A Limits Study of Benefits from Nanostore-Based Future Data-Centric System Architectures

Jichuan Chang[*] Parthasarathy Ranganathan[*] Trevor Mudge[†]
David Roberts[‡] Mehul A. Shah[#] Kevin T. Lim[*]

HP Labs[*], University of Michigan[†], Micron[‡] and Nou Data[#]

ABSTRACT

The adoption of non-volatile memories (NVMs) in system architecture and the growth in data-centric workloads offer exciting opportunities for new designs. In this paper, we examine the potential and limit of designs that move compute in close proximity to NVM-based data stores. To address the challenges in evaluating such system architectures for distributed systems, we develop and validate a new methodology for large-scale data-centric workloads. We then study "nanostores" as an example design that constructs distributed systems from building blocks with 3D-stacked compute and NVM layers on the same chip, replacing both traditional storage and memory with NVM. Our limits study demonstrates significant potential of this approach (3-162X improvement in energy delay product) over 2015 baselines, particularly for IO-intensive workloads. We also discuss and quantify the impact of network bandwidth, software scalability, and power density, and design tradeoffs for future NVM-based data-centric architectures.

Categories and Subject Descriptors

C.4 [**Performance of Systems**];
C.5.5 [**Computer System Implementation**] Servers

General Terms

Design, Measurement, Performance

Keywords

Limits study, system architectures, Nanostores, non-volatile memory, data-centric data centers

1. INTRODUCTION

The amount of data being created in both enterprise and scientific communities is exploding, growing significantly faster than Moore's Law. The size of online data is estimated to have risen nearly 60-fold in the last seven years [1]. Similarly, many scientific disciplines are experiencing a paradigm shift towards data-driven discovery. The annual doubling of gene databases and the petabyte per second produced by the Large Hadron Collider [25] are a few examples representing the unprecedented data challenges for scientists. Richer sensors, digitization of analog content, and new applications like Twitter, web search, etc., will only increase data growth rates. Indeed, it is estimated that only 5% of the world's offline data has been made online so far.

CF'12, May 15–17, 2012, Cagliari, Italy.

The growth in data is leading to a corresponding growth in data-centric applications that operate on data in diverse ways (capture, classify, analyze, process, archive, etc.). Compared to traditional enterprise workloads (e.g., online transaction processing, web serving), emerging data-centric workloads significantly change assumptions about system design. These workloads typically operate at larger scale (up to hundreds of thousands of servers) and on more diverse data (e.g., structured, unstructured, rich media) with I/O intensive, sometimes random, data access patterns and limited locality. In addition, recent software innovations aimed at improving scalability with commodity hardware (e.g., MapReduce [9]) allow such workloads to be implemented and executed on large-scale distributed systems. This evolution of both workload and software support suggests a re-evaluation of system architectures in the datacenters.

At the same time, non-volatile memories (NVMs) are also challenging many traditional assumptions in system design. Flash memories are already pervasive in popular consumer devices (e.g., smart phones) and are becoming important in the enterprise as well (e.g., Fusion-IO, EMC). New NVMs such as phase-change RAM (PCM) or memristors offer significantly better latencies and energy efficiencies than Flash, enabling a new layer in the storage hierarchy. Looking further ahead, these NVMs are predicted to have comparable performance to DRAM, and with better energy efficiency and device scalability [4, 18].

Both NVMs as new architectural building blocks and the growth in data-centric workloads offer an interesting opportunity for new system architectures. We classify recent proposals that incorporate NVM into the system architecture, and identify a trend towards tighter integration of compute with NVM-based persistent data stores [2][7][33][34][35] (Section 2). Extrapolating this trend points to a future where data-centric systems can be built using compute packaged with persistent NVM data stores that are accessed in memory-based interfaces. However, the design space that realizes this vision has not been fully explored. Similarly, there has been inadequate emphasis on distributed systems in prior proposals.

A key challenge in addressing these questions is the lack of an evaluation methodology for NVM-based designs in large-scale, data-centric system architectures. We address this challenge by developing (and validating) a hybrid analytical and simulation based methodology to evaluate such architectures, using benchmarks representing data-centric applications, and carefully choosing baseline architectures and configurations optimized for each benchmark (Section 3).

We focus on an illustrative design point, "nanostores": an approach to build large-scale distributed systems from a new building block that stacks compute and NVM on the same chip and uses NVM to replace traditional disks as well as main

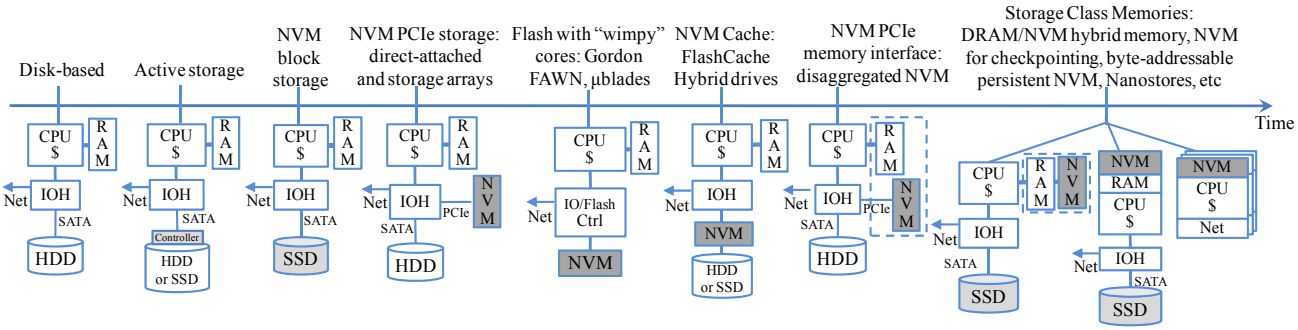


Figure 1: System architectures exploiting NVM

memory. Using carefully chosen architecture parameters for the baseline and proposed designs, our evaluation results (Section 4) indicate significant potential for nanostores, with 3-162X improvements in energy delay product (EDP) for our benchmarks, translating to improved performance as well as better energy efficiency. Concurrently, these results also indicate important future challenges in scaling software, power density, and network bandwidth. Specifically, our design space exploration identifies interesting new tradeoffs and benefits from balanced system designs centered on compute and network provisioning. Using these insights, we discuss other architecture issues and implications for using non-volatile memories (Section 5). Sections 6 and 7 additionally discuss related work and conclusions.

2. BACKGROUND: NVM-BASED DATA-CENTRIC SYSTEM ARCHITECTURES

Figure 1 presents a summary of various current and proposed designs that integrate non-volatile memories into system architectures. Several existing products (e.g., EMC, Fusion-IO, HP, Oracle, Seagate, TMS) expose Flash memories as block devices either through SAS/SATA or PCI Express (PCIe) interfaces, and use them as disk replacements or disk caches, with appropriate software support (e.g., Fusion-IO drivers, Oracle ASM, Facebook Flashcache). Recent research proposals (microblades [18], FAWN [5], Gordon [6]) have additionally noted that coupling such Flash-based storage with "wimpy" lower-power processors can lead to more balanced system designs for cloud workloads, significantly improving energy efficiency. Several research proposals have also discussed DRAM/NVM hybrid memory and NVM as byte-addressable memory devices connected to the memory bus or 3D-stacked on the chip, but using NVM as additional levels in the memory/storage hierarchy and for single-node workloads. These designs do not exploit the non-volatility of NVMs. More recent proposals have begun to examine software support for using NVM as a byte-addressable persistent data store (e.g., BPFS [33], CDDS [34], NV-Heaps [35] and Mnemosyne [7]), and leverage NVM-based data stores in future distributed architectures (Nanostores [2]).

