Active Storage Revisited: The Case for Power and Performance Benefits for Unstructured Data Processing Applications

Clinton Wills Smullen, IV
CS Dept., University of Virginia
Charlottesville, VA 22904
cws3k@cs.virginia.edu

Shahrukh Rohinton Tarapore
Lockheed Martin ATL
Cherry Hill, NJ 08002
starapor@atl.lmco.com

Sudhanva Gurumurthi
CS Dept., University of Virginia
Charlottesville, VA 22904
gurumurthi@cs.virginia.edu

Parthasarathy Ranganathan
HP Labs
Palo Alto, CA 94304
partha.ranganathan@hp.com

Mustafa Uysal
HP Labs
Palo Alto, CA 94304
mustafa.uysal@hp.com

Categories and Subject Descriptors

C.1.4 [**Processor Architectures**]: Parallel Architectures

General Terms

Design

Keywords

Active storage, unstructured data, performance, power

ABSTRACT

The proliferation of digital data has resulted in a mushrooming of data-intensive applications, especially in the area of unstructured data processing. Given the growing popularity of unstructured data processing applications (e.g., Flickr^{TM}, Google Maps^{TM}), it is important to rethink system architectures to efficiently run these applications, from both the performance and power viewpoints. In this paper, we revisit active storage, which proposed offloading computation to disk drive processors, as a possible system architecture for these applications. Unlike previous work, we evaluate the *microarchitectural* aspects of active storage and perform an in-depth examination of the design of the offload processors. Using a set of unstructured data processing benchmarks, we examine two choices along the I/O path where the computation can be offloaded in existing system architectures – a disk drive processor and a disk array controller. Our evaluation demonstrates that there are interesting tradeoffs in the choice of each location and that microarchitectural enhancements to these processors can provide significant performance boosts. We show that active storage architectures can provide *large power savings*, by using lower-power processors along the I/O path, while exploiting the data-level parallelism on the storage side of the system.

1. INTRODUCTION

The past few years have seen a massive growth in the amount of digital data. The commoditization of processing and storage hardware has paved the way for the proliferation of a large number of devices, such as digital cameras, camera phones, and Blackberrys^{TM}, all of which generate data. For example, according to a recent IDC study [15], the volume of data that was captured, stored, and replicated worldwide was over 161 exabytes in 2006 (161 X 10^{18} bytes) and will grow to over *988 exabytes* by 2010, much of which is expected to be stored and managed centrally. One of the key characteristics of this data growth phenomenon is that the majority of the data is *unstructured* - in the form of images, video, audio, e-mail, etc. It is estimated that 80% of the data stored within organizations is in the unstructured form [46, 22] and that over 95% of all data worldwide is unstructured [15]. As a result of this enormous growth in the volume of data, many emerging applications will be centered around creating and manipulating unstructured data. The popularity of applications, such as Flickr^{TM}, Snapfish^{TM}, Youtube^{TM}, and Google Maps^{TM}, all of which manipulate large unstructured datasets, bear testimony to this trend. Therefore, given that unstructured data processing workloads are likely to form an important part of future workload mixes, it is very important to re-examine system architecture designs for these workloads.

Unstructured data processing workloads consist of operations that move, search, and aggregate various data items from large datasets [22], and are very I/O intensive. These applications are also amenable to data-level parallelization so that these operations can be done in parallel over different parts of the dataset. With the emergence of chip multiprocessors, such as the Sun Niagara^{TM}and Intel Core 2 Duo^{TM}, there is an opportunity to leverage the multiple cores within these processors to run unstructured data processing applications in a data-parallel fashion.

