On the performance and use of dense servers

W. M. Felter
T. W. Keller
M. D. Kistler
C. Lefurgy
K. Rajamani
R. Rajamony
F. L. Rawson
B. A. Smith
E. Van Hensbergen

Dense servers trade performance at the node level for higher deployment density and lower power consumption as well as the possibility of reduced cost of ownership. System performance and the details of energy consumption for this class of servers, however, are not well understood. In this paper, we describe a research prototype designated as the Super Dense Server (SDS), which was optimized for high-density deployment. We describe its hardware features, show how they challenge the operating system and middleware, and describe how we have enhanced its software to handle these challenges. Our performance evaluation has shown that dense servers are a viable deployment alternative for the edge and application servers commonly found at conventional Web sites and large data centers. Using industry benchmarks, we have shown that SDS outperforms a comparable traditional server by almost a factor of 2 for CPU-bound electronic commerce workloads for the same space and roughly equivalent power budget. We have observed the same advantage in performance when SDS is compared to the alternative solution of virtualizing a high-end server to handle "scaled-down" workloads. We have also shown that SDS offers finer power management control than traditional servers, allowing higher energy efficiency per unit of computation. However, for high-intensity Web-serving workloads, SDS does not perform as well as a traditional server when many nodes must be configured into a cluster to provide a single system image. In that case, the limited memory of each SDS node reduces its performance scalability, and a traditional server is a better alternative. We have concluded that until technology advances allow denser packaging of memory or more efficient use of memory across nodes, the best performance and energy efficiency can be obtained by heterogeneous deployment of both traditional high-end and dense servers.

1. Introduction

Recently, several companies have begun to offer dense servers containing components designed for mobile computing systems [1–4].[1] Typically, such servers contain a low-power x86 processor, designed for mobile computers, generally clocked at frequencies between 300 MHz and 1 GHz, with memory ranging from 128 MB to 1 GB, up to 40 GB of disk storage, and one to three 100-Mb/s Ethernet connections. Some of these servers are configured as single-purpose appliances, while others are intended for general use. However, little is understood about their performance, their energy efficiency, or the application classes for which they are best suited. For

[1] The IBM BladeCenter* product offering [5] uses a different approach for improving server density and employs traditional server components rather than mobile computing technology.

0018-8646/03/$5.00 © 2003 IBM

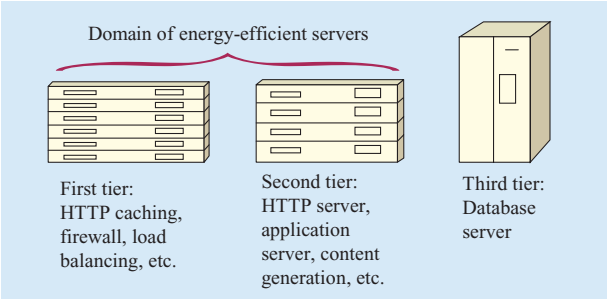

Figure 1

Three-tiered structure for a Web site.

instance, there is a perception that the limited processing resources available on these servers invariably lead to poor performance compared with high-end servers. Additionally, their energy efficiency is difficult to quantify given the lack of standard and agreed-upon metrics. This paper addresses these issues, with a focus on the performance, energy efficiency, and class of applications for which these servers are well suited. We also describe the additional operating system and system management support that is needed to overcome some of the perceived performance and management problems with dense servers.

In Section 2, we provide background material and describe our prototype implementation of a dense server. Section 3 describes the necessary operating system and system management support, and Section 4 provides the bulk of the quantitative performance analysis. In Section 5, we provide a further, qualitative analysis of our research prototype. We discuss future work in Section 6 and related work in Section 7. Section 8 concludes the paper.

2. Dense servers

The case for energy efficiency in servers

There is a growing industry trend to outsource computing services to large data centers accessible through the Internet. These data centers use economies of scale to amortize the cost of ownership and system management over a large number of servers—typically hundreds or thousands, densely packed to maximize floor space utilization, providing the customer with a more cost-effective approach than the alternative of operating the same services in-house.