Classifying these designs across two dimensions – (1) the type of NVM interface exposed and (2) the proximity of NVM to the compute resources – and tracing the timeline of design evolution reveals an interesting trend: future designs are ultimately likely to use NVM as the primary data store (merging both memory and storage) and access the NVM through high-bandwidth memory-based interfaces.

The rightmost design in Figure 1 illustrates one such system architecture that we focus on in this paper. In this "nanostore" design [2], the NVM is 3D-stacked on the chip with the processor, but is used as a single-level, memory-addressed persistent data store replacing the function of traditional main memory as well as disk drives. The two most important aspects of this approach are: (1) the co-location of power-efficient computing with NVM-based data store, and (2) the use of nanostore arrays to build highly-parallel, large-scale distributed systems.

Exploring this architecture invariably identifies open questions representative of the broader design space for future NVM-based data-centric system architectures. How do these new designs compare to aggressive extrapolations of existing architectures? How do the benefits change across the range of data-centric workloads? Do we need to rethink the balance of compute, data, and network for this new architecture for large systems? What are the implications of specific design choices and technology extrapolations, including network bandwidth and packaging assumptions? This paper addresses these questions.

3. EXPLORATION METHODOLOGY

A key challenge in answering these questions is the lack of an evaluation methodology for full-system, distributed architectures with future technology and workloads. Specifically, we need to study large-scale clusters running distributed workloads operating on high volumes of data. We also need to examine tradeoffs at the full system level and model the interactions of compute, memory, storage, and networking subsystems. Conventional architecture simulators cannot model this level of scale and scope. There is also a combinatorial explosion in the design space from various fine-grained and coarse-grained architecture-level options, as well as the choice of technology and workload parameters. To address these challenges, we develop and validate a new evaluation methodology based on hybrid performance models and new data-centric benchmarks.

3.1 Evaluation models and validation

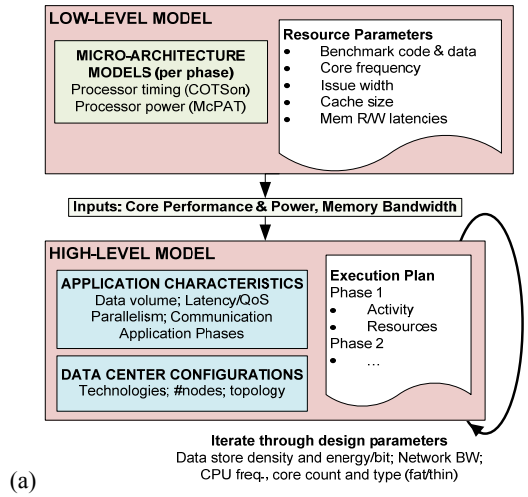
Evaluation methodology

Our evaluation methodology focuses on the application's system-level behavior and allows exploration of broader datacenter issues such as scalability and compute-network-I/O balance. The methodology adopts approaches currently used for database query planning (e.g., [26]) and MapReduce/Hadoop simulation (e.g., [27]). Similar to these approaches, we propose a high-level model using the application's execution template to break down an application's execution into consecutive phases. Each phase consists of compute, network, and I/O subsystem activities, performed sequentially or in parallel. The high-level model takes inputs from low-level performance and power models regarding the performance and usage of compute, memory, I/O and network subsystems to calculate the power and execution time for each activity, phase, and subsequently the whole application.

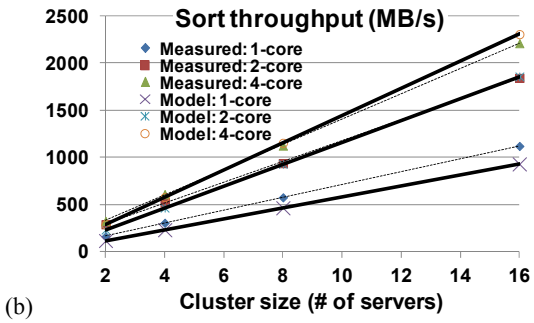
Figure 2(a) summarizes the model, while additional details and examples are presented later in Section 3.2 and Appendix A1. Specifically, the I/O (data store) and network subsystem performance models calculate the execution time for storage access and communication activities as the ratio between (1) the amount of data need to be transferred to and from the I/O and network subsystems for each activity and (2) the provisioned I/O and network bandwidths. For data stores, we model the combined bandwidth needs of both file and memory accesses in our nanostore designs, and calculate read and write time separately.

To model the performance of the compute/memory subsystem, we first run the workloads (discussed next) on an existing Xeon server configured with minimal storage and network overhead (to isolate the performance impact of network and I/O). We measure the workload's performance with two metrics: (1) the application-level data processing throughput, e.g., in terms of MB/s of data sorted or video transcoded by the compute; and (2) the workload's IPC on this Xeon server. We refer to these two numbers as the workload's base compute throughput and base IPC, respectively. We can then calculate the workload's compute throughput on a simulated processor configuration as the base compute throughput scaled by the ratio of the simulated IPC over the base IPC.

Using COTSon [3] for detailed, micro-architectural simulation, this methodology allows us to model processors with different microarchitectural and architectural configurations. The calculated compute throughput and other simulation outputs such as consumed memory bandwidths are fed into the higher-level model for full-system performance and power estimation.



(a)



(b)

Figure 2: Design exploration model and validation. The method in (a) combines application and subsystem level models and simulation results to reason about architecture tradeoffs; (b) presents validation results for an illustrative experiment for *sort*.

For power modeling, we focus mainly on average power consumption (for both active and static power). We also consider peak power, but only to verify compliance with power/thermal density constraints. We use the performance model to compute the utilizations for processor, memory, data store and network. Active power is assumed to scale linearly with utilization. We use the corresponding utilization factors to derate active power based on the component's peak and idle power. We also model static power when appropriate, e.g., for CPU and DRAM leakage power.

During execution phases where the CPU cores are active, we use the read and write memory bandwidth statistics from COTSon to calculate memory utilization. The amortized per-node network power is modeled as its NIC-level power scaled by a network hop multiplier, which corresponds to the average network hops traversed by a packet. Such a multiplier accounts for switch power consumed for packet routing and forwarding. The CPU peak power is determined using McPAT [17], as a function of issue width, frequency and cache size. CPU configurations at each frequency also factor in the power benefits from voltage and frequency scaling supported by McPAT. CPU static power is scaled based on the number of cores and their caches sizes.

Overall, this model provides both sufficient details and a simple way to iterate between various compute, storage I/O, and network options in order to select the optimal designs for a given objective function (e.g., EDP, energy efficiency, or performance).

Model validation

Figure 2(b) presents results validating our model for an MPI-based implementation for one of our benchmark (*sort*), on small cluster sizes ranging from 2 to 16 servers. As discussed later in Section 3.2, *sort* has one of the most complex execution template with multiple phases and concurrent compute, I/O and network activities, which stresses the validation of our approach. Each server in the cluster has 2 dual-core 2.4GHz AMD Opteron processors and a 1Gb/s Ethernet NIC. We use the traffic shaping tool `wondershaper` (http://lartc.org/wondershaper/) to vary the network bandwidths. Figure 2(b) shows the *measured* and *modeled* system-level sort throughput for different cluster sizes and number of cores per server. The predicted performance from our model tracks the real system fairly well. Note that the discrepancies mainly come from 1-core *sort*, where the real-system throughput is higher because the single core can "unfairly" use the entire shared cache and memory bandwidth.

While the model addresses all the compute, I/O, and network components of the system, and provides a powerful way to systematically explore the large design space at practical simulation times, a few caveats need to be noted. First, we assume that computation and network communication can overlap and are purely bandwidth based (i.e., no queuing models are used). These assumptions are acceptable for the distinct phases and coarse-grained data transfer behavior of our benchmarks (and many important real-world applications), but care needs to be exercised in extrapolating the model to other workloads. Notice we also assume data are distributed uniformly, and load-imbalance or skews are not considered in the model (we study the impact of relaxing these assumptions in Section 4.3).