However, multi-core processors have one important constraint - I/O pin bandwidth. Data is moved between the chip and I/O devices via signaling pins that are mounted on the chip package. These pins are operated at a certain signaling rate, based on power constraints, by I/O circuitry located on the chip. Although the number of cores on the die is expected to increase in the future (thanks to Moore's Law), the number of I/O signaling pins are not [25]. Increasing the pin signaling rates to accommodate the higher I/O demands of the larger number of cores can lead to excessive power consumption by the I/O signaling circuitry, thereby making it extremely challenging to design and operate the processor chip within established power budgets [7]. Although I/O pin bandwidth might not be an important bottleneck for applications that do little I/O or whose datasets can be entirely loaded into memory, they can be a serious bottleneck for unstructured data processing applications, which sift through very large datasets and perform streaming I/O.

Moore's Law has been the primary technology driver for microprocessors, and has facilitated building progressively more powerful processors at the same cost point and power budget as their previous counterparts. One clear outcome of Moore's Law has been the advent of multi-core processors. However, Moore's Law benefits not just the host CPU, but *all* the processors in the system. As a result of improvements in processor technology, storage controllers along the I/O path (disk processors and array controller processors) have also been getting computationally more powerful over time. For example, modern storage array controllers use processors that are only a couple of generations behind those used within host machines. In the future, it is likely that we can incorporate more powerful, general-purpose processors along the I/O path at the same cost and power budgets as those used in systems today.

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. Moreover, there are opportunities to change the *microarchitecture* of these storage processors (e.g., provide support for instruction-level parallelism), which might be required to run the offloaded computation in a performance-efficient manner, and still operate these storage-side processors within their existing power budgets.

In essence, we are revisiting the "active storage" concept [6, 1, 37, 27, 21], which proposed offloading of computation on to the disk controllers. However, in contrast to these prior work, we explore active storage from the *microarchitecture* and *power* viewpoints. In our work, we consider 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, and do an in-depth microarchitectural analysis of the design of these offload processors. We show that, by carefully choosing appropriate microarchitectures for the processors on the I/O path, it is possible to achieve *large power savings* by offloading computation on to the storage processors. Since our work is similar in spirit to active storage, we refer to our designs as "active storage architectures".

To this end, we make the following contributions:

- We develop a detailed simulator and benchmark suite that captures the core set of operations used in unstructured data processing applications. Using these, we evaluate the benefits of our proposed approach and consider offloading at two points along the I/O path - the array controller and 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 in today's systems.

- We conduct a detailed design space exploration of active storage architectures, looking at both to which processor to offload computation to, as well as the microarchitectural design of the processors. 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 redesigning the architecture. 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 our architecture space and Section 3 discusses the related work. The simulation infrastructure, workloads, and experimental configurations are described in Section 4. Section 5 presents the experimental results and Section 6 concludes the paper.

2. ACTIVE STORAGE ARCHITECTURES

Our work is based on the observation that the I/O path provides a critical optimization opportunity for improving the performance of future unstructured data processing 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 in the disk drive controllers and storage array controllers. 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 [47] already point in this direction.

In addition to getting an increased number of computation cycles for free, these locations offer additional benefits for workloads. In particular, the proximity to data allows for lower latencies when accessing the data. Furthermore, since the processors on the I/O path consume less power than the host processor, offloading computation to these storage-side processors can potentially improve the energy efficiency of the overall solution.

Programming Model and Software Issues: For general computational offloading to work, there must be broad system-software support to take advantage of its capabilities. Several projects have tackled this issue and the existing proposals include: new interfaces at the subsystem level like OSD [45], searchlet/disklet interface [1, 21], and dynamic file system views [31]; techniques that move system software functions based on workload and current system load [3, 38]; operating system support to move arbitrary computation between components [19]; and techniques to bridge the semantic gap at different parts of the computer system [39]. While general purpose application processing offloading is still not in the mainstream, much of the system functionality has already benefited from offloading [34, 35].

In this paper, we assume the existence of an asynchronous RPC-like interface that allows computation to offloaded to the I/O path processors, similar to that proposed by Sivathanu et al. [38].

