core of 16 GB/core.

Table 2. An example of SPECpower_ssj2008 testing result in 2016.

Performance			Power	Performance to Power Ratio
Target Load	**Actual Load**	**ssj_ops**	**Average Active Power (W)**	
100%	99.80%	24,662,648	3868	6377
90%	90.10%	22,252,836	3481	6393
80%	80.00%	19,758,684	3032	6517
70%	70.00%	17,284,975	2611	6619
60%	60.00%	14,824,481	2340	6336
50%	50.00%	12,350,615	2143	5764
40%	40.00%	9,877,126	1971	5011
30%	30.00%	7,410,001	1823	4064
20%	20.00%	4,949,964	1674	2956
10%	10.00%	2,475,968	1531	1618
Active Idle		0	1080	0
$\sum$ssj_ops/$\sum$power				5316

In order to better understand this paper, we have listed some notations and terms of SPECpower benchmark results:

(1) *Utilization*. In the target load column of specpower results in Table 2, which assume

benchmark to delete all hardware components concertedly, there are 10 utilization levels, ranging from 10% to 100%.

(2) *Peak utilization*. We refer to 100% utilization as peak utilization.

(3) *Energy efficiency* (EE). It is defined as the performance to power ratio with unit of ssj_ops per watt. The formula is as follows:

$$\text{EE} = \frac{ssj\_ops}{Average\ Active\ Power\ (W)} \tag{1}$$

In Table 2, the energy efficiency values are the last column named "performance to power ratio" and we abbreviate energy efficiency as EE. However, for memory systems, we use bandwidth per watt (BpW) to measure memory energy efficiency (MEE):

$$\text{MEE} = \text{BpW} = \frac{Perceived\ Bandwid(MB/s)}{System\ Power(watts)} \tag{2}$$

(4) *Server overall energy efficiency*. The server's overall performance to power ratio, that is, the ratio of the sum of ssj_ops to the sum of 10 utilization levels (from 10% to 100%) and the sum of active idle power ($\sum ssj\_ops/\sum power$). In addition, the server overall energy efficiency is also used as its SPECpower score. For example, in Table 2, the overall energy efficiency of the server (total score) is 5316.

(5) *Peak energy efficiency*. It is defined as the highest energy efficiency of a server among all utilization levels. For example, in Table 2, the server peak energy efficiency is 6619 (at 70% utilization).

(6) *Energy Proportionality* (EP). In this paper, we use the energy proportionality (EP) metric proposed in [64]. Taking the server in Table 2 as an example, we can draw its normalized utilization–power curve in Figure 1. The solid line in Figure 1 is the EP curve of the server in Table 2, the dotted line is the ideal energy proportionality server and the dashed line is our tested and untuned server with 16 GB of memory per core running the SPECpower benchmark. Based on this, we can compute the EP of a real server by the following formula [64]:

$$\text{EP} = 1 - \frac{Area_{real} - Area_{ideal}}{Area_{ideal}} \tag{3}$$

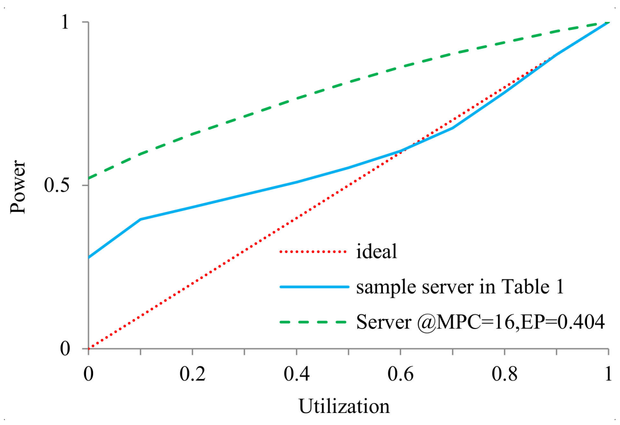

Figure 1. Energy proportionality curve of the server in Table 2 and an ideal energy proportional server (power normalized to power at 100% utilization).

From Equation (3), we can see that EP is greater than or equal to zero but less than 2.0. For the ideal energy proportional server, its EP value is 1.0, which is of great reference significance for the study of energy efficiency characteristics of servers.

4. Experiment Results on Typical Memory-Intensive Workloads

4.1. Experiment Setup

Similarly, we can also derive a server's energy proportionality curve using a memory-intensive benchmark. We used STREAM [65], NAMD [66], and CloudSuite [67], as the memory benchmark. We modified the code to change the array size in STREAM to vary the workload from idle to 100% utilization. We ran benchmarks on three different 2U rack servers, which ran the same x64 version CentOS 7 with Linux kernel 3.10. All the power data were measured by a WattsUP.Net power meter. The base configuration of these servers is listed in Table 3.

Table 3. Base configuration of tested 2U servers.

No.	Name	Hardware Availability Year	CPU Model	Total Cores	CPU TDP (Watts)	Memory (GB)	DISK
#1	ThinkServer RD640	2014	2×Intel Xeon E5-2620 #1	12	80	160(16 G×10) DDR4 2133 MHz	1×SSD 480 GB
#2	ThinkServer RD450	2015	2×Intel Xeon E5-2620 #2	12	85	192(16 G×12) DDR4 2133 MHz	1×SSD 480 GB
#3	Fujitsu	2019	2×Intel Xeon 8260	40	165	384(16 G×24) DDR4 2666 MHz	1×SSD 480 GB

4.2. Results of STREAM Workload

In order to stress the memory system, we ran different numbers of concurrent STREAM threads with varying array size from 4 GB to 16 GB. Due to space limitation, we only provide the results of 4 GB. We present the power consumption of the tested servers in Figures 2 and 3.

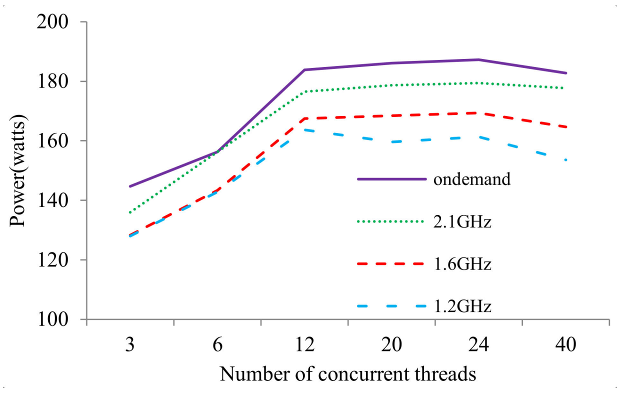

Figure 2. Power consumption of server #1 with array size = 4 GB.

When the cpu or server temperature is too high, the processor's power capping and thermal throttling technology will appropriately reduce CPU frequency to protect the server, in order to protect the cpu or server, so the power consumption of 48 threads is less than 36 threds on server #2. However, overall, our experiments show that with the increment of concurrent threads and therefore memory utilization, the power consumption of the server also increases. We present the perceived bandwidth of a single thread of the tested servers in Figures 4 and 5. The memory energy efficiency is listed in Figure 6.