3.2 Data-centric benchmarks

The space of data-centric workloads is vast, fast-evolving, and characterized by diversity across multiple dimensions. To study a subset of workloads for coverage and representativeness, we systematically classify data-centric workloads to characterize the key dimensions of diversity and pick a subset of workloads that

Response Time	Real-time	Real-time or interactive responses required
	Background	Response time is not critical for user needs
Access Pattern	Random	Unpredictable access to regions of data store
	Sequential	Sequential access of data chunks
	Permutation	Data is re-distributed across the system
Working Set	All	The entire dataset is accessed
	Partial	Only a subset of data is accessed
Data Type	Structured	Metadata/schema/type are used for data records
	Unstructured	No explicit data structure, e.g., text/binary files
	Rich media	Audio/video and image data with inherent structures and specific processing algorithms
Read vs. Write	Read heavy	Data reads are significant for processing
	Write heavy	Data writes are significant for processing
Processing Complexity	High	Complex processing of data is required per data item. Examples: video trans-coding, classification, prediction
	Low	Dominated by data access with low compute ratio. Examples: sort, upload, download, filtering, and aggregation.

Figure 3: Classification of data-centric workloads, used to guide our selection of benchmarks.

exercises all these dimensions. Figure 3 illustrates the taxonomy and attributes of each dimension. Considered dimensions include: response time (real-time vs. background), access pattern (random, sequential or permutation), working set (all vs. partial), data type (structured, unstructured and rich media), read vs. write heavy accesses, and processing complexity (low, medium or high).

Based on this analysis, we choose five representative benchmarks. *Sort*, *cksum* (checksum calculation in data deduplication), and *video* (video transcoding) represent I/O intensive data-centric workloads, an important focus for this study. Additionally, we study *search* and *recommender* (collaborative filtering based recommendation) to represent emerging in-memory data-centric workloads. Within these broad categories, the individual workloads provide coverage of different requirements. Taking processing complexity as an example dimension, *video* requires more computation per unit of data compared to *sort,* which needs more computation relative to *cksum*. Similarly, *recommender* requires more computation than *search*. These workloads also stress different data access patterns: random versus sequential; and different data types: structured, unstructured, and rich media. They also have publicly available implementations that we can use for simulations.

Below we describe these benchmarks and summarize their execution plans (detailed execution plans and performance/power models are presented in Appendix A1).

Sort: This benchmark implements a distributed sort of key-value records. *Sort* is read/write-heavy, and stresses the balance between compute/storage/network subsystems. We model a data-parallel algorithm with two phases: (i) A *shuffle* phase where each server reads records from local storage and, based on the key values, sends them over the network to their destinations. While incoming records fill its buffer, the server sorts the buffer and outputs to storage. (ii) A *local merge* phase after the shuffle, where each server reads previously sorted small files from local storage, performs a merge sort, and outputs the final sort results. We use `nsort` (http://www.ordinal.com) to model the local merge phase.

Checksum in data de-duplication: The benchmark *cksum* models the checksum calculation task in de-duplication, resulting in mostly read-only sequential accesses and low processing complexity. We model a parallel implementation of *cksum* where each server scans its local files to generate block or file signatures using the SHA-1 hash function. We use the Linux utility `sha1sum` in our simulations.

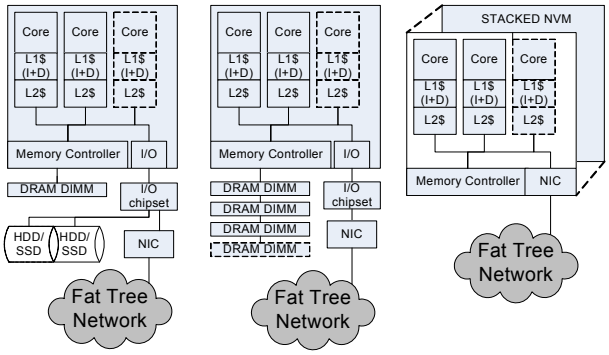


Figure 4: System architectures studied: HDD/SSD-based and memory-based baselines vs. Nanostores (right)

Processor	Baseline	Nanostore
Core count	32	1-128
Frequency (GHz)	2	0.1-2.0
Issue width	4	2, 4
Per-core L1 cache	64K+64K	64K+64K
Per-core L2 cache	1M	512K, 1M
Peak Power/Core (W)	1.83	(model)
Idle Power/Core (W)	0.04	(model)

Network	Baseline	Nanostore
Peak BW/Port (Gbit/s)	40	40
Peak Power/Port (W)	10	10
Idle Power/Port (W)	2	2

Main Memory	Baseline	Nanostore
Peak BW/Unit (GB/s)	25.6	32
Capacity/Unit (GB)	16	25
Peak Power/Unit (W)	10	0.6
Idle Power/Unit (W)	2	0

Hard Disk/SSD	HDD	SSD
Peak BW/Drive (GB/s	0.5	4.5
Capacity/Drive (TB)	6	1.2
Peak Power/Drive (W	10	10
Idle Power/Drive (W)	8	1

Figure 5: Evaluation methodology and parameter details. Notes: (1) Baseline CPU configuration extrapolated from Xeon E7500; (2) per-core cache capacity lowered to model many-core and ultra-low voltage cores; (3) baseline DRAM bandwidth extrapolated from Xeon servers, optimistic power overhead for unused bandwidth (conservative for nanostore benefits); (4) DRAM/NVM capacity based on ITRS extrapolation [4]; (5) HDD baseline extrapolated from 125MB/s 1.5TB as 2011 commodity component, SSD baseline extrapolated from Fusion-IO ioDuo; (6) Network bandwidth and power extrapolation based on [32]. (7) We study a fat-tree network with system nodes as the leaves, but the designs should work equally well or better with other topologies.

Video transcoding: The benchmark *video* models popular video transcoding web services that use the cloud for batch processing. The algorithm reads the video input files, transcodes to a new format, and stores it. For our simulations, we use the `ffmpeg` code over a large dataset. The video is in FLV format with 320x240 resolution and transcoded to JPEG snapshots.

Recommender: The *recommender* benchmark represents sophisticated machine learning algorithms with high compute complexity and regular communication patterns. We model the Netflix challenge [31] over a 5 Terabyte dataset, using matrix factorization. The algorithm iteratively refines two matrices so their product can best summarize the *movie-ratings* matrix. Large matrices are partitioned across the cluster and stored in main memory. Each iteration has four phases: two of them are matrix operations; the other interleaving phases communicate the new results to each server. It requires large memory to host the matrices and compute/communication balance. For our simulations, we use Matlab based on a parallel algorithm [31].

Search: The *search* benchmark models text search across a 128 terabyte data set, using in-memory indices to achieve sub-second response times. The workload is read-only with random access patterns. Similar to Google's in-memory text search, the entire index is partitioned across a large cluster and stored in main memory. Each server searches its local index first, and sends the top-matching document list to the front-end server. In addition to

search query throughput, this benchmark models a quality of service (QoS) requirement of less than 0.5 second average query latency. We use `lucene` in our simulations.

3.3 Specific design choices and parameters

One of the other challenges in modeling nanostores is the existence of a wide range of possible implementations. There are a number of design choices in terms of the provisioning, organization, and balance of the compute, storage, and network resources per nanostore; the sharing model across the individual nodes; as well as the network topology (including potential differences between the on-chip network, interconnect on the PCB board, and cluster-level networks). The design choices are not independent and are often constrained by technology- and circuits-level parameters (e.g., the die size and yield, the number of 3D-stacked or intra-die layers, as well as power/thermal budget per processor socket or server board).