Large-scale deployment, however, pushes the limits of power-supply and cooling systems in data centers. Power and cooling costs are already a significant portion of total operating costs—as high as 25% for some data centers. Intermittent system failures due to insufficient cooling in densely packed data centers also add to operating costs. Furthermore, in many data centers, the power supply to the server racks is a key inhibitor to increasing server density, with a practical limit ranging from 5 to 7 kW per rack. This amount of power is often insufficient to allow the rack to be fully populated with servers, thus exacting a loss of revenue due to under-utilization of space. Dense servers solve these problems.

Built using components designed for low-power operation, dense servers are suitable for environments that require a high level of deployment density without violating power-supply limits or generating excessive heat. They also enable sophisticated cluster-based power management techniques at a fine-grained level of control [1, 6–8], further reducing energy consumption. While the use of low-power components such as mobile processors implies a reduction in the computing resources at each server node, the resulting reduction in power and heat enables more aggressive packaging that increases the density of server deployment for a given volume and power budget. Thus, dense servers trade the level of performance at each server node for a larger number of servers that can be deployed within the same space. In Section 4, we examine the impact of this tradeoff on system performance and power consumption.

Where do dense servers fit?

Computing services accessible through the Internet are typically organized in a three-tiered structure, as shown in **Figure 1**. The first tier consists of an interface to the network, including routers, load balancers, firewalls, and Web servers, among others. Servers deployed in this tier are often referred to as the "edge-of-network servers," or simply "edge servers." The second tier consists of application servers that implement a rich user interface, data presentation, and user interactions with the service. Several technologies exist to implement this tier, including the IBM* WebSphere* [9], the Oracle** Application Server [10], and the Microsoft** .NET [11]. The second tier also connects the first and third tier, with the latter implementing a data warehouse using traditional database technology [12]. The typical workload for a first- and second-tier server is highly parallel, consisting of many independent threads of execution. These workloads are well suited to cluster architectures, in which work is transparently distributed to a set of independent machines. This makes dense servers reasonable candidates for deployment in these two tiers, given that they offer a higher density of processing nodes that can exploit this parallelism.

On the other hand, the database workload on the third tier usually requires frequent inter-thread synchronization to ensure data coherence, and traditional symmetric

multiprocessor servers are better suited for these applications than are dense servers [12]. Therefore, we focus on evaluating dense servers using benchmarks that approximate the typical workloads in the first and second tiers of a Web host.

The Super Dense Server

We have developed a research prototype designated as the Super Dense Server (SDS), which consumes a maximum of 13 W during operation. The SDS consists of a custom-designed board, often referred to as a "blade," that plugs into a standard CompactPCI** [13] backplane. The physical and logical design of the SDS blade is shown in **Figure 2**. The blade contains an Intel** Ultra Low Voltage (ULV) Pentium** III processor,[2] which has a 256K L2 cache and uses SpeedStep** technology [14] to adjust its speed from 300 MHz to 500 MHz. It also includes up to 512 MB of double data rate (DDR) 266 synchronous dynamic random access memory (SDRAM), a universal serial bus (USB) port, two 100-Mb/s Ethernet (designated as Enet in Figure 2) connections along with their physical transceivers (designated as Phy in Figure 2), and a highly integrated Silicon Integrated Systems (SIS) 635 chipset[3] that provides the memory controller, bus controllers, and an integrated drive electronics (IDE) disk interface. Since the chipset was originally intended for mobile devices, it provides a variety of power-saving operation modes from low-latency processor sleep modes to full system hibernation. The SDS blade contains a peripheral component interconnect (PCI) bridge allowing it to interact with standard blades across the CompactPCI backplane. It also contains a service processor that can be controlled remotely through an inter-integrated circuit (I2C) bus. The service processor is responsible for blade power-on, power-off, and hot-swap isolation sequences and supports system management functions.

