DASH: a Recipe for a Flash-based Data Intensive Supercomputer

Jiahua He, Arun Jagatheesan, Sandeep Gupta, Jeffrey Bennett, and Allan Snavely
San Diego Supercomputer Center (SDSC)
University of California, San Diego
Emails: jiahua@gmail.com, {arun, sandeep, jab, allans}@sdsc.edu

***Abstract*—Data intensive computing can be defined as computation involving large datasets and complicated I/O patterns. Data intensive computing is challenging because there is a five-orders-of-magnitude latency gap between main memory DRAM and spinning hard disks; the result is that an inordinate amount of time in data intensive computing is spent accessing data on disk. To address this problem we designed and built a prototype data intensive supercomputer named DASH that exploits flash-based Solid State Drive (SSD) technology and also virtually aggregated DRAM to fill the "latency gap". DASH uses commodity parts including Intel® X25-E flash drives and distributed shared memory (DSM) software from ScaleMP®. The system is highly competitive with several commercial offerings by several metrics including achieved IOPS (input output operations per second), IOPS per dollar of system acquisition cost, IOPS per watt during operation, and IOPS per gigabyte (GB) of available storage. We present here an overview of the design of DASH, an analysis of its cost efficiency, then a detailed recipe for how we designed and tuned it for high data-performance, lastly show that running data-intensive scientific applications from graph theory, biology, and astronomy, we achieved as much as two orders-of-magnitude speedup compared to the same applications run on traditional architectures.**

I. INTRODUCTION

Certain domains of science, such as genomics [1] and astronomy [2], are literally "drowning in a sea of data" in that disks are filling up with raw data from sequencing machines and space telescopes faster than that data can be analyzed. Some data analysis problems can be solved by parallel processing with many compute nodes thus spreading out the data across many physically distributed memories. Others, limited by low parallelism or challenging access patterns depend on fast I/O or large fast shared memory for good performance.

By talking to users, examining their applications, and participating in community application studies [3] [4] [5] [6], we identified data intensive HPC applications spanning a broad range of science and engineering disciplines that could benefit from fast I/O and large shared memory packed onto a modest number of nodes; included are applications in the growing areas of 1) data mining and 2) predictive science used to analyze large model output data.

In a typical *data mining* application, one may start with a large amount of raw data on disk [7]. In the initial phase of analysis, these raw data are read into memory and indexed; the resulting database is then written back to disk. In subsequent steps, the indexed data are further analyzed based upon queries, and the database will also need to be reorganized and re-indexed from time to time. As a general rule, data miners are less concerned about raw performance and place higher value on productivity, as measured by ease of programming and time to solution [8]. Moreover, some data mining applications have complex data structures that make parallelization difficult [9]. Taken together, this means that a large shared memory and shared memory programming will be more attractive and productive than a message passing approach for the emerging community of data miners. I/O speed is also important for accessing data sets so large that they do not fit entirely into DRAM memory.

A typical *predictive science* application may start from (perhaps modest) amounts of input data representing initial conditions but then generate large intermediate results that may be further analyzed in memory, or the intermediate data may simply be written to disk for later data intensive post-processing. The former approach benefits from large memory; the latter needs fast I/O to disk. Predictive scientists also face challenges in scaling their applications due to the increasing parallelism required for peta-scale and beyond [9]; they benefit from large memory per processor as this mitigates the scaling difficulties, allowing them to solve their problems with fewer processors.

As we forecast the characteristics of data intensive applications in the future, we find that today's supercomputers are, for the most part, not particularly well-balanced for their needs. Creating a balanced data intensive system requires acknowledging and addressing an architectural shortcoming of today's HPC systems.

The deficiency is depicted graphically in Figure 1; while each level of memory hierarchy in today's typical HPC systems increases in capacity by 3 orders of magnitude, the costs of each capacity increase are latencies that increase and bandwidths that decrease by at least an order of magnitude at each level. In fact, today's systems have a *latency gap* after main memory. The time to access disks is about 10,000,000

SC10 November 2010, New Orleans, Louisiana, USA 978-1-4244-7558-2/10/$26.00

processor cycles—five orders of magnitude greater than the access time to local DRAM memory. It is almost as though today's machines are missing a couple of levels of memory hierarchy that should read and write slower than local DRAM but orders of magnitude faster than disk. Since some data sets are becoming so large they may exceed the combined DRAM of even large parallel supercomputers, a data intensive computer should, if possible, have additional levels of hierarchy sitting between DRAM and spinning disk. To fill these missing levels, a data intensive architecture has at least two choices: 1) aggregate remote memory and 2) faster disks. We designed a system named DASH to make use of both. With these two additional levels (depicted in the Figure 1 as Remote Memory and Flash Drives), we managed to fill the latency gap and to present a more graceful hierarchy to data intensive applications.

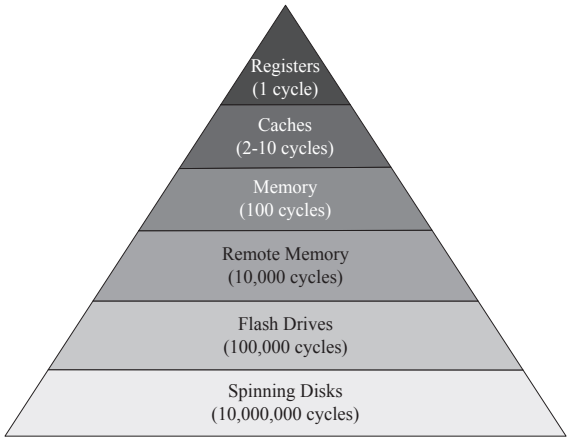

Figure 1. The memory hierarchy. *Each level shows the typical access latency in processor cycles. Note the five-orders-of-magnitude gap between main memory and spinning disks.*

In section 2 we describe the high-level design of DASH and compare its efficiency to other designs in the same space. In section 3 we supply the detailed "recipe" we used to design and tune the high performing flash-based I/O nodes of DASH- the intent is that the description is detailed enough so that anyone can understand our design choices and duplicate them. Section 4 describes the performance of some scientific applications - our experiments showed that DASH can achieve up to two-orders-of-magnitude speedup over traditional systems on these data intensive applications. Section 5 discusses flash generally and lessons-learned. Section 6 is related work.