3. RELATED WORK

Unstructured data processing has received a lot of attention in recent years, with several industry reports pointing to its importance [46, 15]. Software platforms have been developed for building

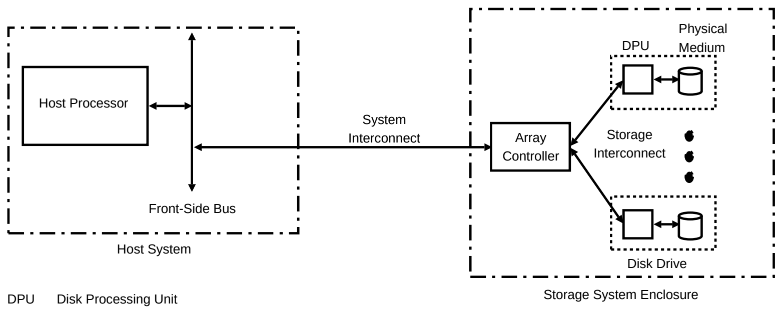


Figure 1: Architecture of the I/O Path.

unstructured data processing applications (e.g., UIMA from IBM [22]), and there are hardware appliances available in the market that facilitate searching through large unstructured datasets (e.g., Exegy TextMiner^{TM}[14], Netezza Performance Server^{TM}[43]).

Our work derives inspiration from earlier systems work on active disks which seeks to move computation close to the data it processes [1, 27, 37, 21]. 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 study to evaluate the microarchitecture and power aspects of computational offloading.

4. EXPERIMENTAL SETUP AND WORKLOADS

4.1 Simulation Infrastructure

Our evaluations are conducted using a cycle accurate execution-driven simulator that we built and validated against real (non active storage) systems. Our simulator has detailed, parametrized models for the processors, disk drives, and interconnection network, and captures the interaction between the various components, and models file system and memory management operations.

4.2 Simulated Architecture

The simulated architecture consists of three microarchitecturally heterogeneous processors: (i) host, (ii) storage array controller (AC), and (iii) 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. 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 has a bandwidth of 320 MB/s. Each disk drive in our simulated architecture consists of a DPU, DRAM, and a mechanical data transfer system (platters and disk arms). The DPU is configured as a 200 MHz Intel XScale PXA255 processor [23], which is similar to the ARM-based embedded cores used in modern SCSI disks [5]. 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, and 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
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 their values for the simulated processors and disk drive.

4.3 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 [9], which is a widely used tool for processor power estimation. Since the AC is modeled as a processor that is two generations behind the host, we assume that the two processors use two different process technologies. 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 [32]. We assume the host uses 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 [24]. 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 proposed by Kim et al [28]. We assume the disks consume their peak power as well. The power consumption of each component is summarized in Table 1.

4.4 Workloads

Unstructured data processing workloads consist of operations that move, search through, and aggregate various data items from large datasets in order to infer useful information. Of the three operations, data search and aggregation tend to be more computationally intense and therefore we focus on these two operations in this paper. The key set of data search and aggregation operations include: text searches [13], content-based searches [40], and data fusion [44]. Text searches, which include proximity searches and regular expression searches, are used in a variety of applications, such as Internet search-engines [8], intrusion-detection systems and anti-virus engines [42], and in business analytics [43]. Content-based searches are used for searching through images and audio files based on their actual content instead of using text-based tags [30]. Content-based searches are also used in song intersection [12] which is used to search for songs based on the acoustic properties of an input song, and scene completion [20] which is used to automatically search for or modify an image using semantically similar parts from other images in a dataset. Data fusion is used in urban planning and forestry [44], in MRI data analysis in medicine [10], and in popular Web-based map applications, such as Google Maps^{TM}, where road atlas data is combined with satellite images and traffic updates to provide detailed information about a certain location to the user. We have developed a set of benchmarks that capture the processing and data access behavior of these unstructured data processing operations. We now briefly describe these benchmarks:

- **Image Edge Detection:** Edge detection is a basic operation that is used in content-based search. This workload uses the Smallest Univalve Segment Assimilating Nucleus (SUSAN) [41] 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 [33]. 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.