Up to 32 blades can be mounted in a CompactPCI chassis that supplies power, cooling, and backplane connections to the blades and support hardware mounted in the chassis. The chassis is a 6-U enclosure, where "U" is the unit of height (1.75 inches) for rack-mounted components. Standard data center equipment racks can accommodate 42 U of components. We designed the SDS blade to be 3 U high, and the connections in the backplane of the chassis were slightly modified to allow two blades to be mounted into a single 6-U-high backplane slot. The CompactPCI backplane carries the Ethernet, I2C, and power connections for the blades, eliminating any need for cables to them. We do not currently use the backplane PCI bus. We modified the connections in the backplane slightly in order to accommodate the half-height blades, since the standard connections are for 6-U-high blades. We chose this packaging format to reduce development time and expense, since it eliminated the need to design a custom mechanical enclosure, reduced the cabling requirements, and allowed us to leverage standard system management and network switching components.

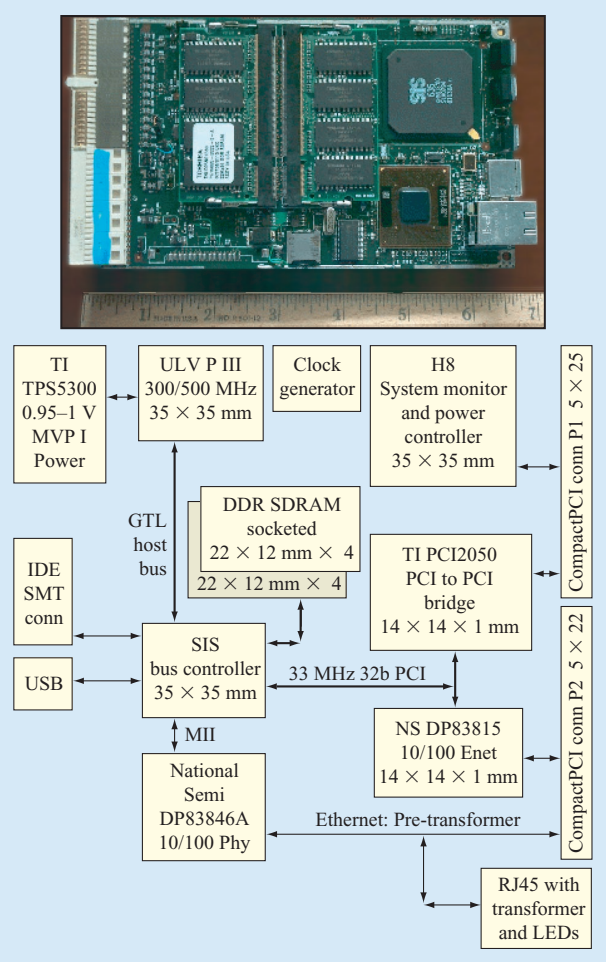

Figure 2

An SDS server blade.

Figure 3 shows a diagram of the resulting configuration. The standard 6-U-high chassis contains slots for two Ethernet switch blades, two system management blades, and four independent power-supply cards. The remaining slots in the chassis are used to hold SDS blades. The Ethernet connections provide the communications among the blades through the Ethernet switches, which in turn connect the chassis to the external network. The I2C bus connections allow the system management blades to

[2] Intel Corporation, Santa Clara, CA.
[3] Silicon Integrated Systems Corporation, Sunnyvale, CA.

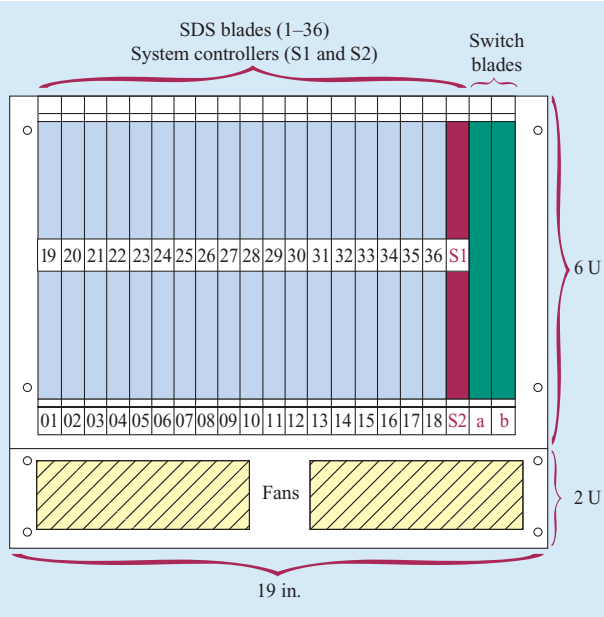

Figure 3

CompactPCI chassis configuration.

communicate with the service processors on each of the blades for control and management.