II. System Overview

DASH is comprised of 4 "supernodes" connected by DDR Infiniband. Each supernode is physically a cluster composed of 16 compute nodes and 1 I/O node, virtualized as a single shared memory machine (see Figure 2) by the vSMP system software from ScaleMP® Inc. [10]. Each compute node comprised of 2 Intel® quad-core 2.4GHz Xeon Nehalem E5530 processors with 48GB of local DDR3 DRAM memory. As a result, each supernode has 128 cores, 1.2TFlops of peak capability, and 768GB of global (local + remote) shared memory. The I/O node is loaded with 16 Intel® X25-E 64GB flash drives, which amount to 1TB in total capacity. DASH has 4 such supernodes in all, 64 compute nodes with 4.8 TFlops, 3 TB of DRAM and 4 TB of flash. DASH is a prototype of the larger National Science Foundation (NSF) machine code-named Gordon slated for delivery in 2011, which will have more (32) and larger (32-way) supernodes and will feature 245TFlops of total compute power, 64TB of memory, and 256TB of flash drives.

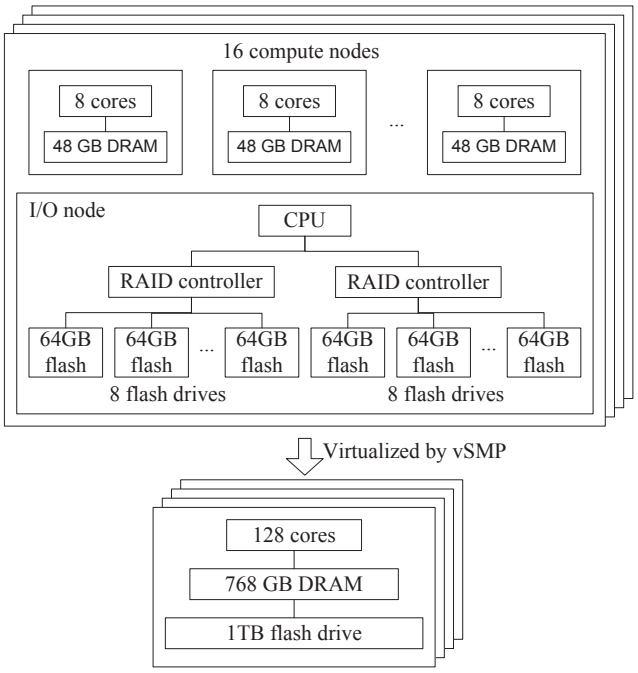

Figure 2. Physical and virtual structure of DASH supernodes. *DASH has in total 4 supernodes IB interconnected of the type shown in the figure.*

A. Storage hierarchy

Flash drives provide the first level (closest to the spinning disk) to fill the latency gap. NAND Flash is a lively research and industry topic recently [11] [12] [13] [14] [15]. Unlike traditional electromechanical hard disks, flash drives are based on solid-state electronics and have quite a few advantages over hard disks, such as high mechanical reliability, low power consumption, high bandwidth, and low latency. Their latency is about 2 orders of magnitude lower than that of spinning disks. With these faster drives, we can bring user data much closer to the CPU. Flash drives can be classified as MLC (Multi-Level Cell) and SLC (Single-Level Cell) drives. We chose SLC for longer lifetime, lower bit error rate, and lower latency. In our prototype system DASH, we have 1 TB of flash drives per supernode (4 TB in all). We will get more (8 TB per supernode) in Gordon (256 TB in all).

Though flash drives are much faster than spinning disks, there is still a big latency gap between DRAM memory and flash drives (see Figure 1). DASH is equipped on each compute node with 48GB of local DDR3 DRAM memory, that is, 6GB per core. In contrast, most existing supercomputers have only 1 to 2GB per core. So DASH already has a better ratio of DRAM to compute power – suitable for data intensive computing. Furthermore, as the second layer of latency-gap filler, we exploit vSMP software to aggregate distributed memory into a single address space. That means every single core in the supernode can access all 768GB of (local + remote) memory possessed by (all 16 compute nodes of) one supernode. With such a large shared memory, users can deal with applications with large memory footprint but limited parallelism, or just use all that memory as a RAM disk for fast I/O. Users with less than ¾ of a TB of data can move their data from spinning disks up to the shared memory in the memory hierarchy, a full 3 orders of magnitude closer to the CPU in terms of latency. Users with less than 1 TB of data can still avoid spinning disk and operate 2 orders of magnitude faster by loading their data on the flash of one supernode. And if a user uses the whole machine he can gain access of up to 7 TB of DRAM + Flash (3TB + 4TB) for truly large data analysis problems.

B. Cost efficiency

DASH is designed to provide cost-effective data-performance. We have focused the architecture on providing cost-efficient IOPS which should benefit all data-intensive applications. It is interesting to compare the three lowest levels of data hierarchy on DASH (the HDD, SSD, and virtually aggregated DRAM layer) to each other and some commercial offerings. Table 1 shows a cost efficiency comparison between DASH data hierarchy levels and two popular commercial products offered by 1) Fusion-I/O (ioDrive [16]) and 2) Sun Microsystems/Oracle (F5100 configuration-1 [17]).

TABLE 1. COST EFFICIENCY COMPARISON BETWEEN DASH AND COMMERCIAL PRODUCTS.

	Generic HDD (SATA)	DASH-I/O node	DASH Super node	Fusion –IO	Sun – F5100
GB	2048	1024	768	160	480
MB/s/$	~0.4	0.16	0.49	0.12	0.07
$/GB	~0.15	19.43	112.63	41.06	90.62
IOPS/$	0.4-1.0	28	52	18	9
IOPS/GB	0.05-0.1	549	5853	725	828

The cost metrics in Table 1 are collated and averaged from different sources including the technical specifications of each product available from its vendor and reseller [16] [17] [18] [19] [20] [21] [22]. The listed prices of these products were observed on the first week of February, 2010. The second column (Generic HDD) was chosen to represent that category within a range of values (price, density, and speed varies by vendor product). The cost of DASH I/O node includes the flash drives, the controllers, and the Nehalem processor; the cost of DASH supernode includes the cost of 16 dual socket Nehalem nodes, their associated memory, and the IB interconnect but not the I/O node (its performance was measured with RAM drive). The comparison to commercial products then gives an unfair cost disadvantage to DASH as the vendor's offerings are just storage subsystems and lack any substantial compute power—nevertheless, it is useful as a relative comparison. The third row (MB/s/$) can be seen as saying that bandwidth per dollar is more favorable for spinning disks and DRAM than for flash and DASH scores the best by this metric at all levels. The forth row ($/GB) says (common sense) that capacity per dollar is (in the order high to low) HDD (spinning disk), SSD (flash), DRAM and that DASH has the cheapest flash for the systems compared (the vendor system's don't have any general-use DRAM just some DRAM cache). The fifth row (IOPS/$) can be seen as saying that IOPS per dollar is more favorable for DRAM and flash than for spinning disk and DASH scores the best by this metric again. As shown on row two (GB) DASH also has more than twice as much flash capacity than either of the vendors. Row six (IOPS/GB) shows that because of having this more capacity the metric IOPS/GB looks better for the vendors at the flash but that is in part because they have less than ½ the flash (DASH still has the highest value in the row six category not due to flash but due to its virtual DRAM supernode layer). DASH then is a very high performing and cost-effective system compared to commercial offerings in the same space and since this paper describes how to build and tune it from commodity parts, people in the market for such a data-intensive system could consider simply building their own DASH by this recipe.