We address this challenge with careful selection of baseline and proposed architecture parameters, and use our methodology to quickly identify key insights through a large parameter space. Figure 4 and 5 summarizes the system architectures and configuration parameters we examine for this study. All the parameters are chosen based on projected data from recent publications and industry sources (e.g., ITRS [4]).

NVM data stores. We assume a nanostore die size of 100mm^2, based on the cost-efficiency sweet-spot design point for memory chips [4]. For a PCM-based design circa 2015, assuming 8 layers of 3D and intra-die stacking, the on-chip data store capacity is 25 GB per socket. The density/capacity, access latency (150 ns) and access energy (2-20 pJ/bit) are based on published PCM models [15]. We also study other parameters for data store capacity, latency, and energy to understand the sensitivity of our results to alternative future NVMs (e.g., memristors [36]), these results provide similar insights but for brevity are not presented here.

Compute. The processor cores in the compute layer are based on low-voltage power-efficient microarchitectures with simple SRAM cache hierarchies. We study multiple different organizations for the compute layer – varying the number of cores (32 options ranging from 1 to 128; the maximum core count of 128 is based on area estimation from McPAT), the clock frequency (100MHz, 200MHz, 500MHz, 1GHz and 2GHz), the issue width and pipeline depth (2-way or 4-way), and the L2 cache size (512KB or 1MB per core).

Power density. To ensure realistic designs, we limit the power density at the socket level (including all the compute and NVM layers). The power density is limited both by practical thermal density for packaging and cooling, and by the number of available power pins. We use a 32W/cm^2 "cap" (based on today's 80W Xeon server with die size of 2.5cm^2) as a design constraint in our performance and energy efficiency evaluation. Later in our sensitivity analysis, we also study the impact of raising this cap to 50W/cm^2 (corresponding to today's high-power 125W servers) or 100W/cm^2 (for advanced future packaging/cooling technologies).

Memory bandwidth. For the projected timeframe, we expect 3D stacking to provide significantly improved bandwidth (32GB/s) between the processor and stacked memory using through-silicon vias, similar to PicoServer [11]. A large body of work exists on 3D technology and 3D-based memory organization; we choose this bandwidth as a tradeoff between bandwidth and power.

Distributed system. We model 80Gbps networking bandwidth per server (two 40Gbs NICs as extrapolated for 2015) as in a traditional architecture, and in the sensitivity study we examine the impact of higher network bandwidth. We assume a large-scale, distributed shared-nothing system abstraction, well-matched with current data-centric software. Each nanostore can be viewed as a complete, independent system executing software needed to implement a data parallel execution environment like MapReduce. We also study the effectiveness of software scaling as a variable parameter, and discuss software issues in Section 5.

Baseline architectures

Another evaluation challenge is to provide a fair comparison between the baselines and the proposed architectures. Different workloads have different resource requirements and utilization patterns of compute, memory, storage I/O and network. Therefore, using a single baseline for all the benchmarks will not accommodate workload-specific sweet-spot configurations and may lead to unfair comparisons.

To address this challenge, we choose the best performing baseline specific to individual benchmarks (using the methodology described in Section 3.1). Depending on the evaluation metric (e.g., performance, energy efficiency, or EDP), different baseline configurations are identified and used in our evaluation. For fair comparison, we fix the dataset size across different configurations (i.e., satisfying the same user requirement), and therefore compare designs with the same persistent storage capacity. Based on the node capacity values in Figure 5, this implies the nanostore cluster can have 40-240x more nodes than SSD and HDD based designs.

The *sort, cksum* and *video* benchmarks keep their data on disks and are each allocated a single DRAM DIMM (to save power). For such IO-intensive workloads, we study both traditional hard disks and SSD-based storage. *Search* and *recommender* are in-memory workloads, so the architecture stores large datasets in the main memory and has no hard disks or SSDs. Overall, we examine three classes of designs, corresponding to the HDD-, SSD- and memory-based architectures in Figure 4.

In choosing parameters for the baselines, we ignore any potential end-of-life device scaling limitations when using DRAM or Flash, and instead extrapolate historical scaling trends for capacity and bandwidth. Note that such assumptions produce baselines that make the nanostore benefits estimates more conservative. We assume configurations of 16GB per DRAM module, doubling the capacity of today's GB/$ sweet-spot 8GB DIMMs. The per-channel DRAM bandwidth is optimistically increased from today's 12.8GB/s to 25.6 GB/s bandwidth in 2015, with each DIMM consuming 10W at peak and 2W at idle. For persistent storage, we assume future HDDs each with 6TB capacity and 500 MB/s bandwidth (two generations into the future under Moore's Law scaling), and an active power consumption varying between 8W to 10W from idle to peak. Using Fusion-IO as today's SSD baseline, we model future SSDs at lower capacity per drive (1.2TB), but higher bandwidth (4.5GB/s) and improved energy efficiency (10W peak and 1W idle power).

4. EVALUATION RESULTS

This section presents evaluation results to answer the following questions: (1) Overall, how well do nanostores perform relative to the baselines, and where do the benefits come from? (2) For the evaluated architectures in Figure 4, what are the design tradeoffs with different choices of compute, network, and data store? (3) How sensitive are our results to limiting bottlenecks such as network bandwidth, power density and software scalability?

4.1 Design space analysis

Figure 6 summarizes the results from our design space exploration, focusing on the inverse of energy delay product

(EDP) across our benchmarks. Notice configurations with higher 1/EDP values are better.

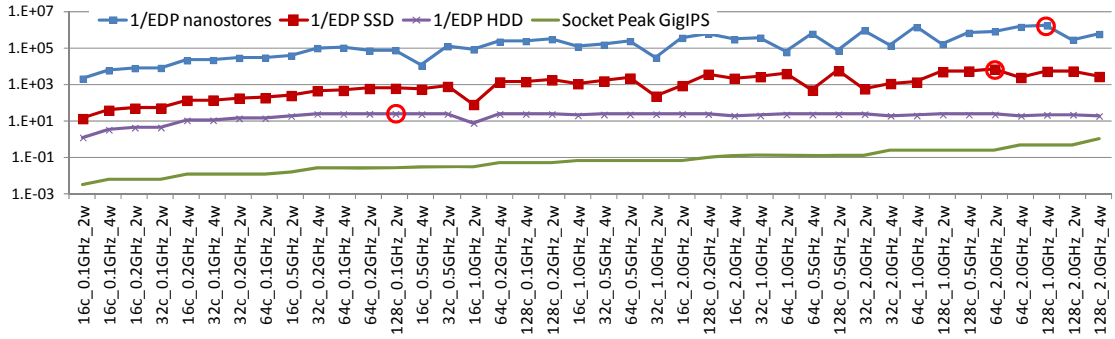
Figure 6(a) shows the 1/EDP results of *cksum*, a storage I/O intensive benchmark, for *nanostores* as well as SSD- and HDD-based designs. The entire design space has over 8000 configurations as a result of considering 32 core count options, 5 frequencies, 4 microarchitectural options in cache and pipeline, and a large number of storage and network bandwidths. For illustration, we present only 40 points along the x-axis by fixing the per-socket network bandwidth and number of storage devices, and only examining a smaller subset of processor core-count options – 16, 32, 64, and 128 cores. The x-axis in Figure 6(a) lists the details of these different processor configurations, sorted in ascending order by per-socket theoretical peak compute bandwidth in GIPS (Giga-Instructions-Per-Second). Here *16c_0.1GHz_2w* refers to a 16-core design with 100MHz, 2-way issue cores. For each configuration, we plot the per-socket peak compute bandwidth GIPS, and the relative 1/EDP values for the three system architectures (*HDD, SSD, and nanostores*).