Although resource constraints allowed us to build only 11 working blades, our design point potentially allows up to 360 systems per standard rack of 42-U height, 19-inch width, and 30-inch depth, including the necessary cooling fans, system management blades, power supplies, and Ethernet switches. The SDS uses conservative packaging for cost reasons, but a more aggressive design could substantially increase the implementation density. The maximum power rating for the rack is within 7 kW, where 5.7 kW of power is allocated for the servers and the remaining power operates the fans, the Ethernet switches, and the power supply. Our design yields a density of 8.57 server blades per 1 U. In raw processing terms, this translates to 180 GHz of x86 processing, 184 GB of main memory, 71.4 Gb/s of Ethernet bandwidth, an aggregate of 92 MB of L2 caches, and 360 independent memory and I/O buses. To put this density in perspective, a rack of 42 traditional high-performance servers that fit in the same space and use contemporary server technology can accommodate raw processing resources of up to 100 GHz of x86 processing, 168 GB of main memory, 84 Gb/s of Ethernet bandwidth, an aggregate of 42 MB of L2 caches, and 42 independent memory and I/O buses.

Our system differs from other dense server designs [1–4] in two important aspects: It does not contain a disk or keyboard, video, and mouse (KVM) connections. Eliminating the disk provides four important advantages. First, it reduces maintenance costs because we eliminate the only source of mechanical failure on the blade. Second, it allows the hardware designer to focus on increasing the design density. Third, it conforms with trends in data centers of consolidating storage into an independent storage area network (SAN) or network attached storage (NAS) for improved reliability and simplified storage management. Fourth, it allows the easy re-tasking of a particular blade by simply pointing its root file system to a different external partition during boot. The elimination of KVM connections also offers several important advantages. The first of these is elimination of cables, which reduces cost, improves reliability, and simplifies server installation and management. The second advantage is improved server density, since the physical connectors and support chips for KVM devices would otherwise reduce space for other components. Elimination of KVM connections is consistent with typical server deployments in modern data centers, where the video capabilities of a server are never used. In fact, the KVM connections seem to be used only when the system administrator has to walk to a malfunctioning machine and connect I/O devices to reboot or reinstall software, which is not a viable solution with the increased blade density.

Implications for operating system and middleware

The hardware design point that we chose for the SDS has several implications that affect systems software. Some of these constitute challenges, such as increased system management requirements and the performance implications presented by the lack of a local disk and relatively small processing power and memory on each prototype blade. Other implications include the need to overcome the lack of KVM connections and the need to implement full console functionality for debugging and management. There is also the opportunity to exercise fine-grained control over power management.

Of these challenges, system management and performance are crucial to establishing these servers as viable alternatives for Internet sites. System management is complicated by the increased density, the need for scalable algorithms, and the desire to have a central point of control that allows the system administrator to easily manage the cluster. Performance is also affected by the need to do swapping across the network and by the need to carefully manage the limited resources available on each node. Inefficiencies in the software are likely to have more serious effects, and relatively sophisticated middleware is needed to enable the blades to function as a single system image with scalable performance.

3. Operating system support

In this section, we describe the operating system support for the SDS. This support adapts the operating system to the lack of traditional hardware features such as disk storage or KVM connections, reduces the complexity of system management, improves performance, and manages power consumption.

*Linux** DSA*

A key enabler for SDS is a *server-class* operating system for a node without a local disk or console. While there are many real-time and client operating systems for hardware without a local disk or any disk access at all, it has generally been assumed that all server systems have a locally attached disk, and a number of operating system and standard subsystem functions such as boot, shutdown, swapping, and system management depend on its existence. During the early 1980s, the developers of diskless workstations encountered a similar set of assumptions and reacted by developing variants of UNIX** for the diskless workstation environment. Although, as described in Section 7, our approach to diskless operation is somewhat different from that used for diskless workstations, it incorporates a number of the same ideas.