C. Power efficiency

Power and cooling costs form a major part of large data center's operating cost. Power and cooling costs can even exceed the server hardware acquisition costs over the lifetime of a system. The power consumption of flash SSDs is low, making them the right choice for DASH. Table 2 compares power metrics between flash SSD, HDD, and DRAM.

TABLE 2. COMPARISON OF POWER METRICS BETWEEN SSD AND HDD.

	DRAM 7x2 GB Dimms (14 GB)	Flash SSD 64GB	HDD 2TB
Active Power	70 W	2.4 W	11 W
Idle Power	35 W	0.1 W	7 W
IOPS per Watt	307	712	35

The numbers in Table 2 were averaged from technical

specifications of various products and independent hardware evaluation tests [18] [23] [24]. The second and the third rows are self-explanatory. The forth row compares the IOPS that can be performed per watt. Since drives are partly active and partly inactive during the course of an application's execution we can say that in general the time savings resulting from flash come with an *additional* power savings over spinning disk, IOPS/Watt may be as much as two-orders-of magnitude better than spinning disk. The substantially higher IOPS of DRAM (an order of magnitude higher than flash) comes at a higher power cost. So if one wishes to optimize IOPS per Watt (or IOPS for operating cost) then a system like DASH may be considered.

Overall, it can be seen that our experimental system DASH is a powerful, high capacity and fast system design even by commercial standards, and offers cost-efficient, power-efficient IOPS for data intensive computing.

III. I/O SYSTEM DESIGN AND TUNING

The DASH supernode (shared memory) results simply from deploying vSMP software on what is otherwise a standard IB connected system. Here we mainly focus on the design and tuning process for the I/O node describing how we chose the controller and tuned the RAID system.

To evaluate the performance of storage systems, bandwidth and IOPS are both important metrics. Bandwidth measures sequential performance while IOPS shows the throughput of random accesses. This section presents the whole tuning process of the DASH storage system. Since our target applications are characterized as intensive random accesses, we biased towards achieving high IOPS more than bandwidth in the design. To pursue and measure the peak I/O performance of the system, we adopted RAID 0 for this paper.

IOR [25] and XDD [26] are two of the most accurate, reliable, and well-known I/O benchmarks in our experience. We used both to verify each other and their results were always similar in our tests. For each software and hardware configuration, we ran four tests: sequential write, sequential read, random write and random read respectively.

Figure 3 summarizes a series performance results obtained relative to our starting baseline obtained by default settings, about 46K IOPS with 4KB blocks. After basic tunings, we obtained 88K IOPS (1.9x of the baseline) random read rate with 4KB blocks out of one I/O node; this is only about 15% of the theoretical upper bound of 560K IOPS (16x35K= 560K IOPS since the manufacturer spec is 35K IOPS random read per Intel® X25-E SSD and each I/O node has 16 drives). We figured out that a bottleneck came from the low-frequency processor embedded in our first RAID controllers (RS2BL080) and switched to simpler HBAs (9211-4i) and software RAID (using the fast Nehalem processor on each I/O node as the I/O controller rather than the embedded processor). This helped the system to scale linearly up to 8 drives, with obtained performance of about 255K IOPS (5.6x of the baseline). To keep the linear scaling up to 16 drives though we had to remove even the software RAID and handle the separated drives directly, which gave us (a little more than) theoretical upper-bound performance of 562K IOPS (12.4x of the baseline). With help of the vSMP distributed shared memory system, we were able to exploit the shared memory of DASH as a single RAM drive and boost the performance again up to 4.5 million random read IOPS, (98.8x of the baseline using DRAM in place of flash). Details of how these results were obtained are described in the following sections.

Since a single hard disk (HDD) can only do about 200 IOPS per disk (random read 4KB blocks) depending on manufacturer, it can be seen that DASH can provide two orders of magnitude higher IOPS from its flash-equipped I/O nodes and yet another two orders of magnitude from aggregated DRAM as RAM disk. These options effectively fill the latency gap.

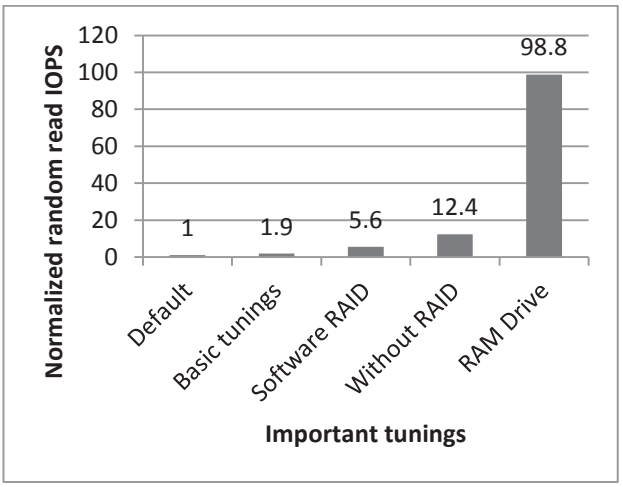

Figure 3. Random read performance improvements with important tunings.

A. Single drive tuning

Before tuning the whole I/O system, we started with tuning a single flash drive first. Table 3 shows some important tuning parameters for flash drives. We also need to tune the software components, such as I/O benchmarks and the operating system, for single-drive tests, which will be discussed later.

TABLE 3. IMPORTANT TUNING PARAMETERS FOR FLASH DRIVES.

Parameters	Descriptions	DASH setting
Write Caching	Write through or write back in the drive ram-cache	Write back
Read Ahead	Read the data into the drive ram-cache before they are requested according to the access pattern.	On
AHCI	Advanced Host Controller Interface, API for SATA host bus adapters.	On

Write caching and read ahead on other system levels might not be helpful for an intensive random workload. However, the situation is a little bit different on the flash drive level. Since the internal structure of a flash drive is highly parallel and logically continuous, pages are usually striped over the flash memory array, prefetching multiple pages and background write-back can be very efficient, while disabling these options, especially write caching, could cause a dramatic performance drop [11].