The variation in 1/EDP values across different architectures (between curves) and configurations (within a curve) illustrates several trends. First, *NVM-based data stores remove I/O bandwidth and energy bottlenecks for data-centric workloads, and consequently can exploit a matching increase in compute bandwidth.* This observation is reflected directly in the shape and slope of the different curves. The 1/EDP curve for HDD-based designs is almost *flat,* showing the diminishing returns in EDP improvement after a modest increase in compute bandwidth due to the storage bottleneck. Higher storage bandwidth and lower energy from SSD and nanostores (the other two 1/EDP curves) can address this limitation, allowing 1/EDP to better scale with increased compute bandwidth.

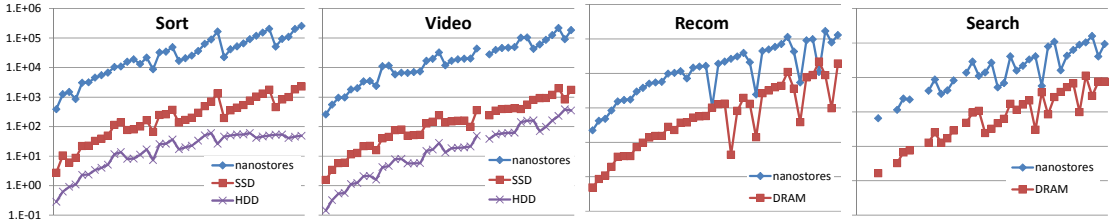
Second, although higher I/O and compute bandwidth generally improves EDP, none of the best designs (indicated by the three circled data points in the figure) choose the highest GIPS configuration. *Instead, theses designs achieve better EDP by balancing their compute bandwidth to match the storage bandwidth, and by careful performance/power tradeoffs.* Specifically, imbalanced nanostore designs with under-provisioned compute have worse EDP than balanced SSD-based designs, mainly due to lower performance and idle power overhead. Similarly, over-provisioning with peak-performance processors can degrade EDP due to under-utilized high compute power and the mismatch with I/O-intensive workloads.

Third, Figure 6(a) also illustrates *the two key factors contributing to higher compute bandwidth: (1) parallelism from increased socket count and (2) increased per-socket compute provisioning.* Because nanostores have smaller per-socket data stores than SSD- and HDD-based systems, maintaining the same system-wide data store capacity across these designs actually significantly increases the socket-level parallelism for nanostores (shown by the gap between different curves). For data points on the same curve, compute bandwidth varies with per-socket compute configuration.

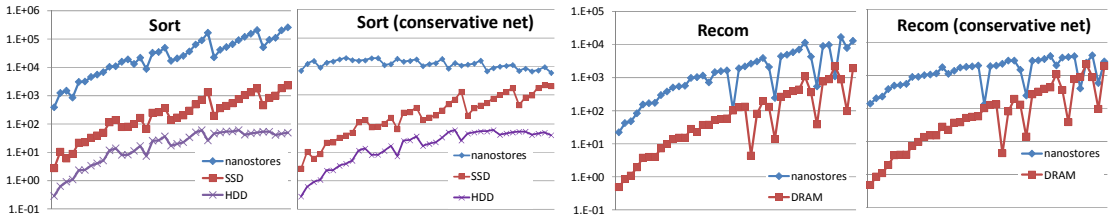
Figure 6(b) plots the 1/EDP results for the remaining four benchmarks. The figure is dense with data points, but is included to illustrate the *workload diversity*. Comparing the sub-figures



(a) 1/EDP curves for *cksum*. Two trends are highlighted: (1) NVM and in particular nanostores can remove the storage bottleneck of HDD-based designs; (2) optimizing for balance (between compute and I/O, and between performance and power) also has significant impact.



(b) 1/EDP for the remaining benchmarks, showing workload diversity and design implications. The X-axis uses the same 40 points as sub-figure (a).



(c) 1/EDP for *sort* and *recommender* with *optimistic vs. conservative* network bandwidths. *Sort* and *recom* figures from (b) are repeated for clarity.

Figure 6: Design space exploration insights.

including Figure 6(a), we observe the gaps and slopes between 1/EDP curves for each workload are different. Focusing only on nanostore curves, the best configurations are also different across workloads. For example, similar to *cksum*, the EDP improvement of *sort* with HDDs is also limited by the storage bottleneck. However the compute bandwidth at the point of diminishing return is much higher than with *cksum*. This indicates *sort* has a different compute to storage ratio requirement and can exploit higher compute bandwidth. For more compute intensive workloads (e.g., *video*), the storage bottleneck is even less visible.

These observations imply the importance of understanding and optimizing for the diversity of workloads for data-centric system architectures. Beside compute, these benchmarks also have diverse communication and quality-of-service (QoS) requirements. For example, the missing points in the *search* subfigure represent invalid configurations with low compute bandwidth that cannot satisfy *search*'s latency QoS requirements. Although not visualized here, workload diversity in these aspects can have significant performance and energy-efficiency impacts, and their combination sometimes can change the balanced design points in non-intuitive ways.

Finally, Figure 6(c) shows the need to further balance compute and IO with another key resource—network bandwidth. When the total network bandwidth of nanostores is conservatively set to be the same as in the SSD design, the EDP improvement for communication-intensive workloads (*sort* and *recommender*) becomes limited, especially for the high-compute configurations.

Overall, from a limits perspective, the nanostore design has the potential to significantly outperform traditional designs. For I/O-intensive data-centric workloads, nanostores can improve EDP by 2-3 orders of magnitude compared to HDD-based baselines, and 1-2 orders of magnitude over SSD-based baselines. The in-memory workloads achieve an order of magnitude better EDP compared to DRAM/DIMM baselines. Note that our aggressively optimized baselines make such benefits estimation more conservative. The next section discusses the benefits in details.

4.2 Performance and energy efficiency

Figure 7 presents the improvements in performance and energy efficiency from the nanostore designs relative to the baselines (for the IO-intensive workloads, HDD-based and SSD-based; and for the in-memory workloads, DIMM-based). For brevity, we plot only the *best EDP-point* from the design space exploration and show the details of the specific configuration in Figure 7(b). The results show that *for all our benchmarks, the EDP benefits from nanostores translate into both better performance and energy efficiency.* For I/O intensive benchmarks – *sort, cksum, and video* – the nanostore designs achieve 1-3 orders of magnitude higher performance improvement with 3X-16X better energy efficiency. For the in-memory benchmarks with DRAM DIMM baselines – *recom, search* – nanostores achieve 2X-6X speedup with 2X-4X better energy efficiency. The relatively smaller performance improvements for the in-memory benchmarks compared to the IO-intensive benchmarks can similarly be traced back to the baseline's high DRAM bandwidth.

Analysis of the results shows that the *greatest improvement correlates with the aggregate data store bandwidth*, resulting from the combination of both the higher per-nanostore bandwidth and lower per-socket capacity. For example, with more than 5000 times higher data bandwidth, the three I/O-intensive benchmarks no longer have any data store access bottleneck. With co-located compute, nanostores also allow significantly higher compute bandwidths (e.g., *cksum* and *video*) and network bandwidths (e.g., *sort*) to match the increased data store bandwidth, regaining the balance across resource subsystems to improve performance.