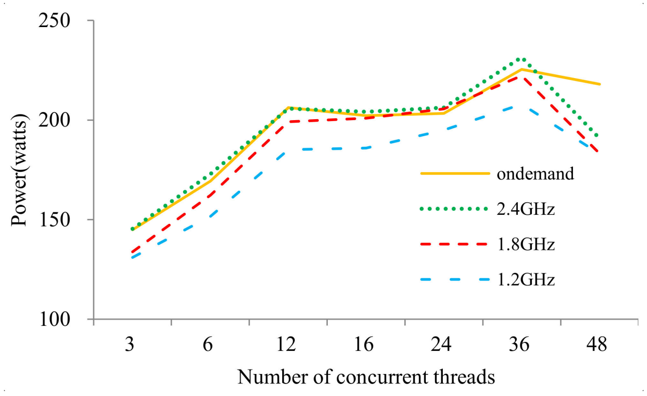

Figure 3. Power consumption of server #2 with array size = 4 GB.

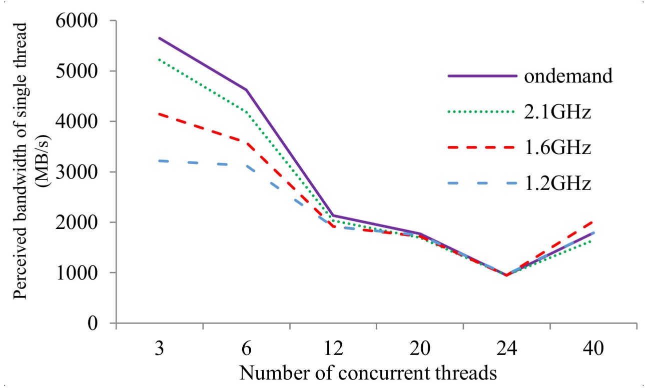

Figure 4. Average perceived bandwidth of single thread with array size = 4 GB (MB/s) on server #1.

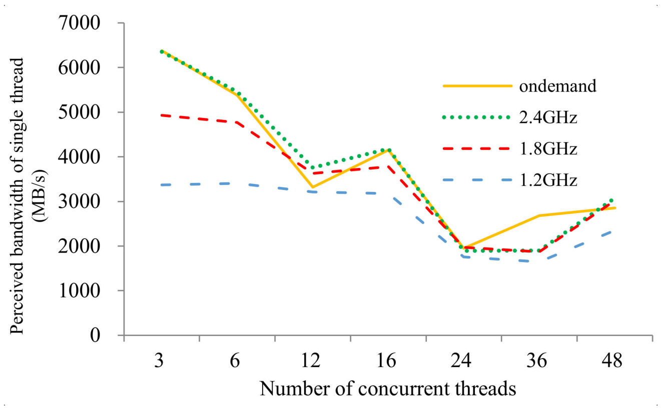

Figure 5. Average perceived bandwidth of single thread with array size = 4 GB (MB/s) on server #2.

From Figures 3–6, we observe that (take server #2 as an example):

(1) When the number of concurrent threads is 36, the power consumption and CPU utilization are the highest at the CPU frequencies of 1.2 GHz, 1.8 GHz, 2.4 GHz, and on-demand governor. Generally, power consumption grows with the CPU frequencies.

(2) When threads increase, the perceived bandwidth of triad computation in a single STREAM thread decreases at first and reaches its lowest at 36 threads. Then it bounces a little at 48 threads because of the contention and starvation of execution threads.

(3) The perceived bandwidth increases while the bandwidth growth rate decreases as CPU frequency increases. Moreover, the bandwidth of different CPU frequency is almost the same at 24 threads, which is because the server has 12 physical cores and 24 execution threads in total.

(4) Both the memory energy efficiency and its change rate decreases as the number of concurrent threads increases. It can also be inferred that frequency scaling cannot improve memory energy efficiency a lot in a highly contented condition.

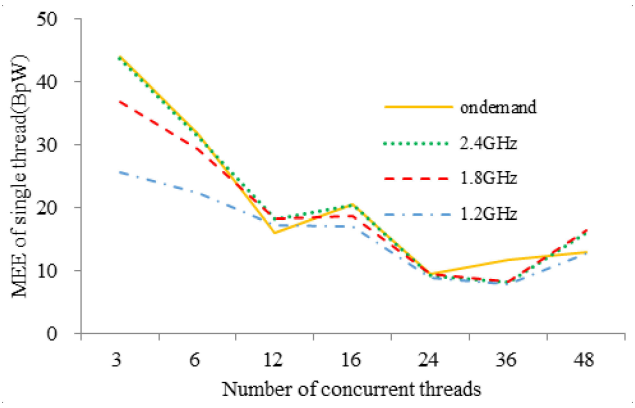

Figure 6. Average memory energy efficiency of single thread with array size = 4 GB (MB/s) on server #2.

4.3. Results of NAMD Workload

We used the NAMD (Nanoscale Molecular Dynamics) simulator to simulate large systems (millions of atoms). We run the NAMD simulations of two virus structures, one contains 8 million atoms and another contains 28 million atoms, namely, the stmv.8M.namd configuration and stmv.28M.namd configuration. The results are shown in Figures 7 and 8. For all NAMD experiments, the system power is significantly correlated with the memory and CPU utilization on different machines with different hardware configurations and CPU generation. However, the correlation coefficient on server #2 is less than that of server #1. One possible reason may be that server #2 has a newer CPU than server #1 and the CPU difference makes sense when running NAMD.

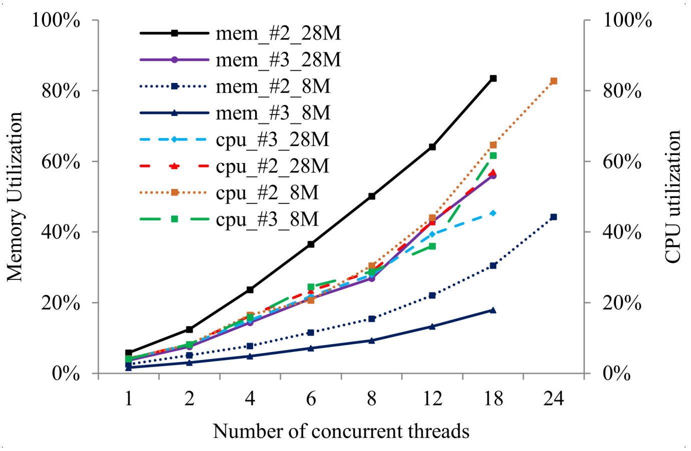

Figure 7. CPU and Memory utilization of NAMD.

We list the Pearson correlation coefficients of power and memory and CPU utilization on server #1 and #2 in Table 4. We can see that for NAMD benchmark, both memory and CPU utilization are good indicators for system power consumption on both server #1 and #2.

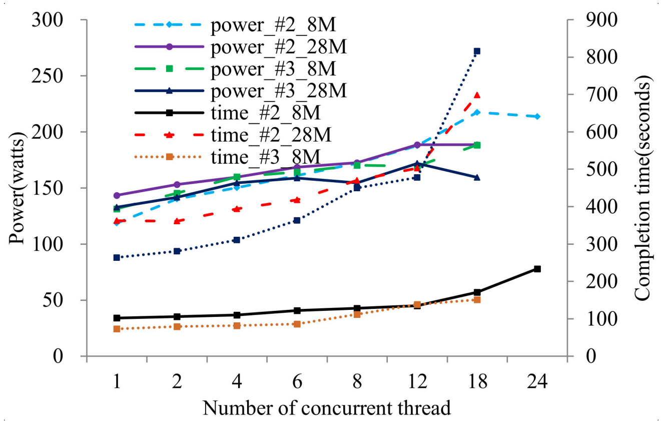

Figure 8. Power and completion time of NAMD.

Table 4. Correlations among power, memory, and CPU utilization running NAMD.