TABLE 4. I/O TEST RESULTS OF A SINGLE FLASH DRIVE.

	Sequential Write (MB/s)	Sequential Read (MB/s)	Random Write (4KB IOPS)	Random Read (4KB IOPS)
Measured	203	261	10724	39756
Spec	170	250	3300	35000

Advanced Host Controller Interface (AHCI) is Intel®'s API specification for SATA host-controllers. One of its advantages is to enable Native Command Queuing (NCQ). In a traditional spinning disk, NCQ is designed to hold (and also schedule) the I/O requests not served by the disk fast enough. In flash drives, the purpose is the opposite. It is used to stock I/O requests in case the CPU is busy and cannot summit new requests in time [27]. For backward compatibility, AHCI is disabled by default in our system. After enabling the option, we obtained more than 10x improvements on random read IOPS. Table 4 shows the I/O test results with a single flash drive. These performance numbers actually exceed the published specs of the Intel® X25-E which are also listed in the table.

B. Basic RAID tuning

The tuning parameter space of the DASH storage system is large. To achieve maximum performance, we have to coordinate all the software and hardware components of the system: I/O benchmarks, the operating system, and hardware RAID. Table 5 summarizes the important tuning parameters of these components.

Usually the operating system will try to cache the data from/to disks for future uses. Our RAID controller also has its own RAM cache for similar purposes. Unfortunately, cache doesn't always help. For example it may not help large-scale random I/Os (or even very large sequential I/Os) with low temporal locality. Even worse, it will introduce extra overhead on the data path. We enabled direct I/O to bypass the OS buffer cache and turned off the RAID cache.

There are quite a few APIs (libraries) one can use for I/O accesses. IOR supports four: POSIX, MPIIO, HDF5 and netCDF while XDD only supports POSIX. Since POSIX is the most common and typical in application code, we chose it for our tests. MPIIO is also widely used in HPC community. Unfortunately, it doesn't support direct I/O.

Chunk size is decided according to the test type and the stripe size. For sequential tests, we are trying to measure the maximum bandwidth across all the underlying flash drives and the chunk size should be larger than the stripe size times the

TABLE 5. IMPORTANT TUNING PARAMETERS FOR THE DASH I/O SYSTEM.

Components	Parameters	Descriptions	Final DASH setting
I/O Benchmarks	Cache Policy	Cached or direct I/O, use the OS buffer cache or not.	Direct I/O
	API	I/O APIs to access drives such as POSIX, MPIIO, HDF5 and netCDF.	POSIX
	Chunk Size	The data size of each request. I/O benchmarks usually generate fixed-sized requests.	4MB for sequential tests, 4KB for random tests
	Queue Depth	The number of outstanding I/O requests.	1 for sequential tests and 128 for random tests
Operating System	I/O Scheduler	Schedule and optimize I/O accesses. There are 4 algorithms in the 2.6 Linux kernel: CFQ (default), Deadline, Anticipatory, and No-op.	No-op
	Read Ahead	Read the data into cache before they are requested according to the previous access pattern.	Off
Hardware RAID	Cache Policy	Cached or direct I/O, use the RAID controller cache or not.	Direct I/O
	Write Policy	Write through or write back.	Write through
	Read Ahead	RAID-level read ahead.	Off
	Stripe Size	The block size in which RAID spread data out to drives.	64KB

number of flash drives (16 in our case). We chose 4MB, which is big enough for our stripe sizes. For random tests, we are trying to evaluate how well the system deals with small chunks of random access. Since the access unit (page size) of our flash drives is 4KB, we believe that is a reasonable (minimal) setting.

Queue depth also depends on the test type and the number of underlying flash drives. For sequential tests, since each request already covers all the underlying flash drives, we chose a setting of 1 to guarantee a strict sequential access pattern. As for random tests, to maximize the throughput, we chose 128, which is large enough comparing with the number of flash drives (16 in our case), and hopefully can make a full use of each flash drive.

There are 3 goals for I/O scheduler: merging adjacent requests together, re-ordering the requests to minimize seek cost (elevator scheduling), and controlling the priorities of requests. Since there is no drive head movement in flash drives, elevator scheduling is not necessary. Also, we are not running any time critical applications and don't need prioritization either. In our experiments, the simplest No-op scheduler, which only proceeds request merging, always gave us the best result.

We can set read ahead on 3 levels: operating system, RAID controller, and flash drive. We discussed above the settings per SSD on the drive level, but things are different on the other two levels. Read ahead is good for sequential performance, but it doesn't help random accesses. Sometimes it may even waste bandwidth with extra reads and hurt random performance. Since direct I/O was adopted and read ahead became irrelevant, we just turned it off.

Again, we already discussed write-back and write-through on the drive level, but it is different on the RAID level. The common wisdom is that write-back is always better. However, it is only true for light workloads. In our case, with intensive random accesses, the write-back cache is not helpful. Also, the extra copy on the data path will hurt the performance. As a result, we adopted write-through on the RAID level.

To decide the stripe size is a difficult optimization. Usually, small stripe size will hurt sequential bandwidth because the start-up overhead dominates. For flash drives, it is even worse by causing serious fragmentation, which was proved to cause dramatic performance downgrading [11]. However, larger is not always better. After some threshold, large stripe size will limit the parallelism of I/O accesses and then the RAID system cannot exploit the bandwidth of all the underlying flash drives. We tried different sizes from 8 KB to 1024 KB and found that 64 KB and 128 KB are the best configurations for our system and workload.

With the settings in Table 5, we obtained the performance numbers for the stripe sizes of 64KB and 128KB shown in Table 6. As we measured in Table 4, the random read performance of a single flash drive is 39,756 4KB IOPS. That means the upper bound for the whole IO node should be more than 600K IOPS, which is much higher than we obtained at this stage. In the next sub-section, we will continue our adventure to figure out the problem.

TABLE 6. I/O TEST RESULTS WITH 2 DIFFERENT STRIPE SIZES.

Stripe Size (KB)	Sequential Write (MB/s)	Sequential Read (MB/s)	Random Write (4KB IOPS)	Random Read (4KB IOPS)
64	1179	2199	3749	87563
128	1275	2056	3121	79639

C. Advanced tuning

As shown above, we achieved only about 15% of the maximum performance after all those tunings. What's the problem? After some investigations, we suspected that the bottleneck might be the RAID controller. To implement the RAID function and other advanced features, also to reduce the CPU loads, the controller is embedded with a low-frequency processor (800MHz in our case). This small processor is enough for spinning disk, but not fast enough to work with flash drives. Are there any faster RAID controllers? To our best knowledge, it is the state-of-the-art RAID controller (Intel® RS2BL080) we can get that is compatible with our drives. Another option is to use simple Host Bus Adapters (HBA) without embedded processors and share the power from the host CPU. Our motherboard happens to have an on-board HBA similar to our RAID controllers but without embedded processor or hardware RAID function. We connected only 6 flash drives to compose a software RAID and achieved 153,578 4KB IOPS, almost 2x of the hardware RAID performance. This confirmed our speculation.