Our detailed statistics showed, surprisingly, that *not all potential bandwidth improvements enabled by 3D-stacking were fully exploited*. Further increasing the compute and network bandwidth can potentially realize the full potential of such high bandwidth, but are currently limited by processor power density and network aggregate bandwidth. These newly exposed bottlenecks limit how well the memory and storage bandwidth is used; our additional experiments (data not shown in table for space) show significantly higher performance improvements when these limitations are relaxed. Furthermore, as discussed in Section 3, our performance model and the COTSon-generated per-core memory bandwidth numbers used as input to the model are both conservative about the effect of improved memory bandwidth on performance, likely contributing further to these results. Finally, the nanostore design's *memory-like data store latency has huge performance potential* for workloads that are latency sensitive or dominated by random access patterns. However, the benchmarks we study are throughput-oriented and our performance model is mainly bandwidth based; therefore our results do not demonstrate the potential benefit of better latency.

Focusing on the energy efficiency benefits, our analysis show that *the nanostore benefits stem from three primary sources*: (1) the energy-efficiency improvements of the NVM-based data store, relative to HDDs, SSDs, and DRAM DIMM, due to lower access energy (device technology and 3D stacking) and better power proportionality (almost no idle power), (2) the use of low-power, more energy-efficient, processor cores enabled by compute co-location with lower per-nanostore capacity, and (3) reduced energy for data movement between the logically separate segments of memory and persistent storage in the nanostore's collapsed hierarchy. The last effect is conservatively modeled in the integrated model that we consider, but separate calculations show that it can provide significant benefits (e.g., 10%-30%).

4.3 Impact of the power density and network bandwidth limitations

The significant potential of nanostores motivates us to further understand the limits of such designs. So far our evaluation has focused on exploring the storage and compute subsystems; next, we address the remaining key aspects of system architecture, namely, networking, packaging/cooling (using power density as a proxy metric), and software (in terms of scaling overhead).

Figure 8(a) visualizes the effect of relaxing the socket power density and network bandwidth constraints. The socket-level power density baseline is 32W/cm^2 (as of today's 80W 2.5cm^2 Xeon server chips), and is relaxed to 50 and 100 W/cm^2. The aggregate network bandwidth (and apportioned per-socket bandwidth) is relaxed by a factor of 4 and 16 (X4 and X16 in the figure). All results are normalized to the nanostore design with conservatively extrapolated power density and network bandwidth, darker shades illustrate higher benefits.

Allowing higher power density has a positive performance effect for all workloads, matching our analysis in Section 4.2. On the other hand, raising the network bandwidth only affects the two network-heavy benchmarks (*sort* and *recom*), especially *sort* where network is the new bottleneck for performance scaling. Power density is the first bottleneck for *recom*, which has to trade core count with higher network bandwidth within the power envelope to get better performance.

(a)

Result	Performance									EE								
Watt/cm2	32			50			100			32			50			100		
Net_BW	x1	x4	x16	x1	x4	x16	x1	x4	x16	x1	x4	x16	x1	x4	x16	x1	x4	x16
Sort	1	4	22	1	4	22	1	4	22	1.0	1.0	0.8	1.0	1.0	0.8	1.0	1.0	0.8
Cksum	1	1	1	2	2	2	2	2	2	1.0	1.0	1.0	0.8	0.8	0.8	0.8	0.8	0.8
Video	1	1	1	2	2	2	3	3	3	1.0	1.0	1.0	0.7	0.7	0.7	0.6	0.6	0.6
Recom	1	2	2	1	3	3	1	3	6	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	0.9
Search	1	1	1	2	2	2	2	2	2	1.0	1.0	1.0	0.6	0.6	0.6	0.6	0.6	0.6

(b)

Result	Performance			EE			1/EDP		
Penalty	x1	x1.25	x1.5	x1	x1.25	x1.5	x1	x1.25	x1.5
Sort	1.0	0.8	0.8	1.0	0.9	0.7	1.0	0.7	0.6
Cksum	1.0	1.2	1.1	1.0	0.7	0.7	1.0	0.9	0.8
Video	1.0	0.8	1.0	1.0	1.0	0.7	1.0	0.8	0.7
Recom	1.0	1.0	0.9	1.0	1.0	1.0	1.0	0.9	0.9
Search	1.0	1.3	1.2	1.0	0.7	0.6	1.0	0.8	0.7

Figure 8: Impact of (a) power-density and network constraints and (b) software scaling overheads

Another important issue is around scaling of distributed software. The performance improvements from nanostores partly come from the larger scale of the distributed workloads, increasing the node count by factors ranging from 100 to 500. Figure 8(b) summarizes the impact when we consider the penalizing overhead of software execution time due to increased cluster size. The table considers a 25% and 50% additional penalty in overall execution time relative to an ideally scaled system with all nodes instantly starting and finishing execution and perfect load-balance. As expected, software scalability is an important consideration to achieve the benefits from nanostores. EDP degrades with reduced software efficiency in all our experiments. (Since we present a single data point for the EDP-optimal design after the design space exploration, some chosen designs would have higher performance and lower energy efficiency, but the 1/EDP value is lower in all cases.)

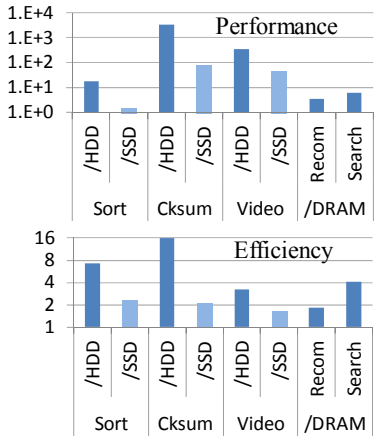
In summary, our sensitivity study leads to two key learnings: (1) network bandwidth, power density and software scaling are all potential barriers in further scaling the nanostore designs, motivating holistic optimizations across various design aspects; (2) for different benchmarks, these limitations also manifest themselves in different order and magnitude, demonstrating workload diversity and calling for workload-optimized designs.

5. DISCUSSION

Software scaling. Scaling software for nanostores presents two key challenges. First, current software stacks are developed with decades-old assumptions of traditional rotational disks. They will have to be re-architected to leverage NVM persistent data stores under the memory interface. However, recent work around byte-persistent file systems [33], consistent durable data structures [34], persistent transactions [35], and RAMCloud [10] presage the growing interest in the systems community and progress being made in this space. The second challenge is around horizontal scaling required for large-scale distributed systems. While it is worth noting that over the decade from 1998 to 2009, Google's infrastructure is reported to have scaled performance (queries processed/day) by 1000X while scaling the infrastructure size by 1000X [1], more scalable distributed algorithms are required for future nanostore designs. Recent work on data-centric software stacks (e.g., Google BigTable and MapReduce, Microsoft Dryad, Facebook Memcached) illustrate progress in this area.

Endurance. Write endurance is an important issue to consider for NVM-based architectures. NVMs such as PCM and memristor offer significantly better functionality than Flash (10^7-10^8 or more writes per cell compared to the 10^5 writes per cell, respectively). Optimizations at the technology, circuit, and systems levels [15][20][21][30] have been shown to further address endurance issues, and more improvements are likely as the technologies mature and gain widespread adoption. For the peak memory bandwidth we consider, in theory, storage wear-out can occur in 2 years for PCM based on the nanostore capacity and endurance. However, in practice, not all applications sustain write rates at peak level and the average across the application is much lower, leading to significantly longer lifetimes across the array. Wear-leveling schemes must still be used to spread writes across the memory space to prevent early failure of hot blocks. Assuming a previously proposed approach – start-gap wear leveling – at an efficiency of 90% of optimal wear-leveling [20], and using the memory write bandwidths from our simulations, we estimate per-nanostore lifetimes of 7-18 years for our benchmarks on the PCM-based design. Nevertheless, techniques that carefully manage wear-out warrant further study.