Server	Power-Memory	Power-Cpu	Cpu-Memory
#1_8M	0.936	0.958	0.995
#1_28M	0.973	0.966	0.997
#2_8M	0.922	0.938	0.983
#2_28M	0.671	0.750	0.944

4.4. Results of CloudSuite Workload

In order to confirm if the memory utilization is a good indicator for system power under other memory-intensive applications, we ran the CloudSuite In-Memory Analytics on server #2. The results show that neither memory nor CPU utilization is a good indicator for system power consumption. For example, the correlation coefficient of power and CPU utilization is 0.053 and −0.09 in Table 5 when we run the CloudSuite in-Memory Analytics and both In-memory Analytics and Data Serving benchmarks. We plot the real-time power and system utilization data in Figures 9 and 10. We also conducted experiments where the memory utilization ranged from 30% to 98% and, again, neither memory nor CPU utilization is a good indicator for system power consumption. This implies that we should not implement power-aware scheduling according to only a single parameter like memory or CPU utilization, even for large memory nodes running memory-intensive applications.

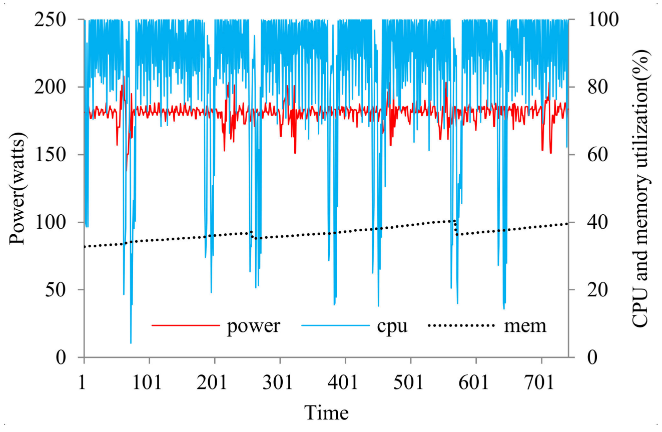

Figure 9. Power and utilization of in memory analytics on server #2.

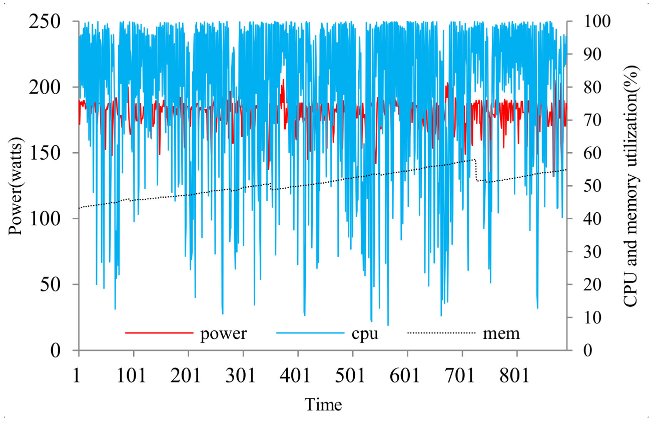

Figure 10. Power and utilization of in memory analytics and data serving on server #2.

Table 5. Correlations among power, memory and CPU utilization running in-memory analytics on server #2.

	Power-Memory	Power-Cpu	Memory Utilization
IM	−0.57	0.05	0.39
IM_DS	−0.52	−0.09	0.48

5. Characterizing Energy Efficiency of Memory Systems

5.1. Economies of Scale in Memory Utilization

In order to investigate the power consumption of each server at varying memory utilization, we conduct experiments with different concurrently running STREAM threads. We then compute the power consumption per percentage of memory utilization in Figures 11 and 12. For other servers and array size, we can obtain similar results. Figures 11 and 12 suggest that when the number of threads increases, the power per percentage of memory utilization decreases.

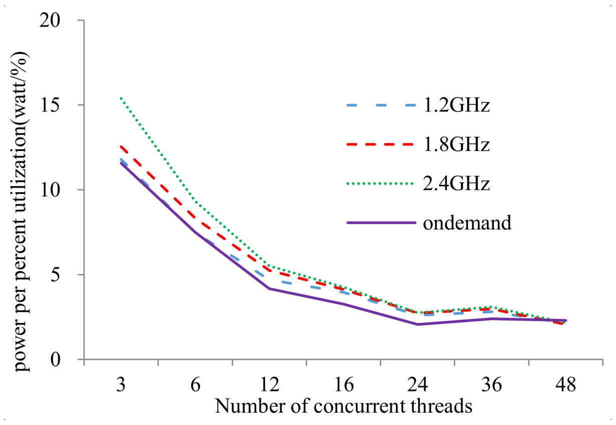

Figure 11. Power per percent memory utilization of server #2 with array size = 4 GB.

We also present the power per percent utilization of server #2 running the SPECpower benchmark in Figure 13 and compare the power per percent utilization of SPECpower and STREAM of server #2 in Figure 14. From Figures 13 and 14, we observe that the power consumption per percent utilization of SPECpower and STREAM benchmark decreases when system utilization increases. However, SPECpower has lower power per percent utilization than STREAM during all utilization levels.

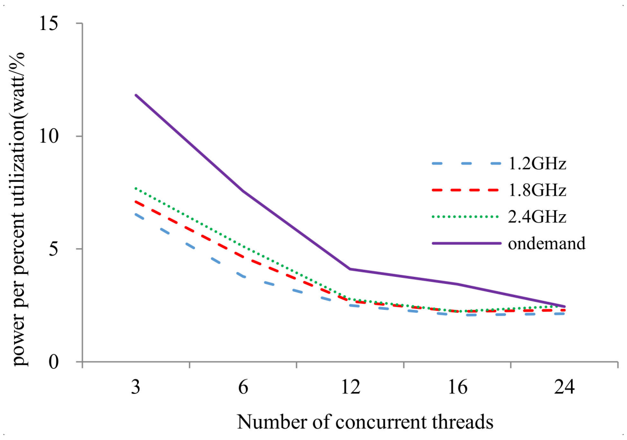

Figure 12. Power per percent memory utilization of server #2 with array size = 8 GB.

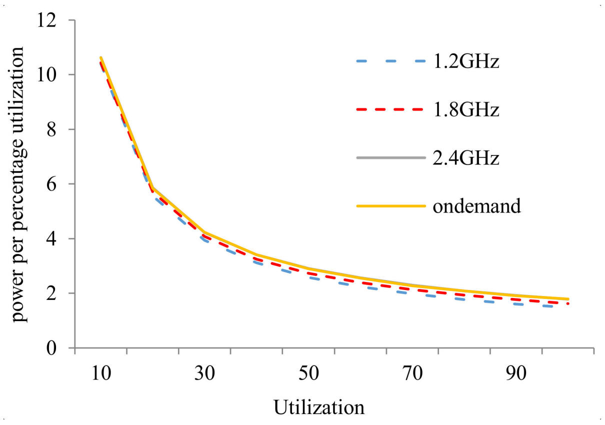

Figure 13. Power per percent utilization of server #2 running SPECpower benchmark.

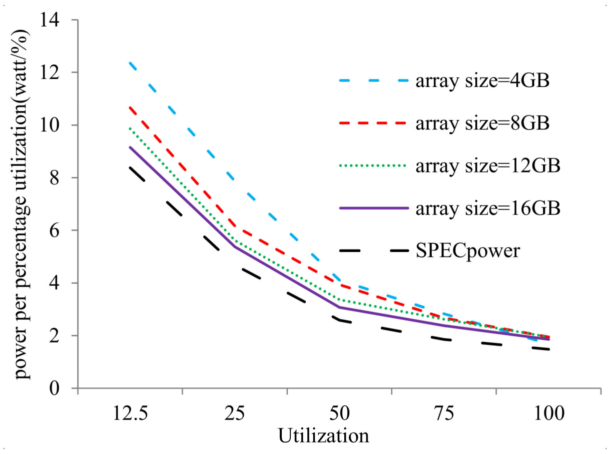

Figure 14. Power per percent utilization of server #2 running SPECpower and STREAM benchmark (CPU frequency at 1.2 GHz).

