The Case for Storage-Centric Architecture Design

Shahrukh Rohinton Tarapore† Clinton Wills Smullen† Sudhanva Gurumurthi†
Parthasarathy Ranganathan‡ Mustafa Uysal‡

† Department of Computer Science, University of Virginia
Charlottesville, VA 22904
{shahrukh,cws3k,gurumurthi}@cs.virginia.edu

‡ Hewlett Packard Labs
Palo Alto, CA 94304
{partha.ranganathan,mustafa.uysal}@hp.com

Technical Report CS-2007-08
May 2007

Abstract

The proliferation of digital data is likely to cause storage and information-centric applications to be a key part of future enterprise workloads. This likely change in the workload mix will need a corresponding rethinking of system architecture designs to be *storage-centric*. In this paper, we evaluate one possible dimension in this space: the offload of computation to processing elements closer to the storage. We make two contributions in this regard. First, we examine a continuum of choices where the computation is offloaded to *existing system components*–a disk drive processor, a disk array controller, or a management processor. Our evaluation demonstrates that there are interesting tradeoffs in the choice of each location. Second, we evaluate the impact of microarchitectural changes to enhance the offload processor. We show that our new architecture can achieve orders of magnitude higher performance, at significantly lower power.

1 Introduction

The past few years have seen an explosive growth in digital data in all market segments–desktop, enterprise, and mobile systems. For example, IDC estimates that almost 30 exabytes (30 x 10^{18} bytes) of total storage capacity were shipped in 2005 alone. This rate is expected to double in the next two years [9]. This huge growth in the amount of data has come with associated challenges in processing and management. Consequently, there has been a corresponding increase in the number and complexity of applications that process large volumes of data. These applications address a range of tasks from passive data maintenance such as virus scanning and data indexing to more active processing such as data analytics and data mining. Furthermore, applications like bioinformatics and drug discovery, scientific data processing and visualization, geographical information systems, etc. have traditionally processed large amounts of data. These are likely to continue to be important in the future as well.

We call this overall class of applications *storage-centric* to reflect their common feature of having to process large volumes of data effectively.

Given that such storage-centric workloads are likely to form an important part of future workload mixes, we argue that it is important to have a corresponding rethinking of system architecture designs targeted at these workloads. This paper evaluates one such *storage-centric* architecture design. In particular, we observe that current systems have device controllers of varying levels of complexity outside the main host processor, and that Moore's Law are likely to provide increased computational power at current costs. We leverage these observations to propose and evaluate new architectures where the storage-centric computation is *offloaded* to processing elements closer to the stored data. The ability to exploit higher-levels of parallelism closer to the storage improves performance, while the use of more power-efficient components improves energy efficiency. A key feature of our approach is the *reuse of components already present in the system*, which allows us to achieve these improvements at similar costs to previous generations. This approach is similar in spirit to past work on "Active Storage" [6, 1, 33, 22] which considered simple computation on disk controllers. However, in contrast to these studies, our work considers the entire I/O path as a programmable computational substrate, with support for general-purpose computing at various points on the I/O path to optimize performance-power tradeoffs. To this end, we make the following contributions:

- We develop a detailed simulator and benchmark suite that is representative of storage-centric applications. Using these, we evaluate the benefits of our proposed approach and consider offloading at three levels: (i) the management processor; (ii) the array controller; and (iii) the disk controller. We first discuss our approach in the context of the computational capacity found on current systems. Our results show that there is a potential for benefit, even with the restrictive computational capacity available for offloading.

- We conduct a detailed design space exploration of storage-centric architectures, looking at both to which processor to offload as well as its microarchitectural characteristics. For two of the four benchmarks that we consider, we see dramatically improved performance by factors of three to six compared to the baseline, and a 10-20% performance degradation for the other two. We discuss the tradeoffs between instruction-level parallelism (ILP), fine and coarse-grained data-level parallelism (DLP), and clock frequency, and we suggest opportunities for more radical redesigns. In all of these cases, we find offloading provides significant power efficiency. We also show how the power efficiency of computation offloading could be further improved by employing frequency scaling at the host.

The rest of the paper is organized as follows. Section 2 provides more background on storage-centric architectures and discusses the related work. The simulation infrastructure, workloads, and experimental configurations are described in Section 3. Section 4 presents the experimental results and Section 5 concludes the paper.

2 Storage-Centric Architectures

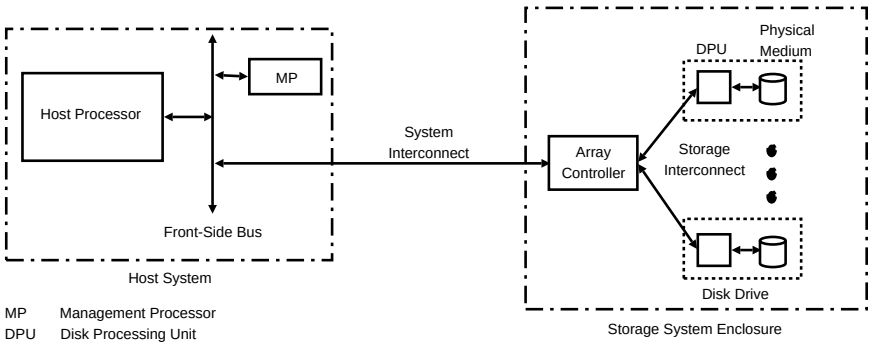


Figure 1: Architecture of the I/O Path.

Our work is based on the observation that the I/O path provides a critical optimization opportunity for improving the performance of future storage-centric applications. Figure 1 illustrates a typical system architecture. The I/O path includes the hierarchy of components between the host processor and disk drives and includes computation power embedded at various locations–in the disk drive controllers, in the array controllers, and in the management processors. Currently, these embedded processors provide limited computing power. However, Moore's Law, and the consequent increase in performance at a given cost point, is likely to provide more powerful, general-purpose processors along the I/O path in the future (subject to power budget constraints at each level). Indeed, some recent trends toward general-purpose storage controllers [13] and management processors [17] already point in this direction.

In addition to getting an increased number of computation cycles for free, these locations offer additional benefits for storage-centric workloads. In particular, the proximity to data allows for lower latencies when accessing the data. Furthermore, these embedded processors typically provide greater power efficiency, by virtue of not needing to optimize for aggressive general-purpose workloads. Offloading computation to these processors can potentially improve the energy efficiency of the overall solution. Additionally, the more local view of power and temperature events offered by these I/O path processors can enable more aggressive performance-power tradeoffs than have been considered in the past. Finally, as with any other approach to offload computation, a key issue pertains to the software model for offloading. In this paper, we assume a data-parallel implementation where the

application offloads its computation to the I/O path processors through a well-defined interface [35].

Note that we are not proposing storage-centric architectures as replacements for multi-core processors. Instead, storage-centric architectures harness the computational capability of processors on the I/O path, which are otherwise underutilized, to improve end-to-end performance. The cores on the host processor can still be used for other compute-intensive tasks in conjunction with the processing on the I/O path. However, in many servers, where the number of disks in the system tend to significantly outnumber the cores on the host [33], storage-centric architectures offer a more scalable solution, performance and power wise, for extracting DLP.