The on-board HBA has a corresponding external version, which is rated higher than 150K 4KB IOPS by the vendor. Each HBA can connect 4 flash drives. Our motherboard can hold 4 HBAs. By this means, with the same number (16) of flash drives, we can expect the random read performance of about 600K 4KB IOPS, which is very close to the upper bound.

With similar settings as the previous sub-section except replacing the hardware RAID with the HBAs plus the Linux software RAID, we repeated the tests. The random read performance scaled almost linear as we expected at the beginning. With 8 drives, we obtained about 250K IOPS, almost 3x as before. However, the scaling stopped after that. In Figure 4, we can see that there is a plateau from 8 to 16. Is it the RAID problem again? To answer the question, we removed the software RAID and performed our tests directly on separate drives. This time we obtained almost linear scaling from 1 up to 16 drives. The highest performance was 562,364 IOPS. We also removed the file system (XFS) and tested directly on the raw block devices. The results were almost the

same. It seems we reached the upper bound.

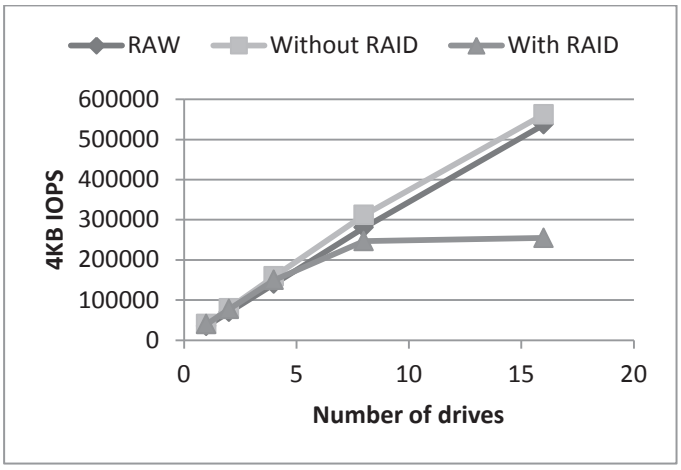

Figure 4. Random read performance with and without RAID. *The configuration with RAID only scales up to 8 drives while the one without RAID can scale linearly up to 16 drives. We also ran tests with raw block devices.*

In Table 7, we list all the I/O test results on 16 drives with and without RAID. You can see that the configuration without RAID is not only good for the random read test, but all the other tests. However, the performance with software RAID in fact is not too bad. By comparing with the results in Table 6, you will find that it still beats the original hardware RAID on almost all the tests. For the random tests, it achieved up to 5x the original performance. Though we are still investigating the RAID problem [28], it is safe to conclude that the software RAID configuration delivers a good balance between high performance and convenience. For the users who still need higher performance and don't care about the hassle to deal with 16 separate drives, the configuration without RAID is still an option and in fact there are some programming libraries around like STXXL [29] that can help to ease the job of managing the separate SSDs.

TABLE 7. I/O TEST RESULTS WITH AND WITHOUT RAID.

	Sequential Write (MB/s)	Sequential Read (MB/s)	Random Write (4KB IOPS)	Random Read (4KB IOPS)
With RAID	1395	2119	19784	254808
Without RAID	2958	3225	143649	562365

D. RAM drive

With the flash drives, we obtained optimal results at the limit of the existing hardware technologies. However, with the special design of DASH, it is still possible for us to achieve even higher performance. As mentioned above, DASH adopts vSMP distributed shared memory software system to aggregate separate physical memories into a single virtual memory. Besides the part of the memory used by the vSMP software and reserved for cache, a user has access to about 650GB visible memories from any processor in each supernode. Such a big memory space can be used as a RAM drive by mounting with the RAMFS file system. Since DRAM accessed over IB is even faster than flash drives (by up to 3 orders of magnitude!), RAM drives are expected to achieve much higher performance and the results in Table 8 show this.

TABLE 8. I/O TEST RESULTS OF THE RAM DRIVE.

Sequential Write (MB/s)	Sequential Read (MB/s)	Random Write (4KB IOPS)	Random Read (4KB IOPS)
11,264	42,139	2,719,635	4,495,592

IV. PERFORMANCE OF REAL-WORLD DATA-INTENSIVE APPLICATIONS

The behavior of I/O benchmarks as described above may be interesting and useful for comparison by simple metrics but the question remains "what is the implication for real applications"?

To partially answer this question we chose one application core from predictive science and two full applications from data mining. We present the performance results of 1) external memory BFS, a common component in several predictive science graph-based applications 2) Palomar Transient Factory a database application used to discover time-variable phenomena in astronomy data. 3) Biological pathway analysis in an integrated data-mining of heterogeneous biological data framework. All three applications generate intensive random data accesses.

A. External memory BFS

Data in several domains such as chemistry, biology, neuroscience, linguistics, and social science, are implicitly graph structured or graphs may be induced upon them. For example, semantic tagged information is encoded as a graph where nodes represent concepts and labeled edges are relationships. Search engines model the World Wide Web as a graph, with web-pages as nodes and hyperlinks as edges. Researchers in linguistics use graphs to represent semantics expressed in sentences. Networks of roads, pipelines, neurons etc. can all be viewed as graphs. Moreover, due to technological advancements, scientists are increasingly harvesting massive graphs in their respective fields. For example, human interaction networks as large as 400 million edges in size are already extant [30]. Information repository

such as NIH's Neuroscience Information Framework (NIF) [31] is projected to have more than a billion edges. Web-graphs which are studied by social scientist, mathematicians, and linguistics can be on the order of tens of billions nodes. As semantic web gains prominence and natural language processing improves, we shall see an exponential growth in graph structured data sets.

A basic type of computation over graphs that appears frequently in all such domains is that of graph traversal. Although the nature and characteristics of a graph exploration varies across domains and even across problems within a domain, they are commonly modeled after breadth-first-search (BFS). Further, other domain specific problems such as finding complexes in protein-interaction network, clustering of web-graphs, computing distance-distribution in graph models etc. utilize BFS operation. Since the total size of the graph and the content associated with every nodes and edges can run up to the order of several tera-bytes, scalable and efficient BFS computation when graphs reside in external memory would help advanced research across all these domains. This problem in literature has been referred to as external memory BFS or EM-BFS.