- **Nearest Neighbor Search:** Nearest Neighbor Searches are used in text, as well as content-based searches. 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 [26]. 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.

- **Malware Search:** This workload simulates the behavior of text-based search engines, in particular, a malware scanner. 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. 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.

- **Geo-Spatial Data Fusion:** This workload implements pixel-based geo-spatial data fusion [16] on a dataset that consists of several satellite images, which we obtained from the NASA World Wind database [36]. 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 images, 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.

5. EXPERIMENTAL RESULTS

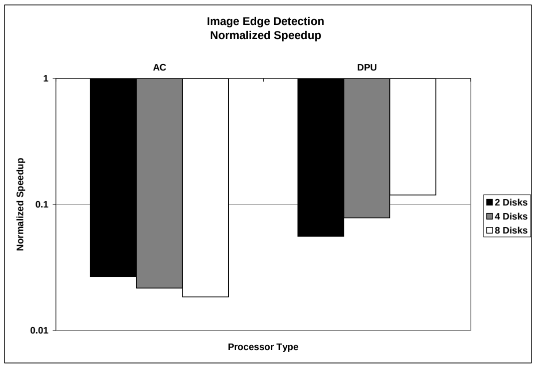
The first set of experiments quantifies the impact of offloading an application to each of the processors on the I/O path. We then conduct a detailed design-space exploration of active storage architectures, quantifying the performance and power benefits of offloading for various I/O path processor microarchitectures.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

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

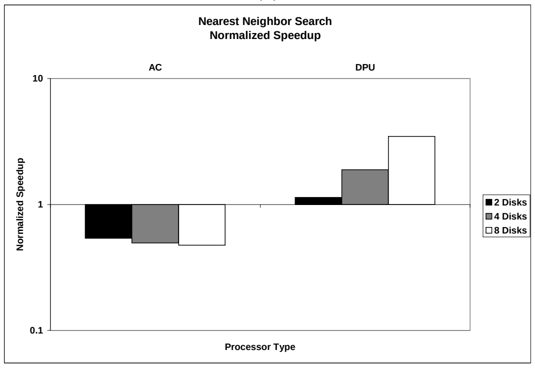
Active storage architectures offer three key advantages: (i) a reduction in the volume of data that needs to be transported to the host, (ii) freeing up resources on the host by offloading computation onto the I/O path, and (iii) speeding up computation by exploiting DLP at the disk level. The degree 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 balance all these factors carefully 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 active storage 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. 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. We use a stripe-size of 8 KB for these storage system organizations.

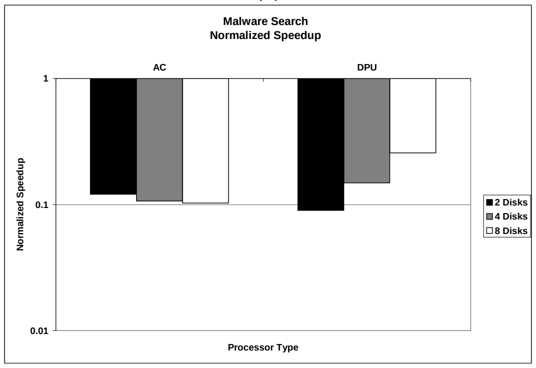
We consider two different processors, namely, AC, and the DPU, that are candidates for running computation offloaded from the host. One simple technique is to offload the entire application to one of these processors and return only the results to the host. The effect of such offloading is shown in Figure 2. For each workload, each