5.2. SMP and NUMA Energy Efficiency Comparison

Server memory access architecture is divided into SMP and NUMA. In the SMP architecture, all CPUs share all resources (such as bus, memory, I/O system, etc.). In addition, since data exchange between the CPU and the memory is performed by the bus, the bus bandwidth can easily become a bottleneck of the SMP data transmission. To solve the limitation of the SMP bus bandwidth, the most intuitive solution is to increase the number of buses, so the non-uniform memory access (NUMA) architecture emerged. NUMA is a memory architecture designed for multiprocessor devices. In this architecture, the memory access time depends on the location of the memory relative to the processor, and it is quicker for processors to gain access to memory in the local node than in different nodes. This is due to the fact that there are multiple NUMA nodes, which have their own CPU and memory. In the NUMA architecture, each node's CPUs share a memory controller. Therefore, the memory access time is the same for CPU in a same NUMA node. However, access to the memory of another NUMA node needs to pass through the router, and the data consistency guarantee needs to be provided by the cache consistency protocol, so access speed to the memory of the remote node will be slower. It can be illustrated that the NUMA architecture application performance is related to the memory allocation strategy. Access to the remote memory will increase the access delay, thereby causing the application performance degradation. In our experimental platform in this section, we use a two-way server, so it is only divided into local nodes and remote nodes under the NUMA architecture. The experiments in this section are based on server #3 in Table 3.

In order to study the memory efficiency changes of the server in NUMA, we rewrite the STREAM test benchmark so that it can determine the proportion of memory allocated to the near-end and far-end, such as 10% memory are allocated to near-end memory and 90% to far-end memory. Through different memory allocation methods, we can experiment and analyze the memory energy efficiency of the server under the NUMA architecture. This article uses 16 G, 32 G, 64 G, and 128 G different STREAM array sizes, and divides the near-end memory and the far-end memory according to the proportions of 0%, 20%, 50%, 80%, 100%, etc. Zero percent means that the whole memory is allocated to the far-end, and 100% means that the whole memory is allocated to the near-end. In the experimental results, the frequency drive of the CPU is in the on-demand mode. For the memory system, this article uses the bandwidth per watt (BpW) as shown in Equation (4) to measure the energy efficiency (MEE) of the memory system, where the power consumption of the memory system is the cumulative power consumption of all memory:

$$\text{MEE}(\text{Memory Energy Efficiency}) = \frac{Preceived\ Bandwid(MB/s\ )}{MemoryPower(watts)} \tag{4}$$

First, the memory energy efficiency data of different STREAM array sizes are obtained under the SMP architecture of the same experimental platform. As shown in Table 6, different array sizes represent different memory utilization rates, but the power consumption and energy efficiency of the memory system do not change much. Next, the BIOS in the experimental platform and the NUMA switch in the kernel are turned on to activate the NUMA architecture. Figures 15 and 16 show the power consumption and energy efficiency of the memory system with different near-end and far-end memory allocation strategies when the STREAM array size is 16 G and 128 G, respectively. As the allocation ratio between the near end and the far end is different, the power consumption of the memory system in the two NUMA nodes also changes. Furthermore, as the percentage of memory allocated to the near-end increases, the speed of operating the memory becomes faster, and the average bandwidth of the STREAM load gets higher and higher, so the energy efficiency of the memory system gradually increases.

Table 6. Experimental results of different STREAM array sizes under SMP architecture.

Size of STREAM	Memory Power Consumption (W)	MEE
16 G	28.54	341.45
32 G	28.43	342.12
64 G	29.85	331.52
128 G	29.08	340.78

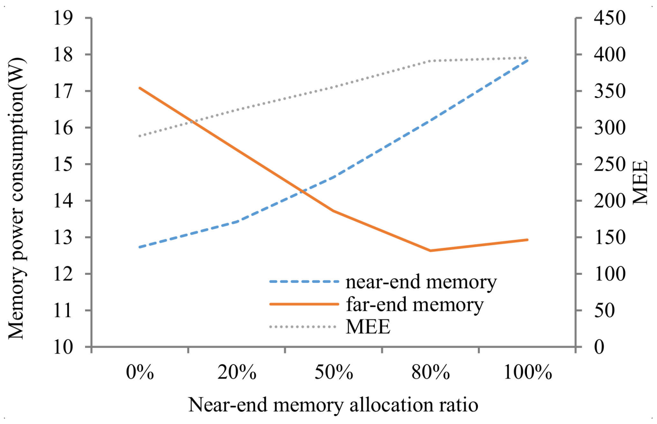

Figure 15. Experiment of singe thread STREAM array size of 16 G under NUMA architecture.

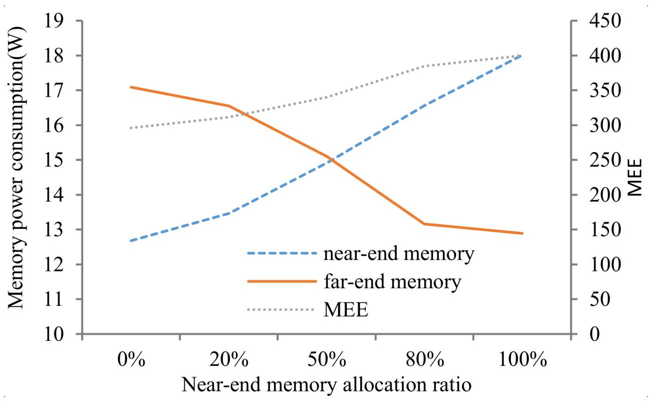

Figure 16. Experiment of singe thread STREAM array size of 128 G under NUMA architecture.

Although the allocation ratio of the memory system changes and the power consumption of the near-end memory and the far-end memory changes, the power consumption of the entire memory system (near-end memory + far-end memory) has not changed significantly, and the total power consumption of the memory is always 30 W. Further, the memory efficiency of the array size of 128 G under the same memory allocation method is slightly higher than that of 16 G. This shows that the relationship between memory power consumption and memory utilization in a memory system is not strongly related. It is not the case that the higher the memory utilization, the higher the power consumption of memory usage. Comparing the experimental results of the SMP architecture in Table 6, under the same array size of 128 G, all the memory of the NUMA architecture is allocated to the near end, and its average energy efficiency will be 16% higher than the SMP architecture. However, if all the memory is allocated to the remote end, the energy efficiency will be reduced by 13% compared to the SMP architecture. Therefore, using the NUMA architecture is beneficial to improve the energy efficiency of the memory system, but if the memory allocation method is not ideal, it will significantly affect the memory energy efficiency of the system.

In order to study the relationship between memory power consumption, energy efficiency, and the number of memory operations, we recompile the STREAM with OpenMP and set the STREAM array size to be 128 G under the NUMA architecture. Then the whole memory is allocated to near-end node and the number of parallel threads is 2, 4, 8, 16, 32, 40, 60, and 80, respectively. It should be noted that since the number of logical cores of a single CPU of the experimental platform is 40, the threads use logical threads on CPU0 if the number of parallel threads is less than 40. CPU0 and its near-end memory can be regarded as a near-end node. The memory energy efficiency improved significantly as the number of parallel threads increased. The experimental results of the NUMA architecture are shown in Figure 17.