Costs. In this paper, we focus primarily on architectural and technology implications for best future designs, but cost is another issue that also needs to be considered. Current Flash memories have about an order of magnitude higher cost on a $/byte basis compared to disk. The NVMs we consider in this paper have the potential to lower these costs by more aggressive stacking and simpler fabrication processes. The improved energy efficiency of our design can also further reduce the total costs of ownership. Based on these observations, and given the increased performance, we expect the nanostore design to be competitive for performance/$ compared to high-performance storage solutions.



(a) Factors of improvements

Bench	Scheme	Configuration
Sort	HDD	28-core, 2.0GHz, 2-way issue, 1MB per-core L2$, 2x HDD, 4Gbs network
	SSD	128-core, 2.0GHz, 2-way issue, 1MB per-core L2$, 2x SSD, 10Gbs network
	PCM	22-core, 0.1GHz, 2-way issue, 512KB per-core L2$, nanostores, 0.1Gbs network
Cksum	HDD	40-core, 2.0GHz, 4-way issue, 1MB per-core L2$, 14x HDD, 8Mbs network
	SSD	104-core, 0.5GHz, 4-way issue, 512KB per-core L2$, 2x SSD, 8Mbs network
	PCM	128-core, 0.5GHz, 2-way issue, 512KB per-core L2$, nanostores, 8Mbs network
Video	HDD	40-core, 2.0GHz, 4-way issue, 1MB per-core L2$, 2x HDD, 8Mbs network
	SSD	88-core, 2.0GHz, 2-way issue, 1MB per-core L2$, 2x SSD, 8Mbs network
	PCM	128-core, 0.5GHz, 2-way issue, 1MB per-core L2$, nanostores, 8Mbs network
Recom	DRAM	56-core, 2.0GHz, 4-way issue, 1MB per-core L2$, DRAM, 4Gbs network
	PCM	128-core, 2.0GHz, 2-way issue, 1MB per-core L2$, nanostores, 4Gbs network
Search	DRAM	80-core, 2.0 GHz, 4-way issue, 1MB per-core L2$, DRAM, 8Mbs network
	PCM	128-core, 0.5 GHz, 2-way issue, 512KB per-core L2$, nanostores, 8Mbs network

(b) Summary of balanced design points presented in sub-figure (a)

Figure 7: Performance and energy efficiency improvements over 2015 baselines

Workload diversity and implications. The diverse workload requirements shown in our study motivates system architectures that can support a wide range of applications. For workloads that can exploit high memory and storage performance (in bandwidth or latency) from the tight integration of compute and NVM, nanostores can offer significant better performance and EDP. Such compute-to-data integration can be implemented either through 3D-stacking or side-stacking (i.e., 2.5D-stacking via silicon interposer), but the balance between resource types needs to be predetermined at the time of manufacturing/packaging. For workloads that are not memory and I/O bandwidth limited, or instead bottlenecked by network or software parallelism, their preferred architectures are likely to incorporate NVM in different ways. The need to support diversity also stems from distinct phase behaviors within a single application. We believe that such flexibility is likely to be supported by system architectures that integrate and utilize heterogeneous building blocks, either on-chip or distributed across discrete components/systems.

6. RELATED WORK

Section 2 already discussed the large body of prior work on system architectures using non-volatile memory (e.g., [5][6][7][8][13][14][15][16][21][30]). To the best of our knowledge, this work is the first limits study of benefits from such designs in large-scale data-centric system architectures.

The co-location of compute close to the data store in our nanostore designs is thematically similar to Active Storage [23]. However, Active Storage incorporates compute closer to disk, in the form of more powerful disk controllers for offloading and streaming. The main processor is still a deep memory hierarchy away. The IRAM and PIM proposals [12][19] examine integrating a processor with the main memory system, but mainly address challenges with CPU-logic/DRAM integration in the same fabrication process and benefits with vector streaming programming models. In contrast, our work integrates the persistent data store with compute with 3D-stacking and addresses system balance for distributed data-centric workloads.

Recently, the RAMCloud project [19] has proposed distributed systems where all data resides in DRAM. Their research primarily focuses on the software stack, around low-latency RPC, durability, data model, scaling, and consistency, etc. Although several of their motivating arguments are similar to ours, we differ in our assumptions around all data residing in 3D-stacked NVM and in our architectural explorations around balanced designs.

Other recent studies have examined using lower-power "wimpy" cores for energy efficiency [5][6][8][18][24] while also being aware of the impact on quality of service [22]. There has also been prior work on ultra-low-voltage core design [28]. Recent architectural proposals have studied 3D stacking and demonstrated its viability and benefits for improved bandwidth and memory redesign (e.g. [11][29]). We use these techniques as well, but in a different context. Several studies have proposed optimizations to improve endurance [15][21][29][30] and others have identified potential improvements in the future [4].

7. CONCLUSIONS

Data and data-centric computing are steeply on the rise. The recent adoption of non-volatile storage, both in the HPC and business worlds, presents an interesting inflection point and an opportunity for new designs for this market. This paper explores how we should design future NVM-based system architectures targeted at data-centric workloads.

We analyze current proposals for system designs using non-volatile memory and identify an important trend towards using NVM as persistent disk replacement in close proximity to the compute element. We develop and validate a new evaluation methodology, including representative data-centric workloads, and analyze an example proposal in this space, nanostores, that incorporate compute with 3D-stacked NVM on a single chip and use NVM as both memory and storage. Our evaluation shows significant benefits (an order of magnitude or higher improvement in EDP) for such an approach but also highlights the challenges – particularly in software, networking and power density scaling – to achieve this potential. Our analysis also illustrates new insights on the implications of system balance for future architectures. Overall, the findings of our study argue for future NVM-based system designs to incorporate and tradeoff three key principles – co-location of compute closer to data, higher parallelism better supported by modern software stack and networking infrastructure, and emphasis on system balance across compute, network, and storage subsystems to improve energy efficiency along with performance.

We further quantify the impact of potential limitations to this design. Given the smaller capacities of per-node storage, the number of nodes in the system increases dramatically. This can potentially increase the stress on the networking subsystem specifically due to bandwidth contention (particularly for all-to-all communication), topological complexity, port count, and power. Software scalability can also be an issue. While large-scale deployments of data-centric workloads have been demonstrated, latency requirements (e.g., sub-second response time for a search request) will still have to carefully factor in the sizing of the system. Finally, chip-level thermal constraints can limit the amount of compute packaged per nanostore, leading to a potential compute bottleneck (e.g., for machine learning algorithms used in recommendation systems).

Looking ahead, while our results are promising, we believe we have only scratched the surface of what is possible. We are currently examining the rich architectural space enabled by future data-centric designs, including heterogeneous architectures and integrated optics. There are also interesting opportunities for hardware/software co-design including new interfaces and persistent data store resilience. The large scale and low latency of such designs will likely enable new, previously-not-possible applications, allowing for more sophisticated insights from larger diverse data; these will provide even more opportunities for future research.

8. AKNOWLEDGEMENT

This research was partially supported by the US Department of Energy under Award Number DE - SC0005026. The disclaimer can be found at http://www.hpl.hp.com/DoE-Disclaimer.html.

REFERENCES

[1] M. Mayer. The physics of data. *Talk at Xerox PARC*, 2009.

[2] P. Ranganathan, From Microprocessors to Nanostores: Rethinking Data-Centric Systems. IEEE Computer Vol. 44(1), 2011, pp. 39-48.

[3] COTSon: Infrastructure for system-level simulation. *MICRO Tutorial*, 2008.

[4] ITRS roadmap. *http://www.itrs.net/*, 2009.

[5] D. Andersen, et al. FAWN: A fast array of wimpy nodes. *SOSP,* 2009.

[6] A. Caulfield, et a;. Gordon: an improved architecture for data-intensive applications. *IEEE Micro,* 30(1), 2010.

[7] H. Volos, A. Tack, et al. Mnemosyne: Lightweight Persistent Memory. *ASPLOS*, 2011.