2.1 Related work

Our work derives inspiration from earlier systems work on active disks which seeks to move computation close to the data it processes [1, 22, 33]. In this paper, we argue for generalization of the computation offloading principle and evaluate a continuum of architectural choices for offloading computation onto the I/O path. Moreover, we present the first detailed analysis of computational offloading in terms of overall system power consumption and its effects on the microarchitectural design of the offload processors. To the best of our knowledge, ours is the first comprehensive evaluation of computational offloading that considers all the system attributes, including power and microarchitecture, in addition to application performance, as in [19, 25, 40].

For the general computational offloading to work, there must be broad system-software support to take advantage of its capabilities. Several projects have tackled this issue, the existing proposals include: new interfaces at the storage subsystem level like OSD [41], searchlet/disklet interface [19], and dynamic file system views [25], which serve to bridge the semantic gap at different parts of the computer system [36]. While general-purpose application processing offloading is still not in the mainstream, much of the system functionality has already benefited from offloading [30, 31].

Several projects have tackled the problem of increasing the power efficiency of computer systems at the component or subsystem level, for example dynamic voltage scaling of processors [14], memory [18], and disk drives [16], to conserve energy. Our work benefits from these techniques and introduces new possibilities to apply each of them at multiple places where we offload computation.

3 Experimental Setup and Workloads

3.1 Simulation Infrastructure

Detailed evaluation of storage-centric architectures requires good simulation support. We have built a cycle accurate execution-driven simulator for evaluating such architectures. Our simulator has detailed, parameterized models for the processors, disk drives, and interconnection network, and captures the interaction between the various components. The simulator models file system and memory management operations such as data layout and swapping. The interconnection network model provides timing information for all simulated data transfers and accounts for contention and queuing in the system. The simulator also provides an API to facilitate building applications for storage-centric architectures. We use this API to develop the suite of applications that we use in this study.

3.2 Simulated Architecture

The simulated architecture consists of four microarchitecturally heterogeneous processors- (i) host, (ii) management processor (MP), (iii) storage array controller (AC), and (iv) disk processing units (DPU). The host processor is assumed to be an 8-way superscalar processor running at 3.2 GHz with 2 GB of main memory. We assume the Front Side Bus (FSB) bandwidth of the host to be 24 GB/s. The MP is typically implemented as an ASIC. We used the description of the iLO processor given in [11] to model the MP. The MP is modeled as a 66 MHz in-order processor with 8 MB of RAM. We assume the storage system to be an array enclosure, consisting of a controller and a set of disk drives. As shown in Figure 1, the system interconnect links the storage array to the host system, which we assume to be a 3-foot VHDCI-type SCSI cable. The AC is assumed to be two generations behind the host processor and is simulated as a 500 MHz 2-way superscalar processor with 128 MB of main memory. The AC communicates with the disk drives in the enclosure through a SCSI interconnect that we assume to have a bandwidth of 320 MB/s. Each disk drive in our simulated architecture consists of a DPU, DRAM, and the mechanical data transfer system (platters and disk arms). The DPU is configured as a 200 MHz Intel XScale PXA255 processor [20], which is similar to the ARM-based embedded cores used in modern SCSI disks [4]. The disk memory size is set to 32 MB while the physical media of the disk consists of a single 3.3” platter, rotating at 10,000 RPM, having an average seek time of 4.48 ms.

Hardware Device	Power (Watts)
Host Processor	118.38
Host Main Memory	13.824
AC Processor	44.84
AC Main Memory	1.43
MP Processor	5
Spindle Motor	4.92
Arm Actuator	6.27
Windage	0.87
Read/Write Channel	2.00
200 MHz DPU	1.637
Disk Main Memory	0.252

Table 1: Power consumption parameters and values of simulated processors and disk drive.

3.3 Modeling Power Consumption

We model the power consumption of the simulated hardware components using analytical models. The total power consumed by the host and AC processors was determined using Wattch [8]. Since the AC is modeled as a processor that is two generations behind the host, we assume each of them to use a different process technology. The power consumption of the host and AC processors is calculated assuming the use of 100 nm and 250 nm process technologies, respectively. When calculating power, we conservatively assume that all the microarchitectural structures in the processor are active. The power consumed by their main memory system is calculated using Micron datasheets [28]. We assume the host to use a registered DIMM consisting of sixteen 1 Gb 333 MHz DDR2 SDRAM devices and the AC memory to use a single 1 Gb 167 MHz DDR SDRAM device. We estimate the power under the conservative assumption that all of the banks are performing data transfers. For the disk, we account for the power consumed by the DPU, memory, and the spindle and arm assemblies. The peak power consumed by the DPU is obtained from the datasheet of the PXA255. For calculating the power consumed by the disk memory, we used the specifications of a 256 Mb 133 MHz mobile SDRAM device, again calculated from Micron datasheets. For the electro-mechanical parts, we calculate the power consumed by the spindle motor, windage losses due to the spinning platter, the arm assembly, and the read/write channel that transports bits between the platters and the drive electronics. The power consumed by these parts of the disk are calculated using the models given in [23]. We assume the disks to consume their peak power as well, which occurs during periods of heavy seek activity. Since power-related details about the iLO processor are not available in the public domain, we corresponded with engineers in industry, and estimated its power to be 5 W. The power consumption of each component is summarized in Table 1.

3.4 Workloads

To evaluate the architecture considered in this study, we need applications that are inherently storage-centric. Specifically, we need applications that do significant processing and disk I/O, preferably with variations in their characteristics along both these dimensions. We identified four such storage-centric applications, which we shall now describe briefly:

- **Image Edge Detection:** This workload uses the Smallest Univalve Segment Assimilating Nucleus (SUSAN) [38] edge detection algorithm. The application detects the facial features of individuals in an image. Our image dataset is drawn from the MIT CBCL Face Recognition Database [29]. The application consists of a loop in which a processor requests an image from disk, applies the SUSAN algorithm over the pixels of the image, and moves on to the next image in the dataset. Edge-detection algorithms are widely used for Content-Based Image Recognition [37], which is becoming a popular search technique, and also for matching fingerprints [26].

- **Nearest Neighbor Search:** This search procedure is used to find the k closest matches for a given query on a database. We use a database from the National Oceanic and Atmospheric Administration which contains information about tropical cyclones in the North Atlantic from the years 1851-2005 [21]. Our nearest neighbor search procedure finds the three tropical storms that occurred closest to a given latitude and longitude coordinate. This query is processed by scanning through the entire database, calculating the Euclidean distance between the target coordinates and the storm coordinates stored in each record, and maintaining a list of the three closest matches. The Nearest Neighbor Search workload is representative of a number of applications that perform proximity searches across large datasets. These applications include data mining [5], spatial databases [34], and bioinformatics [27].

- **Malware Search:** Given the highly networked nature of modern computing systems, protecting against computer viruses and worms is a serious concern. The most commonly used malware detection technique is signature-matching, where the bytes of a file are compared against a database of known malware byte-sequences, known as "signatures." We implement parallel signature-matching using the efficient Boyer-Moore string searching algorithm [7]. We use a database of 256 randomized signatures where each signature is, on average, 64 B in length. The dataset to be scanned is randomly generated with each file being 256 KB, on average. Our technique detects the presence of malware by running an instance of the Boyer-Moore algorithm for each signature in the database. If any one of the signatures matches, the name of the file is

returned, and we skip to the next file. The Malware Search application captures the behavior of several security-related applications including: antivirus software, spyware scanners, rootkit scanners and intrusion detection systems. In fact, the performance of malware detection is considered such an important issue, especially for servers, that there are commercially available co-processors that are designed specifically to perform signature-matching [39].