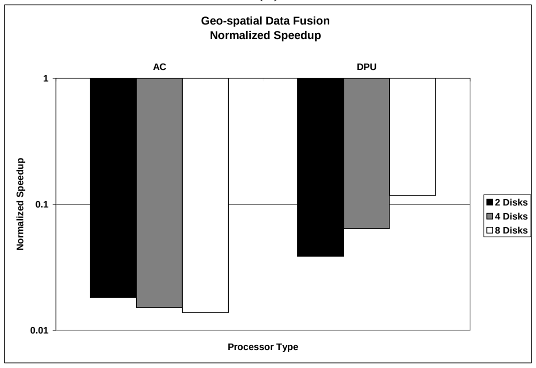
(a)



(b)



(c)



(d)

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

set of bars shows the speedup achieved when the application is offloaded to the AC and DPUs respectively. Note that the y-axis of all the graphs are in log-scale.

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 the four workloads, we observe that offloading the computation onto the AC leads to a higher performance degradation than offloading to the DPUs. The performance degradation of the AC is exacerbated when more disks are added, since the higher number of disks results in a larger amount of data that the AC needs to handle, causing it to bottleneck even further. Although the AC is computationally more powerful than a single DPU (500 MHz 2-way superscalar vs. a 200 MHz scalar pipeline), the AC suffers a higher performance loss than the ensemble of DPUs and does not provide a speedup over the baseline for any of the datapoints. On the other hand, 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 the AC processor is microarchitecturally identical to the DPU. The utilization values for the AC (U_{AC}), 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_{AC} are very small. 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. These results clearly indicate that the network is not fully saturated and therefore does not significantly influence the performance of the offloading strategies. Instead, the reason for the lower utilization of the offload processors is primarily the processing overheads. Given this result, the remainder of this study focuses on optimizing the microarchitecture of the offload processors.

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

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

5.2 Microarchitectural Analysis of the Offload Processor Design

Having seen that application performance relies more on the offload processors than on the network, 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 power consumption range as the PXA255. For example, the Hitachi SH4 [4] 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 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 to reduce their power consumption has also been proposed for the design of heterogeneous multi-core processors [29]. Assuming the AC uses the same 100 nm process technology as the host, we calculate its 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.

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

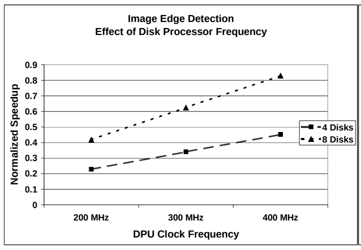
Table 3: Power consumption of DPU and AC. The power consumption of the 200 MHz DPU and 500 MHz 2-way superscalar AC used in the original configuration are 1.637 W and 44.84 W respectively.

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. Since the AC accesses data that is striped across its disks, by having fewer disks the processing on the AC can keep pace with the rate at which data flows into it from the disks. When file sizes are large and there are more disks in the array, the AC receives data from the storage system at a higher rate. 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. Therefore, when there are a larger number of disks, faster array controller processors are needed to match the higher data rates.

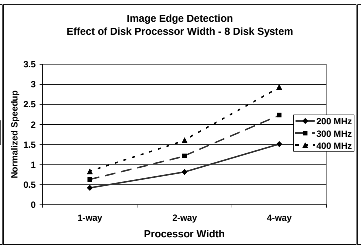
In the discussions that follow, we analyze the behavior of each application, looking at both performance and power aspects of the various offloading strategies and microarchitectural configurations. When we calculate the system power, we conservatively assume that the host, array controller, DPUs, and the electro-mechanical parts of the disk drive operate at their peak power.

5.2.1 Image Edge Detection

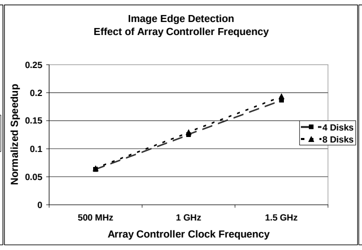
For this workload, optimizing the DPU and AC provides a speedup. For the case where the workload is offloaded to the DPU, the use of superscalar processors (shown in Figure 3(b)) almost always provides a speedup over the baseline. In the scenario where the Image Edge Detection workload is run on the AC, the 1 GHz 4-way and the 1 GHz and 1.5 GHz 8-way superscalar microarchitectures yield a speedup. 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.