We used the external memory package 0.39 implemented by Deepak Ajwani *et al*. [32] in our experiments. Table 9 shows the results of one of the algorithms, MR-BFS. We ran a range of tests on a dataset size of 200 GB and compared the performance of three different storage media (RAM drive, flash drives, and spinning disks) with similar and comparable configurations. The results showed that RAM drive is on average about 2.2x faster than flash drives, and flash drives are about 2.4x faster than spinning disks for an overall speedup of 5.2x. The speedup is substantial but not as good as expected, which could be explained by the mix of bandwidth and latency bound (sparse and dense) accesses in traversing the test graph. As previous works [14] observed, write-intensive nature of an application might also be the cause.

TABLE 9. AVERAGE MR-BFS RESULTS ON THE DASH SUPERNODE FROM DIFFERENT STORAGE MEDIA

	RAM Drive	Flash Drives	Spinning Disks
Total I/O Time (sec)	854 (5.2x)	1862 (2.4x)	4444
Total Run Time (sec)	1917 (3.0x)	3130 (1.8x)	5752

B. Palomar Transient Factory

Astrophysics is transforming from a data-starved to a data-swamped discipline, fundamentally changing the nature of scientific inquiry and discovery. New technologies are enabling the detection, transmission, and storage of data of hitherto unimaginable quantity and quality across the electromagnetic, gravity and particle spectra. These data volumes are rapidly overtaking the cyber infrastructure resources required to make sense of the data within the current frameworks for analysis and study. Time-variable ("transient") phenomena, which in many cases are driving new observational efforts, add additional complexity and urgency to knowledge extraction: to maximize science returns, additional follow-up resources must be selectively brought to bear after transients are discovered while the events are still ongoing.

Current transient surveys such as the Palomar Transient Factory (PTF) [33] and the La Silla Supernova Search [34] (100GB/night each) are paving the way for future surveys such as the Large Synoptic Survey Telescope (LSST) [35] (15TB/night producing petabytes of data each year). The future sky surveys assess their effectiveness and scalability on current surveys such as PTF, in order to maximize the scientific potential of the next generation of astrophysics experiments. Two of the major bottlenecks currently confronting PTF are I/O issues related to image processing (convolution of a reference image with a new one followed by image subtraction) and performing large, random queries across multiple databases in order to best classify a newly discovered transient. PTF typically identifies on the order of 100 new transients every minute it is on-sky (along with 1000 spurious detections related to image artifacts, marginal subtractions, etc.). These objects must be vetted and preliminarily classified in order to assign the appropriate follow-up resources to them in less than 24 hours, if not in real-time. This often requires performing more than 100 queries every minute through 8 different and very large (~100GB - 1 TB) databases. The response times of these queries are crucial for PTF. The forward query and the backward query are two most significant queries used repeatedly by PTF. The average times to run these queries on DASH and the existing production infrastructure used by PTF (with same cache-size, indexes) are provided in Table 10. The difference in query response times can be attributed to the random IOPS provided by SSDs which allow faster index scans of the database rather than sequential table scans. The two-order-of-magnitude improvement in response times makes it possible for PTF to keep up with real-time demands.

TABLE 10. COMPARISON OF PTF QUERY RESPONSE TIMES ON DASH AND PTF PRODUCTION DATABASE WITH SPINNING DISKS.

Query type	Forward Query	Backward Query
DASH-IO (SDSC)	11ms (124x)	100s (78x)
Existing DB	1361ms	7785s

C. Biological pathways analysis

Systems level investigation of genomic information requires the development of truly integrated databases dealing with heterogeneous data, which can be queried for simple properties of genes as well as for complex biological-network

level properties. *BiologicalNetworks* [36] is a Systems Biology software platform for analysis and visualization of biological pathways, gene regulation and protein interaction networks. This web-based software platform is equipped with filtering and visualization tools for high quality scientific presentation of pathway analysis results.

The *BiologicalNetworks* platform includes a general-purpose scalable warehouse of biological information, which integrates over 20 curated and publicly contributed data sources including experimental data and PubMed data for eight representative genomes such as *S.cerevisiae* and *D.melanogaster*. *BiologicalNetworks* identifies relationships among genes, proteins, small molecules and other cellular objects. The software platform performs a large number of long-running and short queries to the database on postgres. These queries are a bottleneck for researchers on this domain when they have to work on the pathways using the visual interface. In our performance tests, we ran some popular queries of *BiologicalNetworks* on three different media on SDSC DASH including hard disks, SSDs and memory (using vSMP).

TABLE 11: QUERY RESPONSE TIMES OF POPULAR QUERIES IN BIOLOGICAL NETWORKS ON DIFFERENT STORAGE MEDIA (HARD DISK, SSD AND MEMORY) AND THEIR SPEED-UP IN COMPARISON TO HARD DISK.

Query	Q2C	Q3D	Q5F	Q6G	Q7H
RAMFS (vSMP)	11338ms (1.42x)	62850ms (3.60x)	3ms (186x)	17957ms (1.54x)	211ms (5.64x)
SSD	11120ms (1.45x)	176873ms (1.28x)	11ms (50.73x)	24879ms (1.11x)	495ms (2.41s)
HDD	16090ms	226023ms	558ms	27661ms	1191ms

Again, as observed in the PTF queries (Table 10), the queries of the *Biological Networks* also show improvement in their response times. But, speedup is not linear or constant across all the queries as each query uses a different query plan producing different quantity of results (or the number of rows scanned and selected from the relational database). Heavily random access patterns speedup by as much as two orders-of-magnitude while long sequential accesses run just a bit faster.

In summary some real applications speed up between 5x and nearly 200x on DASH depending on the I/O access patterns and how much the application can benefit from the random IOPS offered by DASH.

V. MORE DISCUSSIONS ON FLASH DRIVES

A. Performance downgrading

Performance downgrading is one of the concerns about replacing spinning disks with flash drives. There are mainly two causes for the problem. First, fragmentation has proved to be very harmful to the performance [11]. Fortunately, with our high-end SLC flash drives, most of the performance downgrading is still acceptable, especially the random read performance. Furthermore, some test conditions in the above paper are extreme and not common in normal uses.

Also, filling up a new drive will also hurt the performance. A new drive out of factory might be marked as free. However, since there is no abstraction of free blocks in flash drives [12], the drive will be full permanently after each block is written at least once. This will keep the full cleaning pressure and downgrade the performance. To solve the problem, the operating system and the drive firmware have to support the TRIM instruction [37] to inform the drive when the content of a block is deleted. Linux has already supported this since the version 2.6.28. Intel® already released a firmware update with TRIM for its similar product X25-M [38] and the result is promising [39]. Hopefully, the X25-E drives will be supported in a near future.