- **Geo-Spatial Data Fusion:** Data fusion aims to obtain information originating from different sources by combining their individual data into a composite entity. Our workload implements basic pixel-based geo-spatial data fusion [15] on a dataset that consists of several satellite images, which we obtained from the NASA World Wind database [32]. For a given pair of images, the workload performs a sequence of three processing stages: (i) image enhancement, (ii) edge detection, and (iii) pixel-based image fusion. Image enhancement applies a combination of low-pass and high-pass filters to the imagesi, which are then fed into stage (ii), which uses the same algorithm as our Image Edge Detection workload to extract the edges of the images. The third processing stage combines common pixels in the two images to produce a new image that highlights the differences between the two input images. Geo-Spatial Data Fusion is commonly used in Geographical Information Systems, to aid urban planning, forestry, disaster relief etc. [12].

4 Experimental Results

The first set of experiments quantify the impact of offloading an application to each of the processors on the I/O path. We then conduct a detailed design-space exploration of storage-centric architectures, quantifying the performance and power benefits of offloading. Finally, we show how we can reduce the host power without impacting performance using dynamic frequency scaling.

4.1 Impact of Embedding Computation on the I/O Path Processors

Storage-centric architectures offer three key advantages: (i) a reduction in the volume of data that needs to be transported to the host, (ii) freeing up resources on the host by offloading computation onto the I/O path, and (iii) speeding up computation by exploiting DLP at the disk level. The degree to which we can meet these objectives depends on a variety of issues including the "distance" that the data has to travel on the interconnect, the microarchitecture of the offload processors, and the characteristics of the application. We need to carefully balance all these factors to achieve performance and/or power benefits. We now explore these issues in detail.

In our experiments, the scenario where the entire workload is run at the host is our *baseline*. The baseline

system uses the same number of disks as its storage-centric architecture counterparts. The primary metric that we use is *speedup*, which is calculated as the ratio between the total execution time of an application on a given configuration to that on the corresponding baseline. We simulate all disk I/O as being sequential, thereby minimizing seek and head switching activity at the disks. Each simulated disk drive is assumed to be allocated a fixed amount of application data. Therefore, in the experiments where we vary the number of disks, the applications use correspondingly larger or smaller datasets. To optimize I/O performance, we assume that for cases where we run application code at a processor other than the DPU, the files in the underlying storage system are striped across the disks. We cannot stripe the file data if code is run at the DPUs because there is no mechanism for one DPU to communicate with another, either directly or indirectly through a higher level processing component, under our architecture assumptions [41]. We use a stripe-size of 8 KB for these storage system organizations.

We consider three different processors, namely, MP, AC, and the DPU, that are candidates for running computation offloaded from the host. One simple technique is to offload the entire application to one of these processors and return only the results to the host. The effect of such offloading is shown in Figure 2. For each workload, each set of bars shows the speedup achieved when the application is offloaded to the MP, AC, and DPUs respectively. Note that the y-axis of all the graphs are in log-scale.

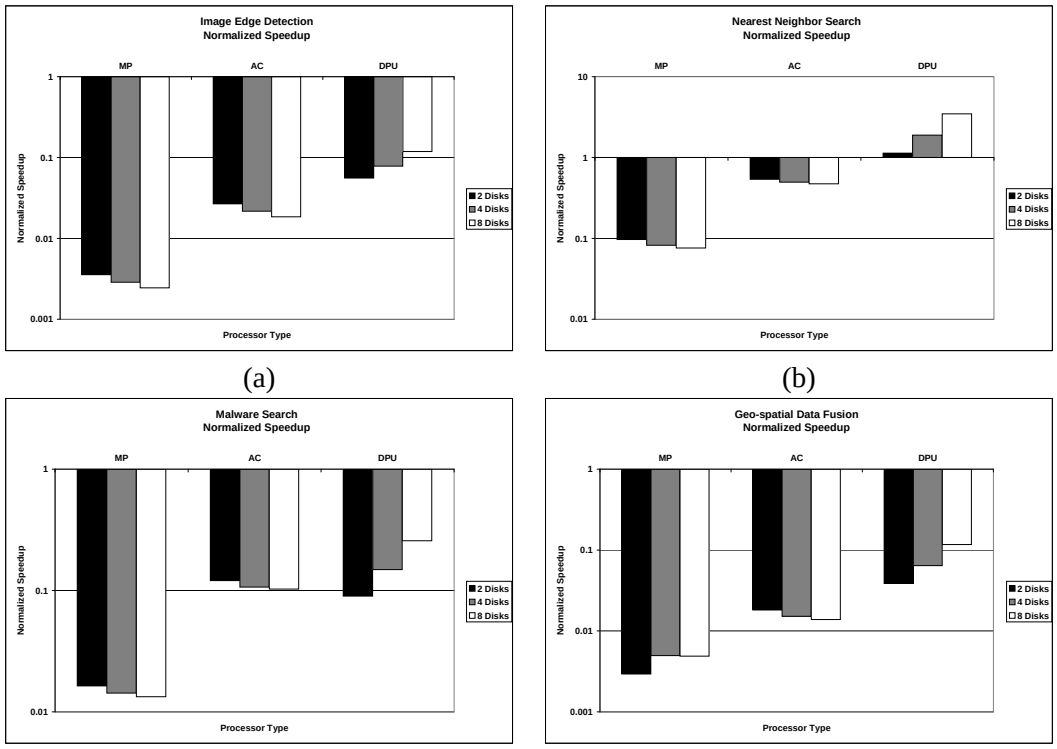


Figure 2: Impact of Embedding Computation on the I/O Path.

As we can see from Figure 2, the result of offloading computation onto the I/O path varies significantly depending upon the application and the processor that we choose to run the computation. Across all the workloads, we observe that the MP suffers the highest performance degradation. The performance degradation is exacerbated when more disks are added, since the higher number of disks results in a larger amount of data that the MP needs to handle, causing it to bottleneck even further. When we choose the AC to be the offload processor, we observe a dramatic change in performance. Recall, from Section 3.2, that the MP is a 66 MHz single-issue processor, whereas the AC is 500 MHz 2-way superscalar with 16 times the amount of memory as the MP. This higher processing capacity significantly improves performance, although the AC does not provide a speedup over the baseline for any of the datapoints. Finally, when we offload the computation to the DPUs, we start observing speedups over the baseline by virtue of being able to exploit DLP. For the Nearest Neighbor Search workload, even a 2-DPU configuration provides a speedup, and the speedup with 8 DPUs is nearly 3.5. However, we observe that for all other applications the speedup is below 1, thereby indicating that it is better to perform all the computation at the host to get the best performance. Moreover, even for those applications that experience a slowdown, the magnitude of the variation between the applications is large.