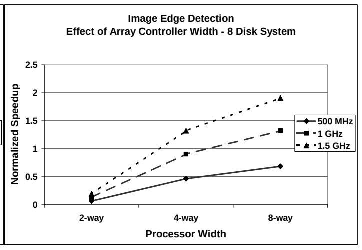
(a)



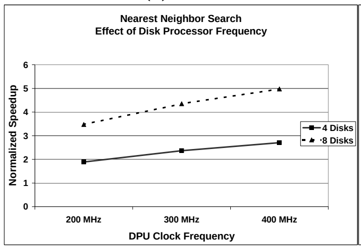
(b)



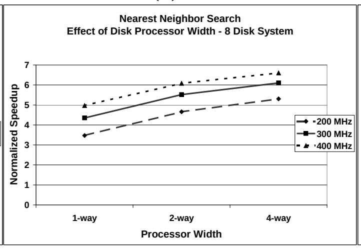
(c)



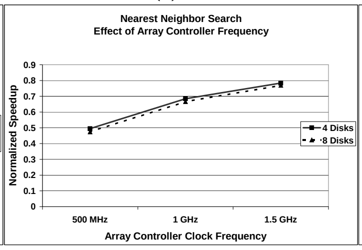
(d)



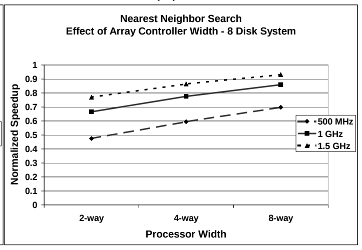
(e)



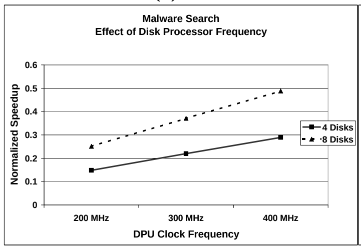
(f)



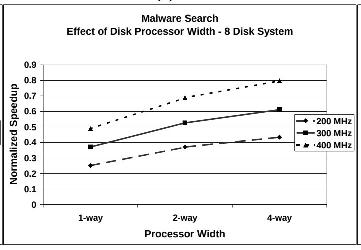
(g)



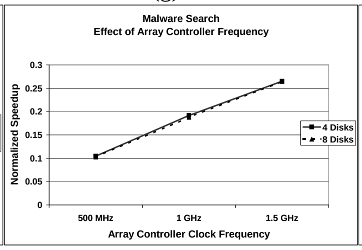
(h)



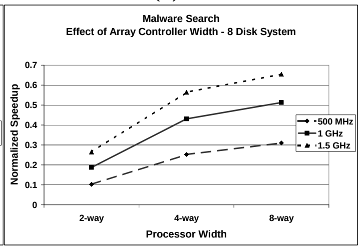
(i)



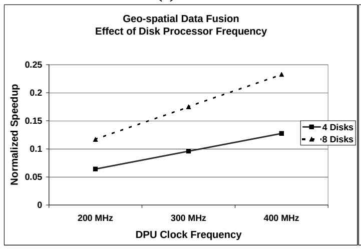
(j)



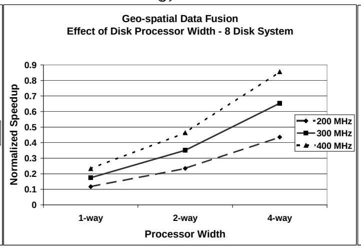
(k)



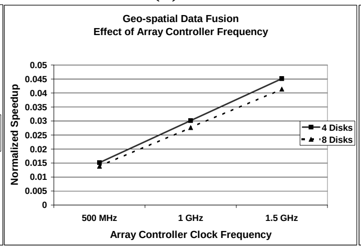
(l)



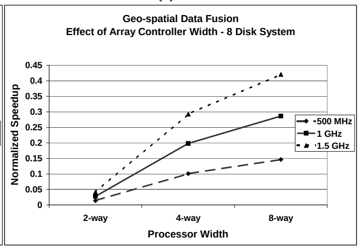
(m)



(n)



(o)



(p)

Figure 3: Design of the Offload Processor.