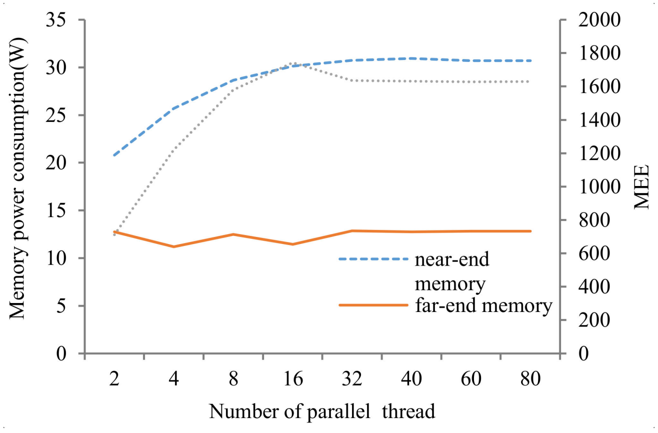

Figure 17. Experiment of multi-thread STEAM array size of 128 G under NUMA architecture.

The memory energy efficiency improved significantly as the number of parallel threads increases. From 200 of a single thread to 1700 in multi-threading, the memory power consumption in the memory system also increased significantly. In previous studies of server energy efficiency, the energy efficiency of a memory system is usually considered to be a fixed value, which is estimated as a constant power consumption. However, compared with the single-threaded experiment, we can see that the memory power consumption is not static, and as the number of parallel threads increases, the memory power consumption also increases. However, it is not the case that the higher the number of threads, the higher the energy efficiency of the memory system. When the number of parallel threads is greater than 16, the memory energy efficiency decreases slightly and tends to be stable. Furthermore, if we still increase the number of threads, the energy efficiency of the memory cannot be increased, and instead it will be degraded.

Figure 18 shows the experimental results of the multi-threaded parallel STREAM array size of 128 G under the SMP architecture. As the number of threads increases, the energy efficiency of the memory system also increases, and it will be decreased slightly and tends to be stable when the number of parallel threads is greater than 16. Besides, when the number of parallel threads is less than 16, the memory energy efficiency of SMP architecture is 40% lower than that of the NUMA architecture. In addition, when the number of parallel threads is greater than 16, the memory energy efficiency of SMP architecture is 5% higher than that of the NUMA architecture. The reason for this may be that when the number of parallel threads increases, some threads are assigned to the remote CPU in the NUMA system, resulting in remote memory access, which reduces memory access speed and memory energy efficiency greatly. Therefore, it is not suitable to use the NUMA architecture if the number of parallel threads of the application is too high. All in all, not all applications using the NUMA architecture can improve the memory energy efficiency and deciding whether to use the NUMA architecture should depend on the number of threads, the distribution of threads, and memory allocation strategies. Data center operators should focus on application types, task planning, and task placement to determine how to use NUMA architecture reasonably to improve memory efficiency.

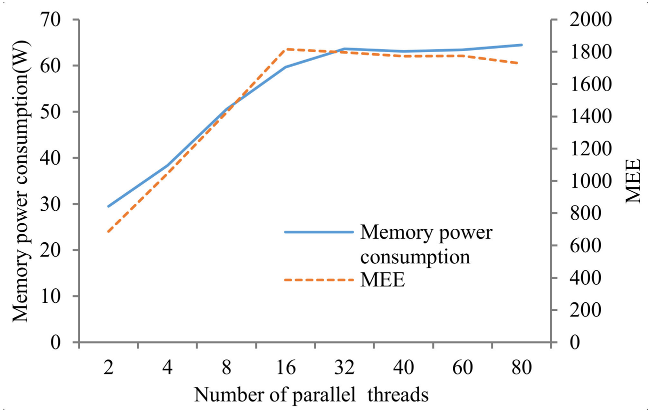

Figure 18. Experiment of multi-thread STEAM array size of 128 G under SMP architecture.

5.3. Insights on Energy Efficiency of Memory-Intensive Applications

From the above observations, we derive some insights for memory-intensive applications in data centers in terms of power and energy consumption.

Insight #1: The power consumption per percentage utilization of the server decreases as array size increases because of the reduction of the number of concurrent STREAM threads and vice versa. This indicates that power consumption of the server may be increased by multiple threaded applications.

Insight #2: Neither memory nor CPU utilization is suitable for evaluating system power con-sumption when it comes to memory-intensive applications. Thus, for large memory nodes which memory intensive applications are running on, we are supposed to con-sider more indicators rather than memory and CPU utilization when implementing power aware scheduling.

Insight #3: The reasonable use of the NUMA architecture will improve the memory energy efficiency significantly. It is necessary to ensure that most of the memory is allocated to the near-end nodes when using NUMA architecture. Otherwise, the memory energy efficiency will be lower than that of the SMP architecture. The experimental results show that it can increase the memory energy efficiency by 16% more than the SMP architecture with a reasonable use of NUMA architecture, but it decreases the memory energy efficiency by 13% than the SMP architecture with an unreasonable use of NUMA architecture.

Insight #4: We should pay attention to the tasks distribution on CPU, and the data center operators should allocate the tasks to a single CPU if possible. Besides, they would better use the near-end memory when allocating memory.

6. Conclusions

Through the evaluation of large memory systems running a memory-intensive application to understand the energy efficiency characteristics of memory system, it can help data center managers and system operators in many folds, including system capacity planning, power shifting, job placement, and scheduling. In this paper, we conducted extensive experiments and measurements to investigate the power and energy characteristics of three 2U servers running various memory-intensive benchmarks. Experiment results show that server power consumption and performance changes with hardware configuration, workload intensity, and concurrent running threads. However, fully utilized memory systems are not the most energy efficient, In addition, though the memory system is not fully utilized, application's performance and server power consumption can be impacted a lot by different powered memory modules of installed memory capacity (the memory capacity per processor core). This implications can inspire us in desing of reconfigurable system and real-time power aware adaption. We verified the effect of different memory allocation and thread count strategies on the memory efficiency of NUMA and SMP architectures. The experimental results showed that proper task placement and memory allocation were required to make full use of NUMA architectures, otherwise the system energy efficiency would be reduced.

Our findings presented in this paper provide useful insights and guidance to system designers, as well as data center operators, for energy-efficiency-aware job scheduling and energy savings. In order to ensure that NUMA architecture improves the energy efficiency of the memory system, data center operators should pay attention to application types, task planning, and placement. As for future work, we plan to take further step into the energy efficiency of large memory systems run-ning more diverse memory intensive applications, such as in-memory databases, Hadoop, and Spark jobs, so that specific configurations can be selected to improve the server's energy efficiency based on the characteristics of the upper-layer applications.

Author Contributions: Conceptualization, C.J.; Data curation, K.Z.; Formal analysis, D.O.; Investigation, Y.Q. and L.Y. All authors have read and agreed to the published version of the manuscript.

Funding: This research was funded by the National Natural Science Foundation of China, grant number 61972118, the Science and Technology Project of State Grid Corporation of China (Research and Application on Multi-Datacenters Cooperation & Intelligent Operation and Maintenance, grant number 5700–202018194A-0-0-00, and the Science and Technology Project of State Grid Corporation of China, grant number SGSDXT00DKJS1900040.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: Not applicable.

Acknowledgments: This work is supported by the National Natural Science Foundation of China (No. 61972118), the Science and Technology Project of State Grid Corporation of China (Research and Application on Multi-Datacenters Cooperation & Intelligent Operation and Maintenance, No. 5700–202018194A-0-0-00), and the Science and Technology Project of State Grid Corporation of China (No. SGSDXT00DKJS1900040).