We now investigate the prime causes for these performance trends. Before we present the analysis, we first define three terms: *effective bandwidth*, *absolute bandwidth*, and *utilization*. The effective bandwidth of an application, B_{eff}, is the total amount of data transferred over the interconnection network divided by the total network transfer time. The absolute bandwidth of an application, B_{abs}, is the total amount of data transferred over the interconnection network divided by the total execution time of the application. The difference between B_{eff} and B_{abs} is that the latter accounts for idleness in the network whereas the former does not. We calculate the bandwidths over all the network links in the system. The utilization, U, is defined as $(\frac{B_{abs}}{B_{eff}})$ and is expressed as a percentage. The utilization, therefore, expresses the time that is spent by an application transferring data over the network as a fraction of the overall execution time. Utilization is an end-to-end metric that takes into account both the processing overheads and the efficiency of data transfer over all the network links. A high value for U can be interpreted as being indicative of low processing overhead and buffering delays as the data streams through the processor. From the network viewpoint, any bottlenecks in the data transfer (e.g., contention for a physical link), would translate to increases in the overall execution time and would therefore lower the U value.

We now analyze the degree to which the utilization is affected by the network and processing components. We perform an experiment where we assume that all the non-host processors are microarchitecturally *homogeneous*. We configure the MP and AC to be identical to the DPU. The utilization values for the MP (U_{MP}) and AC (U_{AC}),

Workload	2 Disks		4 Disks		8 Disks	
	U_{MP} %	U_{AC} %	U_{MP} %	U_{AC} %	U_{MP} %	U_{AC} %
Image Edge Detection	0.27	0.3	0.47	0.53	0.87	0.91
Nearest Neighbor Search	1.17	1.17	2.08	2.08	3.86	3.86
Malware Search	0.12	0.12	0.21	0.21	0.39	0.39
Geo-Spatial Data Fusion	0.16	0.16	0.28	0.28	0.5	0.5

Table 2: Utilization of MP and AC for the 4-disk configuration with homogeneous processors. The MP and AC are configured to be microarchitecturally identical to the DPUs.

for each storage-system size, using these homogeneous processors are given in Table 2.

For each of the 2, 4, and 8-disk configurations, we observe that the values for U_{MP} and U_{AC} are very close or equal to each other. We do not show the utilization for the DPUs, since they read data off the platters, not the network. Instead, the DPUs interact with the network only after they have processed the data. With homogeneous non-host processors, these results clearly indicate that the network is not fully saturated and therefore does not significantly influence the performance of the offloading strategies. Thus, the reason for the lower utilization of the offload processors is primarily processing overheads. Given this result, the remainder of this study focuses on optimizing the microarchitecture of the offload processors. Also, due to its very poor performance relative to the other processors, we do not consider the MP any further.

4.2 Microarchitectural Analysis of the Offload Processor Design

Having seen that application performance relies more on the offload processors, we explore their design in more detail. Our design space consists of: (i) the choice of offload processor, which can be AC or DPU, (ii) number of disks/DPUs, which can be 4 or 8, (iii) clock frequency, and (iv) superscalar width of the offload processor. The results of these design space exploration experiments are given in Figure 3.

The first experiment studies the impact of DPU clock frequency. In addition to the 200 MHz DPU, we consider two higher clock frequency values of 300 and 400 MHz. We choose these clock frequencies because the PXA255 processor is available at these two higher speeds. We then study the speedup that these faster DPUs provide for the 4 and 8-disk storage-system configurations. This experiment is shown in the leftmost column of graphs in Figure 3. The second set of experiments investigate what benefits, if any, could be obtained by exploiting ILP at the DPUs. We study the impact of using a superscalar DPU, for all three clock frequencies, and present the results for an 8-disk configuration in the second column in Figure 3. Since the PXA255 is not a superscalar processor, it is difficult to ascertain the exact power consumption of these DPU configurations. However, there are commercially-available microprocessors that resemble the superscalar DPUs that we simulate, and they operate at the same

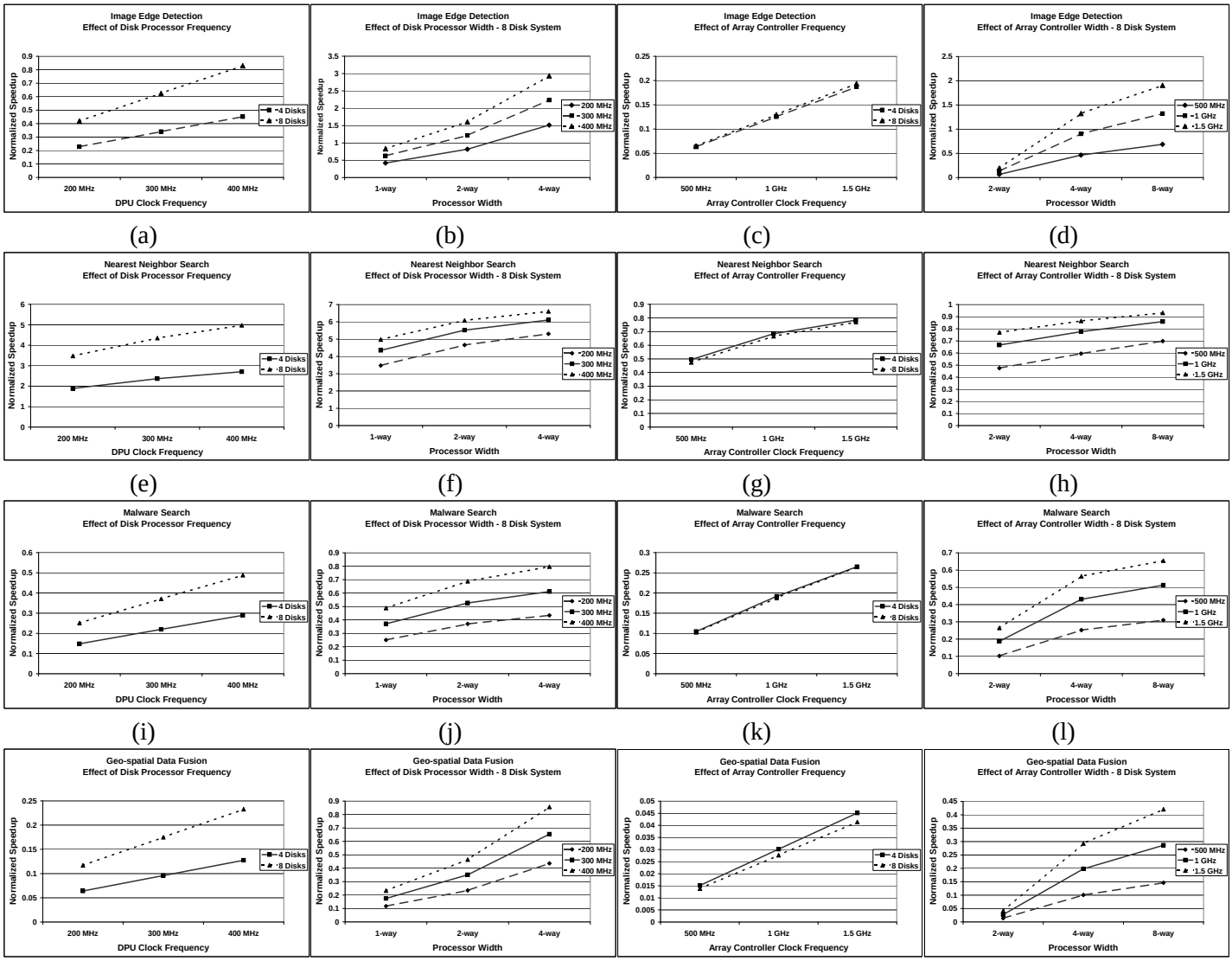


Figure 3: Design of the Offload Processor.