B. Reliability and lifetime

By system reliability, we are concerned about both functional failures and bit errors. Mean Time Between Failures (MTBF) is a widely-used metric for functional failure rate. Without movable mechanical parts, flash drives are more robust and easier to protect. The X25-E drives used in DASH have an MTBF of 2,000,000 hours [18]. As for bit errors, the raw Bit Error Rate (BER) of SLC NAND flash is about 10^{-9}~10^{-11}, commercial products usually apply Error Correction Code (ECC) with different strengths to lower the rate. The final error rate after ECC correction is called Uncorrectable Bit Error Rate (UBER) [13]. The UBER of X25-E is 10^{-15} [18]. That means you will get one bit flip in about 6 days if you keep reading with the sustained speed of 250 MB/s. For practical workloads, the time will be much longer. Moreover, some products such as those from Fusion-IO or Pliant claim UBERs several orders of magnitude lower.

The lifetime of a flash drive is related to its reliability, especially BER. BER increases while a block ages because of writes, i.e. Program/Erase (P/E) cycles. After some point, the flash controller will disable the block. The typical expected lifetime for SLC is 100,000 P/E cycles [15]. Manufacturers usually apply wear-leveling to distribute writes evenly across all the blocks. Our calculations indicate that under extreme use (constant write random access patterns at peak rate) the drives will not exhaust their write endurance for over 1 year. Real usage patterns will result in longer lives. To protect the system, people can adopt traditional methods such as RAID. Furthermore, flash lifetime can be predicted quite accurately with enhanced SMART (Self Monitoring, Analysis and Reporting Technology) including P/E cycle information.

C. Flash-oriented hardware and software

Flash-based SSD is a promising technology to replace traditional spinning disk. Its low latency and high throughput are going to improve the performance of storage systems dramatically. For example, in database systems, capacity is often traded for throughput. With flash drives' high

throughput, it is possible to replace hundreds of small spinning disks with just a few large flash drives [12]. To release the full potential of flash drives, the related hardware and software, such as host peripheral chipset, interconnect, RAID, and operating system, have to be modified or even re-designed. Especially, we found that RAID (hardware or software) is a limiting factor during our tuning process, and we are not the first one to observe the phenomenon [12]. As referred above, operating systems and drive firmware need to support TRIM instruction to avoid dramatic performance downgrading. With flash drives becoming widely accepted, we believe these related technologies will be stimulated to improve soon.

VI. Related work

A. ccNUMA machines

ccNUMA means Cache Coherent Non-Uniform Memory Access. It is a hybrid architecture combining the merits of SMP (Symmetric Multi-Processing) and cluster. With SMP, people can program in the same way as on their PCs. It is the most desired architecture for parallel programmers. However, such architecture is not scalable and usually limited by 32 processors/cores. To scale up, people usually group a bunch of SMP nodes together into a larger cluster. By this way, programmers might need to apply shared-memory programming model intra-node and message-passing model inter-nodes for optimal performance. ccNUMA machines try to turn the distributed memory on these SMP nodes into a single shared memory space by special hardware. There are a few commercial products around like the SGI Altix 4000 series, HP Superdome, and Bull NovaScale 5000 series [40]. With these machines, people can program across all the nodes in shared-memory model. However, these products usually adopt proprietary technology based on customized hardware, and need a long development period, which makes their ratios of performance to price pretty low. As we will discuss in the next sub-section, vSMP is a software implementation of ccNUMA and is much more cost efficient.

B. Distributed Shared Memory (DSM)

Since ccNUMA is an expensive solution, people try to achieve the same function with a software implementation called Distributed Shared Memory (DSM). The idea was first proposed and implemented in IVY [41]. During the late 1980s and early 1990s, there were a lot of projects, such as TreadMarks [42], Shrimp [43], and Linda [44], inspired by the idea and trying to improve in different ways. Though the idea is very attractive, these systems didn't get widely adopted. However, there appeared several commercial and academic DSM systems again recently [10] [45] [46]. We believe it is the right time to revisit the problem for several reasons. First, most of those old systems were developed in late 1980s and early 1990s and mainly worked with Ethernet. The high network latency limited their performance. With the low-latency inter-connect like Infiniband [47] today, the limitation is largely eliminated. Second, the workloads today are changing. Data intensive applications are becoming dominant, and the requirement for large shared memory is becoming stronger. Last but not least, most of the new systems exploit the virtual machine technology and implement the DSM layer under the operating system and right above the hardware. This might bring more opportunities to optimize. Also, it provides a single system image to the operating system and eases the management burden.

VII. Conclusions and future works

We are entering the HPC era of data intensive applications. Existing supercomputers are not suitable for this kind of workloads. There is a 5-orders-of-magnitude gap in the current storage hierarchy. We designed and built a new prototype system called DASH, exploiting flash drives and remote memory to fill the gap. Targeting at random workloads, we tuned the system and achieved ~560K 4KB IOPS with 16 flash drives and ~4.5M 4KB IOPS with 650GB RAM drive. With 3 real applications from graph theory, biology, and astronomy, we attained up to two-orders-of-magnitude speedup with RAM drives compared with traditional spinning disks. As for cost efficiency, flash is cheaper than DRAM but more expensive than disk yet the cost of operation (power) of flash is less than spinning disk.

DASH is a prototype system of the even larger machine called Gordon, which has much more flash drives and memory. To achieve good performance with such a huge system, we need to figure out how to scale up the storage system and the DSM system.

New storage media like flash and PCRAM is a hot research direction. How to integrate flash into the storage hierarchy is one of the difficult topics. It can be used as disk replacement, memory extension, disk cache, and more. We will investigate what is the best way to use flash in our systems.

Acknowledgements

The authors would like to acknowledge Gordon team members including Michael Norman, Eva Hocks, Larry Diegel, Mahidhar Tatineni, Thomas Hutton, Steven Swanson, Lonnie Heidtke and Wayne Pfeiffer. We would like to credit our science collaborators Peter Nugent and Janet Jacobsen of the Berkeley Lab for the description about PTF presented in section IVB, and also thank them for sharing with us the PTF data and its existing runtimes. We would also like to thank our science collaborators Maya Sedova and Mihail Baitaliuc of *BiologicalNetworks* for sharing us their bio-informatics data and its common queries.

This work was sponsored in part by National Science Foundation under NSF OCI #0951583 entitled "I/O Modeling EAGER" and by NSF OCI #0910847 entitled "Gordon: A Data Intensive Supercomputer".