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Reed, D.A.; Dongarra, J. Exascale Computing and Big Data. *Commun. ACM* **2015**, *58*, 56–68. [CrossRef]
2. Zaharia, M.; Chowdhury, M.; Franklin, M.J.; Shenker, S.; Stoica, I. Spark: Cluster computing with working sets. *HotCloud* **2010**, *10*, 95.
3. Ahn, J.; Hong, S.; Yoo, S.; Mutlu, O.; Choi, K. A Scalable Processing-in-Memory Accelerator for Parallel Graph Processing. In Proceedings of the 42nd Annual International Symposium on Computer Architecture (ISCA'15), Portland, OR, USA, 13–17 June 2015; Association for Computing Machinery: New York, NY, USA, 2015; pp. 105–117.
4. Hirzel, M.; Soulé, R.; Schneider, S.; Gedik, B.; Grimm, R. A Catalog of Stream Processing Optimizations. *ACM Comput. Surv.* **2014**, *46*, 1–34. [CrossRef]
5. Zhang, H.; Chen, G.; Ooi, B.C.; Tan, K.-L.; Zhang, M. In-Memory Big Data Management and Processing: A Survey. *IEEE Trans. Knowl. Data Eng.* **2015**, *27*, 1920–1948. [CrossRef]
6. Jiang, C.; Han, G.; Lin, J.; Jia, G.; Shi, W.; Wan, J. Characteristics of co-allocated online services and batch jobs in internet data centers: A case study from alibaba cloud. *IEEE Access* **2019**, *7*, 22495–22508. [CrossRef]
7. Jiang, C.; Qiu, Y.; Shi, W.; Ge, Z.; Wang, J.; Chen, S.; Cerin, C.; Ren, Z.; Xu, G.; Lin, J. Characterizing co-located workloads in alibaba cloud datacenters. *IEEE Trans. Cloud Comput.* **2020**, 1. [CrossRef]
8. Hamdioui, S.; Xie, L.; Anh, A.N.H.; Taouil, M.; Bertels, K.; Corporaal, H.; Jiao, H.; Catthoor, F.; Wouters, D.; Eike, L.; et al. Memrisor Based Computation-in-Memory Architecture for Data-Intensive Applications. In Proceedings of the 2015 Design, Automation & Test. In Europe Conference & Exhibition (DATE), Grenoble, France, 9–13 March 2015; IEEE Conference Publications: New Jersey, NJ, USA, 2015; pp. 1718–1725.
9. Nair, R.; Antao, S.F.; Bertolli, C.; Bose, P.; Brunheroto, J.R.; Chen, T.; Cher, C.-Y.; Costa, C.H.A.; Doi, J.; Evangelinos, C.; et al. Active Memory Cube: A Processing-in-Memory Architecture for Exascale Systems. *IBM J. Res. Dev.* **2015**, *59*, 17:1–17:14. [CrossRef]
10. Pugsley, S.H.; Jestes, J.; Balasubramonian, R.; Srinivasan, V.; Buyuktosunoglu, A.; Davis, A.; Li, F. Comparing Implementations of Near-Data Computing with in-Memory MapReduce Workloads. *IEEE Micro* **2014**, *34*, 44–52. [CrossRef]

11. Tanabe, N.; Nuttapon, B.; Nakajo, H.; Ogawa, Y.; Kogou, J.; Takata, M.; Joe, K. A Memory Accelerator with Gather Functions for Bandwidth-Bound Irregular Applications. In Proceedings of the First Workshop on Irregular Applications: Architectures and Algorithm (IAAA'11), Seattle, WA, USA, 13 November 2011; Association for Computing Machinery: New York, NY, USA, 2011; pp. 35–42.
12. Pawlowski, J.T. Hybrid Memory Cube (HMC). In Proceedings of the 2011 IEEE Hot Chips 23 Symposium (HCS), Stanford, CA, USA, 17–19 August 2011; pp. 1–24.
13. Wang, Y.; Yu, H. An Ultralow-Power Memory-Based Big-Data Computing Platform by Nonvolatile Domain-Wall Nanowire Devices. In Proceedings of the International Symposium on Low Power Electronics and Design (ISLPED), Beijing, China, 4–6 September 2013; pp. 329–334.
14. Weis, C.; Loi, I.; Benini, L.; Wehn, N. An Energy Efficient DRAM Subsystem for 3D Integrated SoCs. In Proceedings of the 2012 Design, Automation & Test in Europe Conference & Exhibition (DATE), Dresden, Germany, 12–16 March 2012; pp. 1138–1141.
15. Hajkazemi, M.H.; Chorney, M.; Jabbarvand Behrouz, R.; Khavari Tavana, M.; Homayoun, H. Adaptive Bandwidth Management for Performance-Temperature Trade-Offs in Heterogeneous HMC + DDRx Memory. In Proceedings of the 25th Edition on Great Lakes Symposium on VLSI (GLSVLSI'15), Pittsburgh, PA, USA, 20–22 May 2015; Association for Computing Machinery: New York, NY, USA, 2015; pp. 391–396.
16. Goswami, N.; Cao, B.; Li, T. Power-Performance Co-Optimization of Throughput Core Architecture Using Resistive Memory. In Proceedings of the 2013 IEEE 19th International Symposium on High Performance Computer Architecture (HPCA), Shenzhen, China, 23–27 February 2013; pp. 342–353.
17. Sharad, M.; Fan, D.; Roy, K. Ultra Low Power Associative Computing with Spin Neurons and Resistive Crossbar Memory. In Proceedings of the 50th Annual Design Automation Conference (DAC'13), Austin, TX, USA, 29 May–7 June 2013; Association for Computing Machinery: New York, NY, USA, 2013; pp. 1–6.
18. Imani, M.; Mercati, P.; Rosing, T. ReMAM: Low Energy Resistive Multi-Stage Associative Memory for Energy Efficient Computing. In Proceedings of the 2016 17th International Symposium on Quality Electronic Design (ISQED), Santa Clara, CA, USA, 15–16 March 2016; pp. 101–106.
19. Ahn, J.; Yoo, S.; Choi, K. Low-Power Hybrid Memory Cubes with Link Power Management and Two-Level Prefetching. *IEEE Trans. Very Large Scale Integr. VLSI Syst.* **2016**, *24*, 453–464. [CrossRef]
20. Islam, N.S.; Wasi-ur-Rahman, M.; Lu, X.; Shankar, D.; Panda, D.K. Performance Characterization and Acceleration of In-Memory File Systems for Hadoop and Spark Applications on HPC Clusters. In Proceedings of the 2015 IEEE International Conference on Big Data (Big Data), Santa Clara, CA, USA, 29 October–1 November 2015; IEEE: Santa Clara, CA, USA, 2015; pp. 243–252.
21. Paraskevas, K.; Attwood, A.; Luján, M.; Goodacre, J. Scaling the Capacity of Memory Systems; Evolution and Key Approaches. In Proceedings of the International Symposium on Memory Systems (MEMSYS'19), Washington, DC, USA, 30 September–3 October 2019; Association for Computing Machinery: New York, NY, USA, 2019; pp. 235–249.
22. Jiang, C.; Wang, Y.; Ou, D.; Luo, B.; Shi, W. Energy Proportional Servers: Where Are We in 2016? In Proceedings of the 2017 IEEE 37th International Conference on Distributed Computing Systems (ICDCS), Atlanta, GA, USA, 5–8 June 2017; IEEE: Atlanta, GA, USA, 2017; pp. 1649–1660.
23. Qiu, Y.; Jiang, C.; Wang, Y.; Ou, D.; Li, Y.; Wan, J. Energy Aware virtual machine scheduling in data centers. *Energies* **2019**, *12*, 646. [CrossRef]
24. SPECpower_ssj$^{®}$. 2008. Available online: https://www.spec.org/power_ssj2008/ (accessed on 1 May 2021).
25. Jiang, C.; Fan, T.; Gao, H.; Shi, W.; Liu, L.; Cérin, C.; Wan, J. Energy aware edge computing: A survey. *Comput. Commun.* **2020**, *151*, 556–580. [CrossRef]
26. Jiang, C.; Wang, Y.; Ou, D.; Qiu, Y.; Li, Y.; Wan, J.; Luo, B.; Shi, W.; Cerin, C. EASE: Energy efficiency and proportionality aware virtual machine scheduling. In Proceedings of the 2018 30th International Symposium on Computer Architecture and High Performance Computing (SBAC-PAD), Lyon, France, 24–27 September 2018; pp. 65–68.
27. Islam, M.; Scrbak, M.; Kavi, K.M.; Ignatowski, M.; Jayasena, N. Improving Node-Level MapReduce Performance Using Processing-in-Memory Technologies. In *Lecture Notes in Computer Science*; Springer International Publishing: Cham, Switzerland, 2014; pp. 425–437.
28. Pattnaik, A.; Tang, X.; Jog, A.; Kayiran, O.; Mishra, A.K.; Kandemir, M.T.; Mutlu, O.; Das, C.R. Scheduling Techniques for GPU Architectures with Processing-in-Memory Capabilities. In Proceedings of the 2016 International Conference on Parallel Architectures and Compilation (PACT'16), Haifa, Israel, 11–15 September 2016; Association for Computing Machinery: New York, NY, USA, 2016; pp. 31–44.
29. Li, S.; Reddy, D.; Jacob, B. A Performance & Power Comparison of Modern High-Speed DRAM Architectures. In Proceedings of the International Symposium on Memory Systems (MEMSYS '18), Alexandria, VA, USA, 1–4 October 2018; Association for Computing Machinery: New York, NY, USA, 2018; pp. 341–353.
30. Asghari-Moghaddam, H.; Son, Y.H.; Ahn, J.H.; Kim, N.S. Chameleon: Versatile and Practical near-DRAM Acceleration Architecture for Large Memory Systems. In Proceedings of the 2016 49th Annual IEEE/ACM International Symposium on Microarchitecture (MICRO), Taipei, Taiwan, 15–19 October 2016; pp. 1–13.