power consumption range as the PXA255. For example, the Hitachi SH4 [3] is a 2-way superscalar processor that operates at 200 MHz and consumes only 1.2 W of power. The next two experiments investigate processing speed and ILP effects when the computation is offloaded to the AC. We consider two higher clock frequencies of 1 GHz and 1.5 GHz and also 4 and 8-way superscalar processors for the AC. An important point to consider when incorporating such high performance processors at the AC is power consumption. In our original system, the AC is assumed to be two generations behind the host processor and therefore uses an older process technology. One way of accommodating the higher performance designs within the power budget is to first migrate the AC processor design to a newer process technology and then make the desired microarchitectural changes. The area and power

DPU Parameters

Microarchitecture	Power (Watts)
300 MHz	2.057
400 MHz	2.598

AC Parameters

Microarchitecture	Power (Watts)
500 MHz 4-way superscalar	24.4286
500 MHz 8-way superscalar	39.2802
1.0 GHz 2-way superscalar	34.1366
1.0 GHz 4-way superscalar	44.1161
1.0 GHz 8-way superscalar	73.8546
1.5 GHz 2-way superscalar	48.1667
1.5 GHz 4-way superscalar	63.1656
1.5 GHz 8-way superscalar	107.715

Table 3: Power consumption of DPU and AC. The power consumption of the 200 MHz DPU and 500 MHz 2-way superscalar AC used in the original configuration are 1.637 W and 44.84 W respectively.

benefits of using newer process technologies provide flexibility in designing more aggressive microarchitectures that would have not been feasible to accommodate within the power budget of an older technology. This technique of reusing or enhancing existing processor designs to newer process technologies has also been proposed for the design of heterogeneous multi-core processors [24]. Assuming the AC uses the same 100 nm process technology as the host, we calculate their peak power consumption, using Wattch, for all combinations of clock frequencies and processor widths. The power consumption of these additional DPU and AC microarchitectures are given in Table 3. The power consumed by several of the AC configurations are very close to the original AC. The power consumption of the faster DPUs are also within 1 W of the 200 MHz processor. The workload simulation results for the AC clock frequency and ILP experiments are given in the third and fourth columns of Figure 3.

When we look at the third column of graphs (Figures 3 (c), (g), (k), and (o)), we see that the performance of running code at the AC improves slightly when there are fewer disks in the system. The reason for this is that the AC accesses disk data that is striped, thus a single file can span multiple disks. This results in the AC receiving data from the storage system at a higher rate, particularly when the size of the files are large and there are more disks. For example, the Malware Search application processes files that are 256 KB in size on average, and therefore, with the 8 KB stripe size, we can even utilize the full bandwidth of the 8-disk configuration to transfer a file to the AC. The combined effect of parallel data transfers coupled with sequential disk I/O results in a high volume of data being transferred to the AC within a very short time interval. Therefore, when there are a larger number of disks, faster array controller processors are needed to match the higher data rates.

In the discussions that follow, we analyze the behavior of each application, looking at both performance and power aspects of the various offloading strategies and microarchitectural configurations. When we calculate the system power, we conservatively assume that the host, array controller, and the DPUs are drawing their peak power. Although we assume that disk I/O is sequential, we conservatively assume that the electro-mechanical

parts of the disk drive draw their peak (seek) power as well.

4.2.1 Image Edge Detection

For this workload, optimizing the DPU and AC provides a speedup. For the case where the workload is offloaded to the DPU, the use of superscalar processors (shown in Figure 3(b)) almost always provides a speedup. In the scenario where the Image Edge Detection workload is run on the AC, the 1 GHz 4-way and the 1 GHz and 1.5 GHz 8-way superscalar microarchitectures yield a speedup over the baseline. This speedup is due to queuing delays which are introduced when data must traverse the I/O path from the disks to the host. Each disk has a dedicated connection to the AC, from which the data must then be multiplexed onto a single channel in order to reach the host. This results in queuing delays that are twice as high for the baseline as for the AC configuration, which raises the latency of data access by the host. Comparing the range of speedup values provided by optimizing the DPU to that obtained by optimizing the AC, it is clear that the former provides more improvements in performance than the latter. This is because exploiting DLP at the DPUs provides more benefit than any latency (i.e., clock frequency) or ILP enhancement at the AC. By increasing either the clock frequency or the ILP at the DPUs, the DLP benefits get magnified.

Within the DPU optimization space, when we compare clock frequency and ILP, we observe large gains for the latter. The reason for this is that pixel data is represented as integers, and increasing the superscalar width correspondingly increases the number of integer functional units available for use. Moreover, the edge detection algorithm itself is highly data-parallel in the sense that the processing of one pixel is not dependent upon the processing of another. Therefore, if we can deliver data to the processor core efficiently and exploit ILP, we can process multiple pixels within a smaller time frame. This can be viewed as a form of *fine-grained DLP*. As we increase the amount of ILP, we exploit more of this fine-grained DLP, and achieve greater speedup. At the same time, offloading the computation onto the DPUs attempts to exploit *coarse-grained DLP*, wherein we process multiple image files, in parallel, across all the disks. Offloading computation to the superscalar DPUs allows us to exploit both fine and coarse-grained DLP, thereby providing a large performance boost.

When we look at the array controller, again we see ILP optimization providing far greater performance benefits than increasing the clock frequency. Again, this is because the AC is able to exploit the fine-grained DLP within each image. However, although the AC can exploit higher ILP than the DPUs in the 8-way configuration, it cannot exploit coarse-grained DLP and therefore the speedup provided is less than in the DPU offloading scenarios. For example, the speedup provided by an 8-way superscalar AC running at 500 MHz and that of 8 independent single-

issue DPUs running at 400 MHz are 0.69 and 0.83 respectively.

Power Analysis: For this workload, we find several iso-performance configurations within the design space of the DPU and the AC. For example, in Figure 3(a), we can see that having four DPUs running at 400 MHz yields roughly the same performance as having 8 DPUs running at 200 MHz. The system power consumed by these two DPU configurations are 238.11 W and 290.07 W respectively. Thus, use of fewer but faster DPUs provides the same level of performance with a 17.9% reduction in the system power consumption. The main reason for this large power difference is because a substantial part of the disk power is consumed by the electro-mechanical parts, e.g., the spindle motor. Comparing DPU and AC configurations, we can see that this 8-DPU configuration matches the performance of a 4-way superscalar AC that runs at 500 MHz. From Tables 1 and 3, we can see that this AC would consume 24.43 W, whereas these eight DPUs would collectively consume only 13.1 W. However, given that the AC configuration used in the original system consumes over 44.84 W, using this 4-way AC provides a 45.6% reduction in the AC power consumption and delivers higher performance than the 500 MHz 2-way controller. Overall, replacing the original AC with this lower power counterpart provides 7% savings in the total system power.