[8] A. Cockcroft. Millicomputing: The future in your pocket and your datacenter. *USENIX invited talk*, 2008.

[9] J. Dean and S. Ghemawat. MapReduce: Simplified data processing on large clusters. *OSDI*, 2004.

[10] J. Ousterhout et. al. The case for RAMCloud. *Communications of the ACM*, 54(7):121-130, 2011.

[11] T. Kgil et al. PicoServer: Using 3D Stacking Technology To Enable A Compact Energy Efficient Chip Multiprocessor. *ASPLOS*, 2006.

[12] M. Gokhale, B. Holmes, and K. Iobst. Processing in memory: the terasys massively parallel PIM array. *Computer*, 28(4):23–31, 1995.

[13] T. Kgil and T. Mudge. FlashCache: a NAND Flash memory file cache for low power web servers. *CASES*, 2006.

[14] T. Kgil, D. Roberts, and T. Mudge. Improving nand Flash based disk caches. *ISCA*, 2008.

[15] B. C. Lee, et al. Architecting phase change memory as a scalable dram alternative. *ISCA,* 2009.

[16] D. Lewis and H. Lee. Architectural evaluation of 3D stacked RRAM caches. *IEEE 3D System Integration Conf.*, 2009.

[17] S. Li, et al. McPAT: An integrated power, area and timing modeling framework for multicore and manycore architectures. *MICRO,* 2009.

[18] K. Lim, et al. Understanding and designing new server architectures for emerging warehouse-computing environments. *ISCA,* 2008.

[19] D. Patterson, et al.A case for intelligent RAM. *IEEE Micro*, 1997.

[20] M. K. Qureshi, et al. Enhancing lifetime and security of pcm-based main memory with start-gap wear leveling. *MICRO-42*, 2009.

[21] M. Qureshi, et al. Scalable high performance main memory system using phase-change memory technology. *ISCA*, 2009.

[22] V. Reddi, et al. Web Search Using Small Cores: Quantifying the Price of Efficiency. ISCA, 2010.

[23] E. Riedel, et al. Active disks for large-scale data processing. *IEEE Computer*, vol 34, , 2001.

[24] S. Rivoire, et al. JouleSort: a balanced energy-efficiency benchmark. *SIGMOD*, 2007.

[25] P. Clark, et al. Processing Petabytes per Second with the ATLAS Experiment at the LHC in CERN. GPU Tech. Conf., 2010.

[26] Zichen Xu, et al. Exploring power-performance tradeoffs in database systems. ICDE, 2010.

[27] Fan Yang, et al. Formalizing mapreduce with CSP. ECBS, 2010.

[28] B. Zhai, et al. Energy efficient near-threshold chip multi-processing. *ISLPED*, 2007.

[29] W. Zhang and T. Li. Exploring phase change memory and 3D die-stacking for power/thermal friendly, fast and durable memory architectures. *PACT*, 2009.

[30] P. Zhou, et al. A durable and energy efficient main memory using phase change memory technology. *ISCA*, 2009.

[31] Y. Zhou et al. Large-scale Parallel Collaborative Filtering for the Netflix Prize. *Algo. Aspects in Information and Management*, 2008.

[32] D. Abts et al. Energy proportional datacenter networks. ISCA, 2010.

[33] J. Condit et al, Better I/O through byte-addressable, persistent memory. *SOSP*, 2009.

[34] S. Venkataraman et al. Consistent and Durable Data Structures for Non-Volatile Byte-Addressable Memory. *FAST*, 2011.

[35] J. Coburn et al. NV-Heaps: Making Persistent Objects Fast and Safe with Next-Generation, Non-Volatile Memories. *ASPLOS*, 2011.

[36] D. Stukov, G. Snider, D. Steward, and R. Williams. The missing memristor found. *Nature*, volume 453, pages 80–83, 2008.

Appendix A1: Detailed execution plans and performance/power models

Figure 9 shows the execution plans for our benchmarks: *sort*, *cksum*, *video*, *recom*, and *search*. The horizontal arrows indicate execution time progress, and the dotted vertical lines in each benchmark demarcate the boundary of execution phases. The execution time and power calculation as well as power density and QoS constraints are described in the embedded formulas and equations.

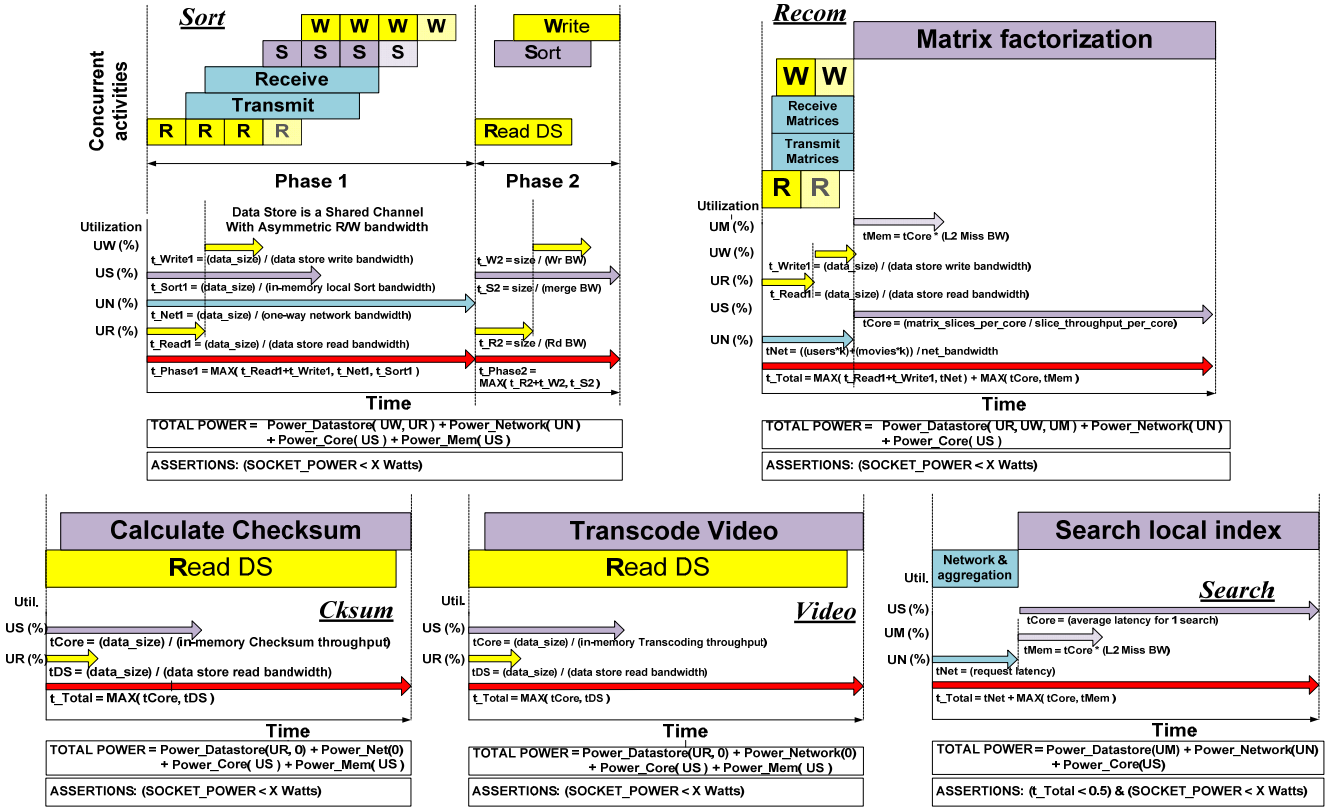


Figure 9: Benchmark execution plans with performance/power models