We have constructed an operating system environment suitable for diskless servers based on the Red Hat 7.1 Linux distribution and the Linux 2.4.17 kernel.[4] We designate this environment as the Linux Diskless Server Architecture (Linux-DSA), and it runs on each blade. This environment was supported by a *MetaServer*, a Linux configuration based on Red Hat** 7.1 Linux that we constructed to run on a CompactPCI system management blade from Ziatech** [15]. This management blade, equipped with a low-power, 20-GB laptop disk, is dedicated to supporting Linux-DSA on the server blades. A laptop disk is used instead of a standard server-class disk to reduce power consumption. Later in this section we describe the techniques used to minimize the space consumed by the system images stored on the MetaServer disk. One MetaServer system management blade is capable of supporting all 36 server blades in an enclosure.

The MetaServer provides several functions in support of the SDS blades. It acts as a dynamic host configuration protocol (DHCP) server that assigns a fixed IP address to each blade on the basis of its media access control (MAC) address, giving the blade an infinite lease. In addition to the IP address, the DHCP server informs the blade which kernel to boot and what to mount as the root file system. It also serves as a boot server for each of the server blades, transmitting the kernel using the standard trivial file transfer protocol (TFTP). For storage access, we used the Network File System (NFS) for root and other system-related file access, and a new network block driver that we have developed, called the Ethernet block driver (EBD) [16], for swapping. EBD implements a high-performance protocol for block data transfer that is optimized for modern high-speed local area networks. We have used this protocol successfully to support the performance-critical swapping from the server blades running Linux-DSA to and from the MetaServer or an alternative swap server. Finally, the MetaServer is also the master of the I2C bus, allowing it to collect system status from individual blades, issue system resets, select blade operating frequency, and individually power on or off each blade. These features are critical to our cluster-based power management techniques, which are described later in this section.

File system layout

To simplify system management and software maintenance while ensuring a uniform level of software across all of the blades, Linux-DSA uses a file system layout that maximizes the amount of standard Linux file space that is shared by all of the blades. Files stored on the MetaServer that are used by the diskless blades are kept in a separate subtree, `/dsa`. Rather than maintain a separate root file system image on the MetaServer for each blade, files are installed only once in a common root file system, `/dsa/root`. The root file system used by the blades during runtime is a standard Red Hat Linux 7.1 root file system image. It is mounted on a non-root directory on the MetaServer and is exported using NFS and is mounted read-only by the blades. Thus, the vast majority of the standard system and subsystem binaries, libraries, and configuration files are common across all blades. This greatly simplifies system management, especially software maintenance and upgrades.

However, it is necessary to maintain separate versions of certain files because they contain configuration information that is specific to a particular blade. Each blade also requires its own copy of certain files written during normal system operation such as logs, subsystem lock, and process ID (PID) files. Since the majority of these files are already under the `/var` directory, Linux-DSA uses a separate `/var` file system for each blade, which the MetaServer exports read–write to the particular blade. The `/var` file systems are configured as directories within a single physical file system on the MetaServer. There are a few cases in which the standard Linux implementation writes into a file in the root file system. Most of these are easily handled using symbolic links. However, most of the configuration files, including most of those found in the `/etc` directory hierarchy, are common across all of the blades. Finally, `/tmp` space is also provided by the `/var` file system by linking `/tmp` to `/var/tmp`.

[4] Red Hat Corporation, Raleigh, NC.

There are several aspects of our use of NFS that are worth noting. First, the current Linux implementation requires that the root file system of the blades be owned by the root user ID on the MetaServer system; this is clearly wrong, but Linux NFS does not support user ID or group ID relativization, which would allow us to have a non-root user own the `/dsa` subtree but have it exported to the blades as root. Second, NFS is often considered to have performance and scalability problems. However, for the purposes of supporting a number of server blades running a typical server workload, the number of root file system references is quite low and well within the capacity of our server. Moreover, although there is more logging activity, logging to the `/var` over NFS has never been the performance bottleneck in our benchmarking. Third, the use of the shared read-only root is critical in our installation; the laptop drive on the system management blade simply does not have space for a full, separate read–write root for each server blade. Moreover, the use of NFS has the advantage that all of the `/var` file systems are actually directories within a single file system on the system management blade, and thus draw new blocks for allocation to the files being written from a common pool of free blocks. This has the advantage of making better use of the available disk space.