4.2.2 Nearest Neighbor Search

When we look at the results for the Nearest Neighbor Search workload, we again find that offloading to the DPUs is the best option for boosting performance. In fact, the performance scaling achieved by using 4-way superscalar DPUs (Figure 3(f)) nearly reaches the theoretical ideal speedup. However, unlike Image Edge Detection, the performance gap between clock frequency scaling and ILP scaling at the DPU level is narrower for this workload than it is for Image Edge Detection. As we increase the superscalar width, we get commensurately higher number of integer functional units but only a moderate increase in the number of floating-point units. The coordinates on which the workload performs the search opertion are represented as real numbers, and therefore the workload is floating-point intensive, using long latency operations such as square roots for calculating Euclidean distances. Since the floating-point execution pipeline is still relatively narrow, even at high superscalar widths, the ILP benefit also tends to be modest.

Power Analysis: For this workload, we find two iso-performance storage system configurations at the DPU level, which are shown in Figure 3(f). These are the (200 MHz, 2-way; 300 MHz 1-way) and (200 MHz 4-way; 400 MHz, 1-way). The fact that we can get the same speedup using faster in-order processors, rather than

going in for slower out-of-order superscalar processors has benefits from a complexity-effectiveness viewpoint. Given the challenges associated with designing processors with high superscalar widths, especially at the very low DPU power budgets, the ability to extract speedup with just in-order processors can be beneficial from the power viewpoint as well. There are design tradeoffs at the AC level as well, although the speedup we obtain by offloading the workload to this component is less than unity. Figure 3(h) shows two pairs of AC organizations that deliver nearly the same speedup. They are: (500 MHz, 8-way; 1 GHz, 2-way) and (1 GHz, 4-way; 1.5 GHz, 2-way). Table 3 shows the power consumption of these AC pairs to be very close to each other as well: (39.28 W; 34.14 W) and (44.11 W; 48.16 W) respectively.

4.2.3 Malware Search

For this workload, none of the microarchitecture optimizations, either at the DPU or at the AC, are able to provide better performance than the baseline. Given the inherent data-parallel nature of the workload, offloading computation to the DPUs is the most advantageous solution for improving performance. This application also shows good sensitivity to ILP, especially as the clock frequency is scaled up, coming close to the baseline performance with 400 MHz 4-way superscalar DPUs, as shown in Figure 3(j). We believe that this is because of the way that the malware scanning process works. Given our signature database and a set of files with randomly generated content, the probability of a mismatch (i.e., not finding the signature in the file) is quite high. Malware Search consists of three nested loops. The outer loop streams across the file as fast as the parallel Boyer-Moore implementation will allow, while the middle loop performs string matching for each signature on the current buffer contents. Since the checking of one signature is independent of the checking of another, there are no loop-carried dependences for the middle loop. This allows more iterations of the loop to be in-flight, as we increase the superscalar width, thus providing a performance benefit.

Power Analysis: As with Image Edge Detection, we again find that the 400 MHz 4-DPU system delivers slightly better performance than the 200 MHz 8-DPU system, thereby providing the same 17.6% reduction in the system power. A more interesting case arises when we compare the 400 MHz 8-DPU point, shown in Figure 3(i), to the 1 GHz 8-way superscalar AC, in Figure 3(l), which both deliver roughly a speedup of 0.5. However, the power consumption of these DPUs and the AC processor are 20.78 W and 73.85 W respectively! If we assume the power budget for the AC to be the same as that for the original AC configuration (44.84 W), then it is not possible to accommodate this 1 GHz 8-way superscalar AC within the system without provisioning additional cooling.

4.2.4 Geo-Spatial Data Fusion

Comparing the graphs for the DPU and AC, we can clearly see the benefits of exploiting DLP at the DPUs. Between clock frequency and ILP, we find that the latter has a more profound impact on performance on both the DPU and the AC. In fact, with 400 MHz 4-way DPUs, we almost break even with the baseline. Geo-Spatial Data Fusion is different from the other three workloads in the sense that the application involves multiple sub-computations. Recall, from Section 3, that there are three stages to the fusion process: image enhancement, edge detection, and pixel-based fusion. The image enhancement stage, like edge detection, can take advantage of fine-grained DLP within images and can therefore be optimized for performance on ILP processors. In the fusion stage, for each edge in the result set of an image, we need to identify the common pixels between it and the edge set of the other image, in the fusion pair; this repeats until all edges of the two images have been processed. The fusion stage does not benefit from ILP because finding common pixels within the edge set is highly dependent on the pixels around it in the images. For example, when checking if pixel i is on a common edge in the two images being fused, we must look at pixel $i + 1$ as well, since it may be part of the same edge in one or both of the edge sets. As a result of this, as the fusion stage iterates over pixels, there is a loop-carried dependence that reduces the amount of ILP that can be exploited in this stage. Therefore, increasing the clock frequency is the only option that we have for optimizing the performance of the fusion stage. Finally, since the three stages for Geo-Spatial Data Fusion need to be done in a pipelined fashion, we cannot exploit any parallelism across the stages for any given pair of images.

When we offload the work onto the DPUs, we get the benefit of coarse-grained DLP across image pairs. However, with superscalar DPUs, we can also exploit fine-grained DLP in the first two stages thus giving the large performance boost shown in Figure 3(n). In fact, when we compare the potential for performance growth with the addition of superscalar DPUs, only Image Edge Detection and Geo-Spatial Data Fusion show a steady upward trend, while the others exhibit diminishing returns. However, due to the higher processing load of the Geo-Spatial Data Fusion application, the speedup does not scale up by the same magnitude as it does for Image Edge Detection and falls short of the break-even point. Increasing the clock frequency of the DPUs only improves the performance of the third stage, thereby providing less speedup. For the AC, large superscalar widths facilitate the exploitation of fine-grained DLP. However, the AC cannot exploit coarse-grained DLP and consequently shows less speedup than when offloaded to the DPUs. On the other hand, the high clock frequencies of the AC improve the performance of the third stage. Since the pixel-based fusion stage is only a small part of the overall fusion process, scaling up the clock frequency does not provide much benefit to overall performance, as shown in Figure

3(o).

Power Analysis: Geo-Spatial Data Fusion again underscores the power consumption benefits of DLP, wherein we can use multiple lower power DPUs rather than a single powerful AC. For example, the set of 400 MHz in-order DPUs and the 1 GHz 4-way AC processor, shown in Figures 3(n) and 3(p), consume 20.78 W and 44.11 W, respectively, and deliver roughly the same performance. From Figure 3(m), we again find that using fewer but faster DPUs is preferable due to the higher power consumption of the electro-mechanical parts of the drive.

4.3 Discussion of the Results

Having studied how applications behave in the design space of storage-centric architectures, we now discuss the overall trends that we can infer from the results. Across the experiments, we observe the following:

- Between DLP, ILP, and clock frequency, DLP is the most important factor to optimize. Once we have provisioned resources for extracting DLP, ILP is the next important factor, and then the clock frequency. These results motivate the use of multiple, but relatively slow DPUs, that can exploit ILP, rather than fast, monolithic processors located elsewhere on the I/O path.