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.

5.2.2 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 operation 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.

5.2.3 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). 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.

5.2.4 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 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.

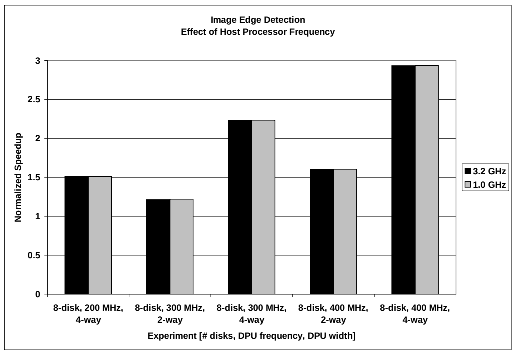
5.3 Discussion of the Results

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

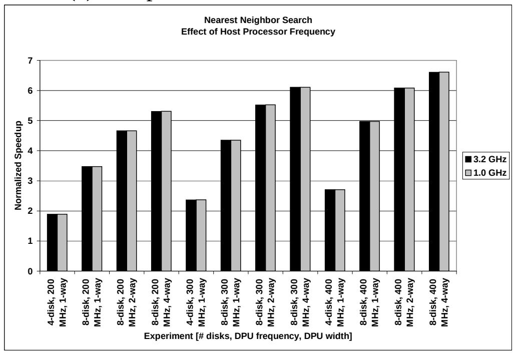
- Between DLP, ILP, and clock frequency, DLP is the most important factor to optimize. Once we have provisioned resources for extracting DLP, ILP is the next important factor, and then the clock frequency. These results motivate the use of multiple, but relatively slow DPUs, that can exploit ILP, rather 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, 27, 37], 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 [4] 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 [11, 18] 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 [17]. 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.

5.4 Reducing The Host Power Consumption

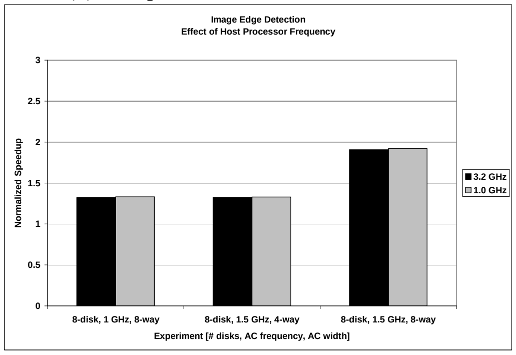
Active storage architectures provide the opportunity to offload computation, normally done on the host, onto processors on the I/O path. Since offloading lowers the utilization of the host processor, we have the opportunity to reduce the power consumption of



(a) Computation Offloaded to the DPUs



(b) Computation Offloaded to the DPUs



(c) Computation Offloaded to the AC

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

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.

6. 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 two architectural choices to offload computation along the I/O path: (i) disk array controllers, and (ii) disk controllers. We find that there are significant 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, with architecture support, active storage is a good solution for building energy-efficient systems for unstructured data processing applications.

7. ACKNOWLEDGEMENTS

We would like to thank Mary Baker and Jeremy Sheaffer for their inputs. This research has been supported in part by NSF CAREER Award CCF-0643925, NSF grants CNS-0551630, CNS-0627527, and gifts from Intel, HP, and Google.