31. Zhang, D.; Jayasena, N.; Lyashevsky, A.; Greathouse, J.L.; Xu, L.; Ignatowski, M. TOP-PIM: Throughput-Oriented Programmable Processing in Memory. In Proceedings of the 23rd International Symposium on High-Performance Parallel and Distributed Computing (HPDC'14), Vancouver, BC, Canada, 23–27 June 2014; Association for Computing Machinery: New York, NY, USA, 2014; pp. 85–98.
32. Xi, S.L.; Babarinsa, O.; Athanassoulis, M.; Idreos, S. Beyond the Wall: Near-Data Processing for Databases. In Proceedings of the 11th International Workshop on Data Management on New Hardware (DaMoN'15), Melbourne, Australia, 31 May–4 June 2015; Association for Computing Machinery: New York, NY, USA, 2015; pp. 1–10.
33. Keeton, K. Memory-Driven Computing. In Proceedings of the 15th USENIX Conference on File and Storage Technologies (FAST), Santa Clara, CA, USA, 27 February–2 March 2017; USENIX Association: Santa Clara, CA, USA, 2017.
34. Imani, M.; Gupta, S.; Rosing, T. Digital-Based Processing in-Memory: A Highly-Parallel Accelerator for Data Intensive Applications. In Proceedings of the International Symposium on Memory Systems (MEMSYS'19), Washington DC, USA, 30 September 2019—3 October 2019; Association for Computing Machinery: New York, NY, USA, 2019; pp. 38–40.
35. Azarkhish, E.; Pfister, C.; Rossi, D.; Loi, I.; Benini, L. Logic-Base Interconnect Design for near Memory Computing in the Smart Memory Cube. *IEEE Trans. Very Large Scale Integr. VLSI Syst.* **2017**, *25*, 210–223. [CrossRef]
36. Sura, Z.; Jacob, A.; Chen, T.; Rosenburg, B.; Sallenave, O.; Bertolli, C.; Antao, S.; Brunheroto, J.; Park, Y.; O'Brien, K.; et al. Data Access Optimization in a Processing-in-Memory System. In Proceedings of the 12th ACM International Conference on Computing Frontiers (CF '15), Ischia, Italy, 18–21 May 2015; Association for Computing Machinery: New York, NY, USA, 2015; pp. 1–8.
37. Yun, H.; Mancuso, R.; Wu, Z.-P.; Pellizzoni, R. PALLOC: DRAM Bank-Aware Memory Allocator for Performance Isolation on Multicore Platforms. In Proceedings of the 2014 IEEE 19th Real-Time and Embedded Technology and Applications Symposium (RTAS), Berlin, Germany, 15–17 April 2014; pp. 155–166.
38. Ahmed, A.; Skadron, K. Hopscotch: A Micro-Benchmark Suite for Memory Performance Evaluation. In Proceedings of the International Symposium on Memory Systems (MEMSYS'19); Washington, DC, USA, 30 September–3 October 2019; Association for Computing Machinery: New York, NY, USA, 2019; pp. 167–172.
39. Patil, O.; Ionkov, L.; Lee, J.; Mueller, F.; Lang, M. Performance Characterization of a DRAM-NVM Hybrid Memory Architecture for HPC Applications Using Intel Optane DC Persistent Memory Modules. In Proceedings of the International Symposium on Memory Systems (MEMSYS'19), Washington, DC, USA, 30 September–3 October 2019; Association for Computing Machinery: New York, NY, USA, 2019; pp. 288–303.
40. Chishti, Z.; Akin, B. Memory System Characterization of Deep Learning Workloads. In Proceedings of the International Symposium on Memory Systems (MEMSYS'19), Washington, DC, USA, 30 September–3 October 2019; Association for Computing Machinery: New York, NY, USA, 2019; pp. 497–505.
41. Liu, J.; Zhao, H.; Ogleari, M.A.; Li, D.; Zhao, J. Processing-in-Memory for Energy-Efficient Neural Network Training: A Heterogeneous Approach. In Proceedings of the 2018 51st Annual IEEE/ACM International Symposium on Microarchitecture (MICRO), Fukuoka, Japan, 20–24 October 2018; pp. 655–668.
42. Dasari, D.; Nelis, V.; Akesson, B. A Framework for Memory Contention Analysis in Multi-Core Platforms. *Real-Time Syst.* **2016**, *52*, 272–322. [CrossRef]
43. Kim, Y.; Han, D.; Mutlu, O.; Harchol-Balter, M. ATLAS: A Scalable and High-Performance Scheduling Algorithm for Multiple Memory Controllers. In Proceedings of the HPCA—16 2010 The Sixteenth International Symposium on High-Performance Computer Architecture, Bangalore, India, 9–14 January 2010; pp. 1–12.
44. Muralidhara, S.P.; Subramanian, L.; Mutlu, O.; Kandemir, M.; Moscibroda, T. Reducing Memory Interference in Multicore Systems via Application-Aware Memory Channel Partitioning. In Proceedings of the 44th Annual IEEE/ACM International Symposium on Microarchitecture (MICRO-44), Porto Alegre, Brazil, 3–7 December 2011; Association for Computing Machinery: New York, NY, USA, 2011; pp. 374–385.
45. Mutlu, O.; Moscibroda, T. Stall-Time Fair Memory Access Scheduling for Chip Multiprocessors. In Proceedings of the 40th Annual IEEE/ACM International Symposium on Microarchitecture (MICRO 2007), Chicago, IL, USA, 1–5 December 2007; pp. 146–160.
46. Mutlu, O.; Moscibroda, T. Parallelism-Aware Batch Scheduling: Enhancing Both Performance and Fairness of Shared DRAM Systems. In Proceedings of the 2008 International Symposium on Computer Architecture, Beijing, China, 21–25 June 2008; pp. 63–74.
47. Kim, H.; de Niz, D.; Andersson, B.; Klein, M.; Mutlu, O.; Rajkumar, R. Bounding Memory Interference Delay in COTS-Based Multi-Core Systems. In Proceedings of the 2014 IEEE 19th Real-Time and Embedded Technology and Applications Symposium (RTAS), Berlin, Germany, 15–17 April 2014; pp. 145–154.
48. Zhu, H.; Erez, M. Dirigent: Enforcing QoS for Latency-Critical Tasks on Shared Multicore Systems. In Proceedings of the Twenty-First International Conference on Architectural Support for Programming Languages and Operating Systems (ASPLOS'16), Atlanta, GA, USA, 2–6 April 2016; Association for Computing Machinery: New York, NY, USA, 2016; pp. 33–47.
49. Jeong, M.K.; Erez, M.; Sudanthi, C.; Paver, N. A QoS-aware memory controller for dynamically balancing GPU and CPU bandwidth use in an MPSoC. In Proceedings of the 49th Annual Design Automation Conference (DAC), San Francisco, CA, USA, 3–7 June 2012; pp. 850–855.
50. Yu, J.; Nane, R.; Ashraf, I.; Taouil, M.; Hamdioui, S.; Corporaal, H.; Bertels, K. Skeleton-Based Synthesis Flow for Computation-in-Memory Architectures. *IEEE Trans. Emerg. Top. Comput.* **2020**, *8*, 545–558. [CrossRef]