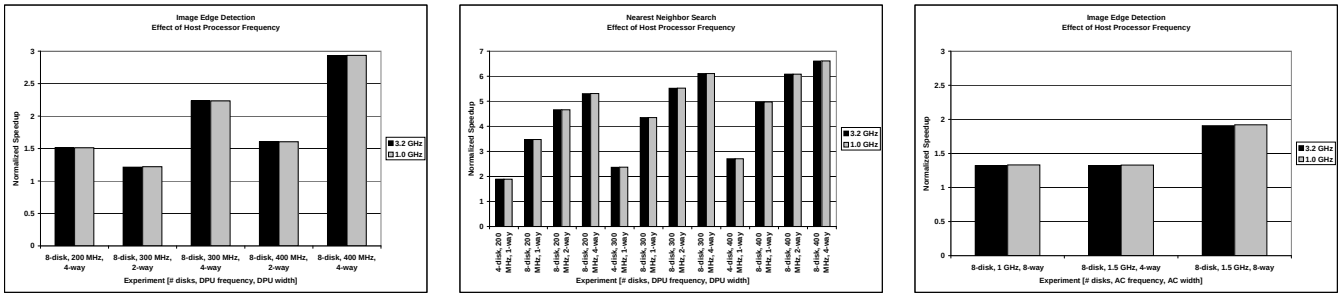
- DLP can be exploited at two granularities. Coarse-grained DLP, which is the approach advocated by active storage [1, 22, 33], has the largest impact on performance and can be obtained by using multiple disks and offloading the workload to the DPUs. Such DPU-level offloading is also typically the best in terms of power efficiency. However, if an application has fine-grained DLP as well (such as the Image Edge Detection and Geo-Spatial Data Fusion workloads), we can get large performance gains by using superscalar processors. Although the AC cannot exploit coarse-grained DLP, we can still exploit the fine-grained component and get good speedup, as we see for Image Edge Detection. As pointed out earlier, there are commercially available superscalar processors [3] that are similar to those that we have used in the experiments and whose power consumption is in the same range as the ARM-type cores that are used in modern disk drives.

- Due to the high power consumption of the electro-mechanical parts of the disk drive, it is preferable to use fewer disks with faster DPUs. In general, we find that using four 400 MHz DPUs yields roughly the same speedup as having eight DPUs clocked at 200 MHz. A large part of the electro-mechanical power comes from the spindle motor, which rotates the disk platters. Given the cubic relationship between disk RPM and power, dynamic RPM scaling techniques [10, 16] could be used to reduce the power of the electro-mechanical parts. This would allow flexibility in provisioning more powerful DPUs within established

power budgets. At the same time, the lower RPM would have a detrimental impact on disk access time. Studying the tradeoffs between the design of the electro-mechanical data transfer system and the DPUs is part of our future work.

- For applications that have phases that are more sensitive to DLP, ILP, or clock frequency than to the other factors, as in the Geo-Spatial Data Fusion application, no single processor is well-suited for meeting all the performance goals. Using multiple DPUs provides DLP, but power constraints limit the ILP or clock frequency that we can provision at each DPU. The AC has a higher power budget and, consequently, can have higher clock frequencies and superscalar widths, but they cannot exploit coarse-grained DLP. It might be more fruitful to investigate how we could use more than one processor to offload an application onto the I/O path. Exploring such offloading possibilities is part of our future work.

4.4 Reducing Host Power Consumption via Computation Offloading



(a) (b) (c)

Figure 4: Reducing power by scaling down the clock frequency of the host processor.

Storage-centric architectures provide the opportunity to offload computation, normally done on the host, onto processors on the I/O path. Since offloading lowers the utilization of the host processor, we have the opportunity to reduce the power consumption of the host and achieve a net performance benefit. The host processor is one of the largest power consumers in the system, accounting for 56% of the power in the 2-disk system and over 40% for a storage system with 8 disks. We now explore the possibility of saving power on the host when the application is offloaded to the I/O path.

We assume that the host processor is equipped with Dynamic Frequency Scaling (DFS) and that it can transition to a clock frequency of 1 GHz (from the 3.2 GHz that we have used in the experiments). Using Wattch, we determine the power consumption at this clock frequency to be 38.94 Watts, which is a 3X reduction from the baseline value, shown in Table 1. This power consumption ratio between the highest and lowest performance

states concurs with trends in modern high-performance microprocessors [2]. We select all the datapoints in our design space from the previous set of experiments that showed a speedup equal to or better than the baseline system and perform simulations with the 1 GHz host configuration. The results from this experiment are given in Figure 4.

For each configuration shown on the x-axis, each pair of bars gives the speedup with a 3.2 GHz and 1.0 GHz host processor respectively. Figures 4(a) and (b) show the datapoints where computation is offloaded to the DPU and Figure 4(c) gives the data for the Image Edge Detection workload when computation is offloaded to the array controller. We can clearly see the power benefits of offloading. Across all the datapoints, the speedup that we obtain is practically the same, whether we use a 3.2 GHz or a 1.0 GHz host.

5 Conclusion

The visible shift in the computational workloads due to explosive growth in digital data and growing concerns over the relentless surge in power consumption of systems force us to rethink our computational paradigms. In this paper, we argue for aggressive computational offloading to various processing components along the I/O path. A key aspect of our approach is the reuse of components that are already present in the system, and therefore would not be adding significant cost to existing architectural designs.

We evaluate a continuum of architectural choices to offload computation along the I/O path: (i) management processors, (ii) disk array controllers, and (iii) disk controllers. We find that there are dramatic performance improvements for two of the four benchmarks we consider by factors of three to six, though we see degraded performance by about 10–20% on the other two benchmarks. We evaluate implications of offloading computation on the system power consumption using analytical models. Our findings are very encouraging: computational offload along the I/O path produces significant power savings in all cases we consider and broadens the architectural flexibility to pick the best places to use the available power budget. We evaluate microarchitecture designs of the offloaded processors, and find that extracting DLP is the most important factor contributing to application performance, followed by ILP, and last clock frequency. Based on our analysis, we conclude that there are significant advantages in terms of performance gains and power savings to be extracted by offloading computation to various processors along the I/O path.

References

[1] A. Acharya, M. Uysal, and J.H. Saltz. Active Disks: Programming Model, Algorithms and Evaluation. In *Proceedings of the International Conference on Architectural Support for Programming Languages and*

Operating Systems (ASPLOS), pages 81–91, October 1998.

[2] AMD Opteron Processor Power and Thermal Data Sheet, May 2006.

[3] F. Arakawa, O. Nishii, K. Uchiyama, and N. Nakagawa. SH4 RISC Multimedia Processor. *IEEE Micro*, 18(2):26–34, March-April 1998.

[4] ARM Collaborates With Seagate For Hard Disc Drive Control, June 2002. ARM Press Release.

[5] C. Bohm and F. Krebs. High performance data mining using the nearest neighbor join. In *Proceedings of the International Conference on Data Mining (ICDM)*, December 2002.

[6] H. Boral and D.J. DeWitt. Database Machines: An Idea Whose Time Has Passed? In *Proceedings of the International Workshop on Database Machines*, pages 166–187, September 1983.

[7] R.S. Boyer and J.S. Moore. A Fast String Searching Algorithm. *Communication of the ACM*, 20(10):762–772, 1977.