Boot and shutdown

The combination of diskless operation and a well-defined configuration tremendously simplifies the boot and shutdown procedures. As indicated above, Linux-DSA obtains its IP address using a kernel-level DHCP client during initialization before mounting the root file system over NFS. Although kernel-level DHCP-based IP configuration is a standard feature of Linux 2.4.17, we used a different implementation that simplified the code and was more easily extended to pass additional configuration information using DHCP. Since the blade is a known hardware environment and in practice has only a single network controller to support, the kernel was statically configured with the correct network driver and no other device drivers. This eliminated the boot-time delays associated with device probing. During the user-level portion of initialization, the blade, on the basis of the host name given it by DHCP, determines which `/var` file system to mount and sets up any blade-specific configuration needed.

Shutdown is also very simple and very fast. Since the blade is diskless and has no nonvolatile state, it can simply terminate the processes that are currently running, unmount the file systems, and either turn itself off or reboot. No lengthy file system synchronization is required. The MetaServer and any data servers maintain the integrity of the file system meta-data.

Installation, upgrade, and management

Wherever possible, we used standard administrative commands and management features of Red Hat Linux. However, in contrast to standard server systems or even other diskless Linux implementations, many administrative commands are run on the MetaServer rather than on the blades themselves, and thus they are run only once, rather than once for each blade. For example, to install and manage software packages in the file system image in `/dsa/root`, we used the standard `rpm` command, which has a parameter that allows it to operate on an alternative root file system. For those commands that do not offer the flexibility needed by our environment, we have written special scripts to perform the required function. Although the existing `chroot` function provides a partial solution to the problem, a better, more general way to deal with it was to use private name spaces such as those in Plan 9 [17]. Changes to the common configuration files affect all of the blades rather than a single one. Blade-specific configuration files are located in the `/var` file system of a blade and, if necessary, are symbolically linked from their standard location in the root file system.

Using a single disk image of the system software that runs on the blades effectively eliminates two problems that plague disk-based, clustered server systems—the difficulty of ensuring that all of the nodes in the cluster run the same software and the time and effort required to perform a separate software install on each node. However, there are environments in which different blades may require different versions of the system software; this can occur, for example, because an installation has incrementally upgraded the blades from an older to a newer level of a set of software packages. Although there are good reasons to do such upgrades one enclosure or one MetaServer at a time, it is quite easy to support multiple system images with a single MetaServer. The name of the `/dsa` directory that contains the system images for the blades is only a convention; by changing the DHCP configuration file and the file that maps individual blades to their associated `/var` file systems, a system administrator may have different blades use different system images. Once this is done, the administrator can make any alterations required to the request distribution mechanism used by the cluster and restart any blades that are to run the new software; the major cost of this is the disk space consumed by the additional system images.

Swapping

Much of our work with Linux-DSA has been done without enabling swapping by running entirely within memory. However, this is impractical for certain workloads. Normally, if we do use swapping with a Linux-DSA system, we set up the MetaServer to act as the swap server

as well by running the EBD server on it with a separate disk partition for the swap space of each blade. However, because of the limited capacity of the disk on the Ziatech board, we used a separate Linux-based swap server for our performance evaluation.

Blade monitoring

Although Linux-DSA supports the standard `sar` and `sadc` programs, it also provides a special performance-monitoring feature that collects performance and utilization data from the blade and sends it, using user data protocol (UDP) packets, to a system-monitoring application. From an architectural perspective, the system-monitoring application should run on the MetaServer for the blades, but in practice we run it on a different machine because the MetaServer is too limited to handle this function in addition to its other responsibilities. This mechanism acts as a distributed replacement for the `sar` and `sadc` and allows the system-monitoring application to collect information about the individual blades and combine it into an overall view of the resource utilization of the blades as a whole. The mechanism is also used to provide the monitoring data used by the request-distribution mechanism described in the section on power-aware request distribution.