8. REFERENCES

[1] A. Acharya, M. Uysal, and J. 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] K. Amiri, D. Petrou, G. Ganger, and G. Gibson. Dynamic function placement for data-intensive cluster computing. In *Proceedings of the USENIX 2000 Annual Technical Conference*, San Diego, CA, June 2000.

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

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

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

[7] S. Borkar. Thousand Core Chips - A Technology Perspective. In *Proceedings of the Design Automation Conference (DAC)*, pages 746–749, June 2007.

[8] S. Brin and L. Page. The Anatomy of A Large-Scale Hypertextual Web Search Engine. In *Proceedings of the World Wide Web Conference (WWW)*, pages 107–117, April 1998.

[9] 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.

[10] R. Bryant. Data-Intensive Supercomputing: The Case for DISC. Technical Report CMU-CS-07-128, School of Computer Science, Carnegie Mellon University, May 2007.

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

[12] M. Casey and M. Slaney. Song intersection by approximate nearest neighbor search. In *Proceedings of the International Conference on Music Information Retrieval (ISMIR)*, pages 144–149, October 2006.

[13] R. Chamberlain, M. Franklin, and R. Indeck. Exploiting Reconfigurability for Text Search. In *Proceedings of the Workshop on High Performance Embedded Computing (HPEC)*, September 2006.

[14] Exegy TextMiner (Whitepaper). http://www.exegy.com.

[15] J. Gantz, D. Reinsel, C. Chute, V. Schlichting, J. McArthur, S. Minton, I. Xheneti, A. Toncheva, and A. Manfrediz. The Expanding Digital Universe - A Forecaset of Worldwide Information Growth Through 2010, March 2007. IDC Whitepaper.

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

[17] S. Gurumurthi. Should Disks be Speed Demons or Brainiacs? *ACM SIGOPS Operating Systems Review Special Issue on File and Storage Systems*, 41(1):33–36, January 2007.

[18] 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.

[19] J. Hartman, P. Bigot, P. Bridges, B. Montz, R. Piltz, O. Spatscheck, T. Proebsting, L. Peterson, and A. Bavier. Joust: A platform for liquid software. *Computer*, 32(4):50–56, 1999.

[20] J. Hays and A. Efros. Scene completion using millions of photographs. *ACM Transactions on Graphics (SIGGRAPH 2007)*, 26(3), 2007.

[21] L. Huston, R. Sukthankar, R. Wickremesinghe, M. Satyanarayanan, G. 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.

[22] IBM Unstructured Information Management Architecture. `http://www.research.ibm.com/UIMA/`.

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

[24] Intel PXA255 Processor - Electrical, Mechanical, and Thermal Specification, February 2004. http://www.intel.com/design/pca/applicationsprocessors/manuals/278780.htm.

[25] International Technology Roadmap for Semiconductors - 2006 Update, 2006. http://www.itrs.net.

[26] B. Jarvinen, C. Neumann, and M. 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.

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

[28] 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.

[29] R. Kumar, D. Tullsen, P. Ranganathan, N. Jouppi, and K. 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.

[30] T. Lehmann, M. Guld, C. Thies, B. Fischer, D. Keysers, M. Kohnen, H. Schubert, and B. Wein. Content-based Image Retrieval in Medical Applications. *Methods of Information in Medicine*, 43(4):354–361, 2004.

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

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

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

[34] 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.

[35] 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.

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

[37] 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.

[38] 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.

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

[40] A. 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.

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

[42] P. Szor. *The Art of Computer Virus Research and Defense*. Addison-Wesley, 2005.

[43] The Netezza Performance Server System. http://www.netezza.com/products/products.cfm.

[44] A. Waxman, D. Fay, B. Rhodes, T. McKenna, R. Ivey, N. Bomberger, V. Bykoski, and G. Carpenter. 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.

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

[46] C. White. Consolidating, Accessing, and Analyzing Unstructured Data, December 2005. Business Intelligence Network article.

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