[8] D. Brooks, V. Tiwari, and M. Martonosi. Wattch: A Framework for Architectural-Level Power Analysis and Optimizations. In *Proceedings of the International Symposium on Computer Architecture (ISCA)*, pages 83–94, June 2000.

[9] J. Buttress and D. Reinsel. Worldwide Hard Disk Drive 2005-2009 Forecast: The Growth Is Sustainable, but the Industry Is in Transition. Technical Report 33432, International Data Corporation, June 2005.

[10] E.V. Carrera, E. Pinheiro, and R. Bianchini. Conserving Disk Energy in Network Servers. In *Proceedings of the International Conference on Supercomputing (ICS)*, June 2003.

[11] Compaq Computer Corporation Technology Brief - Integrated Lights-Out Technology: Enhancing the Manageability of ProLiant Servers, April 2002.

[12] A.M. Waxman et al. Information Fusion for Image Analysis: Geospatial Foundations for Higher-Level Fusion. In *Proceedings of the International Conference on Information Fusion (ISIF)*, pages 562–569 Vol. 1, July 2002.

[13] W. W. Wilcke et al. IBM Intelligent Bricks Project-Petabytes and Beyond. *IBM Journal of Research and Development*, 50(2/3):181–197, March/May 2006.

[14] Krisztián Flautner, Steve Reinhardt, and Trevor Mudge. Automatic Performance Setting for Dynamic Voltage Scaling. In *Proceedings of the International Conference on Mobile Computing and Networking (MOBICOM)*, pages 260–271, New York, NY, USA, July 2001.

[15] R. Gens. Geospatial Data Fusion - Seminar Talk, Geophysical Institute, University of Alaska Fairbanks, February 2004.

[16] S. Gurumurthi, A. Sivasubramaniam, M. Kandemir, and H. Franke. DRPM: Dynamic Speed Control for Power Management in Server Class Disks. In *Proceedings of the International Symposium on Computer Architecture (ISCA)*, pages 169–179, June 2003.

[17] HP Managment Processor. http://h71028.www7.hp.com/enterprise/cache/4230-0-0-0-121.html.

[18] H. Huang, C. Lefurgy, T. Keller, and K.G. Shin. Memory Traffic Reshaping for Energy-Efficient Memory. In *Proceedings of the International Symposium on Low Power Electronics and Design (ISLPED)*, pages 393–398, August 2005.

[19] L. Huston, R. Sukthankar, R. Wickremesinghe, M. Satyanarayanan, G.R. Ganger, E. Riedel, and A. Ailamaki. Diamond: A Storage Architecture for Early Discard in Interactive Search. In *Proceedings of the USENIX Conference on File and Storage Technonologies (FAST)*, April 2004.

[20] Intel PXA 255 Processor. http://www.intel.com/design/pca/prodbref/252780.htm.

[21] B.R. Jarvinen, C.J. Neumann, and M.A.S. Davis. A Tropical Cyclone Data Tape for the North Atlantic Basin, 1886-1983: Contents, Limitations, and Uses. Technical Report NWS NHC 22, National Oceanic and Atmospheric Administration (NOAA), 1984.

[22] K. Keeton, D.A. Patterson, and J.M. Hellerstein. The Case for Intelligent Disks (IDISKs). *SIGMOD Record*, 27(3):42–52, September 1998.

[23] Y. Kim, S. Gurumurthi, and A. Sivasubramaniam. Understanding the Performance-Temperature Interactions in Disk I/O of Server Workloads. In *Proceedings of the International Symposium on High Performance Computer Architecture (HPCA)*, pages 179–189, February 2006.

[24] R. Kumar, D.M. Tullsen, P. Ranganathan, N.P. Jouppi, and K.I. Farkas. Single-ISA Heterogeneous Multi-Core Architectures for Multithreaded Workload Performance. In *Proceedings of the International Symposium on Computer Architecture (ISCA)*, pages 64–75, June 2004.

[25] X. Ma and A.L. N. Reddy. MVSS: An Active Storage Architecture. *IEEE Transactions on Parallel and Distributed Systems*, 14(10):993–1005, 2003.

[26] D. Maio and D. Maltoni. Direct Gray-Scale Minutiae Detection in Fingerprints. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 19(1):27–40, January 1997.

[27] R. Mao, W. Xu, S. Ramakrishnan, G. Nuckolls, and D.P. Miranker. On Optimizing Distance-Based Similarity Search for Biological Databases. In *Proceedings of the IEEE Computational Systems Bioinformatics Conference (CSB)*, pages 351–361, August 2005.

[28] Micron. http://www.micron.com/.

[29] MIT Center for Biological and Computational Learning (CBCL) Face Recognition Database. http://cbcl.mit.edu/software-datasets/heisele/facerecognition-database.html.

[30] J. Mogul. TCP Offload Is a Dumb Idea Whose Time Has Come. In *Proceedings of the 9th Workshop on Hot Topics in Operating Systems (HotOS IX), USENIX Assoc.*, 2003.

[31] D. Nagle, D. Serenyi, and A. Matthews. The Panasas ActiveScale Storage Cluster: Delivering Scalable High Bandwidth Storage. In *SC '04: Proceedings of the 2004 ACM/IEEE conference on Supercomputing*, page 53, Washington, DC, USA, 2004. IEEE Computer Society.

[32] NASA World Wind. http://worldwind.arc.nasa.gov/.

[33] E. Riedel, G. Gibson, and C. Faloutsos. Active Storage for Large-Scale Data Mining and Multimedia. In *Proceedings of the International Conference on Very Large Data Bases (VLDB)*, pages 62–73, August 1998.

[34] N. Roussopoulos. Nearest Neighbor Queries. In *Proceedings of ACM SIGMOD*, pages 71–79, May 1995.

[35] M. Sivathanu, A. Arpaci-Dusseau, and R. Arpaci-Dusseau. Evolving RPC for Active Storage. In *Proceedings of the Tenth International Conference on Architectural Support for Programming Languages and Operating Systems (ASPLOS X)*, pages 264–276, San Jose, CA, October 2002.

[36] M. Sivathanu, V. Prabhakaran, F. Popovici, T. Denehy, A.C. Arpaci-Dusseau, and R.H. Arpaci-Dusseau. Semantically-Smart Disk Systems. In *Proceedings of the Annual Conference on File and Storage Technology (FAST)*, March 2003.

[37] A.W.M. Smeulders, M. Worring, S. Santini, A. Gupta, and R. Jain. Content-based Image Retrieval at the End of the Early Years. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 22(12):1349–1380, December 2000.

[38] S.M. Smith and J.M. Brady. SUSAN - A New Approach to Low Level Image Processing. *International Journal of Computer Vision*, 23(1):45–78, May 1997.

[39] Tarari Anti-Virus Content Processor. http://www.tarari.com/antivirus/index.html.

[40] M. Uysal, A. Acharya, and J. Saltz. Evaluation of Active Disks for Decision Support Databases. In *Proceedings of the International Symposium on High-Performance Computer Architecture (HPCA)*, pages 337–348, January 2000.

[41] R. Weber. SCSI Object-Based Storage Device Commands (OSD). Technical Report T10/1355-D, InterNational Committee for Information Technology Standards, July 2004.