51. Li, H.; Ghodsi, A.; Zaharia, M.; Shenker, S.; Stoica, I. Tachyon: Reliable, Memory Speed Storage for Cluster Computing Frameworks. In Proceedings of the ACM Symposium on Cloud Computing (SOCC'14), Seattle, WA, USA, 3–5 November 2014; ACM: New York, NY, USA, 2014; pp. 1–15.
52. Zhao, D.; Zhang, Z.; Zhou, X.; Li, T.; Wang, K.; Kimpe, D.; Carns, P.; Ross, R.; Raicu, I. FusionFS: Toward Supporting Data-Intensive Scientific Applications on Extreme-Scale High-Performance Computing Systems. In Proceedings of the 2014 IEEE International Conference on Big Data (Big Data), Washington, DC, USA, 27–30 October 2014; pp. 61–70.
53. Jiang, D.; Chen, G.; Ooi, B.C.; Tan, K.-L.; Wu, S. EpiC: An Extensible and Scalable System for Processing Big Data. *Proc. VLDB Endow.* **2014**, *7*, 541–552. [CrossRef]
54. Imani, M.; Gupta, S.; Kim, Y.; Zhou, M.; Rosing, T. DigitalPIM: Digital-Based Processing in-Memory for Big Data Acceleration. In Proceedings of the 2019 on Great Lakes Symposium on VLSI (GLSVLSI'19), Tysons Corner, VA, USA, 9–11 May 2019; Association for Computing Machinery: New York, NY, USA, 2019; pp. 429–434.
55. Zhang, C.; Meng, T.; Sun, G. PM3: Power Modeling and Power Management for Processing-in-Memory. In Proceedings of the 2018 IEEE International Symposium on High Performance Computer Architecture (HPCA), Vienna, Austria, 24–28 February 2018; pp. 558–570.
56. Tan, K.-L.; Cai, Q.; Ooi, B.C.; Wong, W.-F.; Yao, C.; Zhang, H. In-Memory Databases: Challenges and Opportunities from Software and Hardware Perspectives. *SIGMOD Rec.* **2015**, *44*, 35–40. [CrossRef]
57. Makrani, H.M.; Sayadi, H.; Dinakarra, S.M.P.; Rafatirad, S.; Homayoun, H. A Comprehensive Memory Analysis of Data Intensive Workloads on Server Class Architecture. In Proceedings of the International Symposium on Memory Systems (MEMSYS'18), Alexandria, VA, USA, 1–4 October 2018; Association for Computing Machinery: New York, NY, USA, 2018; pp. 19–30.
58. Wang, H.; Singh, R.; Schulte, M.J.; Kim, N.S. Memory Scheduling towards High-Throughput Cooperative Heterogeneous Computing. In Proceedings of the 23rd International Conference on Parallel Architectures and Compilation (PACT'14), Edmonton, AB, Canada, 24–27 August 2014; Association for Computing Machinery: New York, NY, USA, 2014; pp. 331–342.
59. Jiang, C.; Wang, Y.; Ou, D.; Li, Y.; Zhang, J.; Wan, J.; Luo, B.; Shi, W. Energy Efficiency Comparison of Hypervisors. *Sustain. Comput. Inform. Syst.* **2019**, *22*, 311–321. [CrossRef]
60. Yun, H.; Yao, G.; Pellizzoni, R.; Caccamo, M.; Sha, L. MemGuard: Memory Bandwidth Reservation System for Efficient Performance Isolation in Multi-Core Platforms. In Proceedings of the 2013 IEEE 19th Real-Time and Embedded Technology and Applications Symposium (RTAS), Philadelphia, PA, USA, 9–11 April 2013; pp. 55–64.
61. Zhang, D.P.; Jayasena, N.; Lyashevsky, A.; Greathouse, J.; Meswani, M.; Nutter, M.; Ignatowski, M. A New Perspective on Processing-in-Memory Architecture Design. In Proceedings of the ACM SIGPLAN Workshop on Memory Systems Performance and Correctness (MSPC'13), Seattle, WA, USA, 16–19 June 2013; Association for Computing Machinery: New York, NY, USA, 2013; pp. 1–3.
62. Lee, S.; Kim, N.S.; Kim, D. Exploiting OS-Level Memory Offlining for DRAM Power Management. *IEEE Comput. Arch. Lett.* **2019**, *18*, 141–144. [CrossRef]
63. Gray, L.D.; Kumar, A.; Li, H.H. Workload Characterization of the SPECpower_ssj2008 Benchmark. In *Performance Evaluation: Metrics, Models and Benchmarks*; Springer: Berlin/Heidelberg, Germany, 2008; pp. 262–282.
64. Ryckbosch, F.; Polfliet, S.; Eeckhout, L. Trends in Server Energy Proportionality. *Comput. Long Beach Calif.* **2011**, *44*, 69–72. [CrossRef]
65. Memory Bandwidth: Stream Benchmark Performance Results. Available online: https://www.cs.virginia.edu/stream/ (accessed on 1 May 2021).
66. CloudSuite. Available online: http://cloudsuite.ch/ (accessed on 1 May 2021).
67. NAMD—Scalable Molecular Dynamics. Available online: https://www.ks.uiuc.edu/Research/namd/ (accessed on 1 May 2021).