References

[1] F.S. Collins, M. Morgan, and A Patrinos, "The Human Genome Project: Lessons from Large-Scale Biology," *Science*, vol. 300, no. 5617, pp. 286 – 290, April 2003.

[2] D.G. York, "The Sloan Digital Sky Survey," *Astronomical Journal*, 2000.

[3] A. Snavely, R. Pennington, and R. Loft, Eds., *Workshop Report: Petascale Computing in the Geosciences*.: http://www.sdsc.edu/~allans.

[4] A. Snavely, G. Jacobs, and D. A. Bader, Eds., *Workshop Report: Petascale Computing in the Biological Sciences*.: http://www.sdsc.edu/~allans.

[5] P. Kogge, Ed., *ExaScale Computing Study: Technology Challenges in Achieving Exascale System*.: http://www.sdsc.edu/~allans.

[6] A. Snavely et al., "Performance Modeling of HPC Applications," in *ParCo*, 2003.

[7] A. Szalay and J. Gray, "Science in an exponential world," *Nature*, vol. 440, pp. 413-414, March 2006.

[8] A. Funk, V. Basili, L. Hochstein, and J. Kepner, "Analysis of Parallel Software Development using the Relative Development Time Productivity Metric," *CT Watch*, vol. 2, p. 4A, November 2006.

[9] D.A. Bader, Ed., *Petascale Computing: Algorithms and Applications*.: Chapman & Hall/CRC Press.

[10] ScaleMP inc., http://www.scalemp.com/.

[11] F. Chen, D. Koufaty, and X. Zhang, "Understanding intrinsic characteristics and system implications of flash memory based solid state drives," in *SIGMETRICS/Performance*, 2009, pp. 181-192.

[12] N. Agrawal et al., "Design Tradeoffs for SSD Performance," in *USENIX Annual Technical Conference*, 2008, pp. 57-70.

[13] N. Mielke et al., "Bit error rate in NAND Flash memories," in *IEEE International Reliability Physics Symposium*, 2008, pp. 9-19.

[14] S. Park and K. Shen, "A Performance Evaluation of Scientific I/O Workloads on Flash-Based SSDs," in *Workshop on Interfaces and Architectures for Scientific Data Storage (IASDS'09)*, 2009.

[15] L. Grupp et al., "Characterizing Flash Memory: Anomalies, Observations, and Applications," in *MICRO*, 2009.

[16] Fusionio Technical Specification of 160 GB SLC PCIe ioDrive, http://www.fusionio.com/products/iodrive/?tab=specs.

[17] Sun F5100 Technical Specification and price information, http://www.sun.com/storage/disk_systems/sss/f5100/specs.xml.

[18] Intel X25-E datasheet and technical documents, http://download.intel.com/design/flash/nand/extreme/extreme-sata-ssd-datasheet.pdf and http://www.intel.com/design/flash/nand/extreme/technicaldocuments.htm.

[19] Amazon Marketplace (retrieved on 02/10/2010), http://www.amazon.com/gp/product/B002MD05SA.

[20] Hardware Canucks Review, Western Digital 2TB HDD (retrieved on 02/10/2010), http://www.hardwarecanucks.com/forum/hardware-canucks-reviews/24310-western-digital-caviar-black-2tb-hard-drive-review-7.html.

[21] Amazon.com Marketplace, Intel X25-E 64GB (retrieved on 02/10/2010), http://www.amazon.com/X25-E-64GB-Slc-Sata-SSDSA2SH064G101/dp/B001W7E4K2.

[22] Dell Accessories (retrieved on 02/10/2010), http://accessories.us.dell.com/sna/productdetail.aspx?sku=A2595172&cs=04&c=us&l=en&dgc=SS&cid=27722&lid=628335.

[23] iXBT Labs, HDD Power Consumption and Heat Dissipation of Enterprise Hard Disk Drives, http://ixbtlabs.com/articles2/storage/hddpower-pro.html.

[24] Tom's Hardware, Intel's X25-M Solid State Drive Reviewed, http://www.tomshardware.com/reviews/Intel-x25-m-SSD,2012-13.html.

[25] IOR benchmark, http://sourceforge.net/projects/ior-sio/.

[26] XDD benchmark, http://www.ioperformance.com/.

[27] NCQ page at Wikipedia, http://en.wikipedia.org/wiki/Native_Command_Queuing.

[28] J. He, J. Bennett, and A. Snavely, "DASH-IO: an Empirical Study of Flash-based IO for HPC," in *The 5th annual TeraGrid Conference*, Pittsburgh, 2010.

[29] STXXL, http://stxxl.sourceforge.net/.

[30] Facebook, http://www.facebook.com.

[31] Neuroscience Information Framework (NIF), http://nif.nih.gov/.

[32] External memory BFS, http://www.madalgo.au.dk/~ajwani/em_bfs/.

[33] Palomar Transient Factory (PTF) website, http://www.astro.caltech.edu/ptf/.

[34] La Silla Observatory website, http://www.eso.org/sci/facilities/lasilla/.

[35] Large Synoptic Survey Telescope (LSST), http://www.lsst.org/lsst/about.

[36] Biological Networks website, http://biologicalnetworks.net/.

[37] TRIM page at Wikepedia, http://en.wikipedia.org/wiki/TRIM.

[38] Intel SSD firmware update, http://www.intel.com/go/ssdfirmware.

[39] The SSD Improv: Intel & Indilinx get TRIM, Kingston Brings Intel Down to $115, http://www.anandtech.com/storage/showdoc.aspx?i=3667&p=1.

[40] Overview of recent supercomputers 2009, http://www.nwo.nl/nwohome.nsf/pages/NWOA_7X8HXB_Eng.

[41] K. Li, "IVY: A Shared Virtual Memory System for Parallel Computing," in *International Conference on Parallel Processing*, 1988, pp. 94-101.

[42] C. Amza et al., "TreadMarks: Shared memory computing on networks of workstations," *IEEE Computer*, vol. 29, pp. 18–28, Feb 1996.

[43] M. Blumrich et al., "Virtual memory mapped network interface for the SHRIMP multicomputer," in *International symposium on Computer architecture*, 1994, pp. 142-153.

[44] S. Ahuja, N. Carriero, and D. Gelernter, "Linda and friends," *IEEE Computer*, vol. 19, no. 8, pp. 26-34, Aug. 1986.

[45] 3Lear system, http://www.3leafsystems.com/.

[46] M. Chapman and G. Heiser, "vNUMA: A virtual shared-memory multiprocessor," in *USENIX Annual Technical Conference*, 2009.

[47] Infiniband page at Wikipedia, http://en.wikipedia.org/wiki/InfiniBand.