Other system software enhancements

Beyond Linux-DSA and the MetaServer infrastructure that it requires, we have also implemented the following:

- A mechanism for distributing requests coming from the network across the blades in such a way as to reduce overall power consumption.
- Boot-time firmware that supports very fast boot of our prototype blades.
- Network-based console and logging support.
- Distributed file caching to optimize the use of blade memory when running Web-oriented workloads.

Power-aware request distribution

We implemented a power-aware request distribution (PARD) policy to distribute service requests to the minimum number of servers in the cluster required to maintain the desired quality of service, allowing the remaining servers to be turned off to reduce power consumption. Each blade runs the performance-monitoring feature described above to periodically report its resource utilization to a PARD server. Although this server could be the same system as the MetaServer, as indicated above, in our prototype it is a separate machine. Our PARD server implementation [18] is based on the Linux Virtual Server (LVS) [19] which has been extended with the PARD algorithm to direct the load balancing. The PARD server turns idle blades off by requesting the MetaServer blade to send a power-down command to the server blades over the I2C bus. The policy for power management itself is relatively simple and uses the aggregate workload and well-defined thresholds to decide which machine to turn on or off. We used a connection-counting approach in PARD to estimate the current workload; that is, the incoming workload is estimated by the total number of active connections established with the blades in the cluster. Prior measurements were used to determine the capacity of each blade, or *connection_capacity*, which is the maximum number of connections possible while maintaining an acceptable quality of service (response time) and allowing for estimated spurts in workload over short time intervals. Overprovisioning the *connection_capacity* allows for power-up delay as newly activated blades join the cluster and start providing service. PARD steers requests only toward blades that are currently powered on, and it selects the blade for each new connection as the one with the lowest number of current connections.

PARD uses the LVS to do the request steering, running it in direct routing mode so that incoming packets go through the LVS distributor but outgoing packets from blades go directly back to the client. Since the incoming request packets are small, LVS is not a bottleneck for our system. The pre-calibrated *connection_capacity* is used to estimate the number of blades required to serve the current workload, and PARD turns blades on as necessary. When the number of requests to the cluster decreases so that fewer blades are required, PARD selects a blade to turn off, stops sending requests to it, waits until there are no connections to it, and then directs the MetaServer blade to turn it off. More sophisticated policies can be implemented [1, 6, 8, 18].

Although we have not experimented with blade failure detection and recovery, the combination of the blade monitoring, PARD, and the rapid reboot of our blades provides the mechanisms needed to support them. If a blade does not provide monitoring information to the PARD server within a suitable timeout, the PARD server can assume that the blade has failed and mark it as unavailable to receive new requests. Using the MetaServer blade, the PARD server can then force the blade to reboot. When the blade again begins providing monitoring information to the PARD server, the PARD server detects that it is running and marks it as available to receive new requests. If the blade fails to reboot successfully after some number of tries, the PARD server can declare it broken and notify the system administrator of the problem.

Boot-time firmware

We used a version of Linux BIOS [20] to initialize the blade hardware and execute the DHCP protocol to obtain

the Linux-DSA kernel image to be booted on the blade. With Linux BIOS, our SDS blades complete the boot process much faster than a traditional server. Even with the Linux-DSA boot sequence over the Ethernet, a typical boot from initiation to an operational Web server takes about twenty seconds. This rapid boot time dramatically reduces the time needed to recover or reconfigure a blade server. It also benefits our PARD policy, since it reduces the amount of extra capacity that must be kept in the active state to handle rapid increases in workload. We also considered fully embedding the OS in electrically erasable programmable read-only memory (EEPROM), which would have reduced boot time to only five seconds, but rejected this alternative because it would require reprogramming the EEPROM for every kernel change. One limitation of our implementation of Linux BIOS and Linux-DSA is that the blade has to run DHCP twice, once at the BIOS level to get the kernel and once at the kernel level to configure the IP information: It would be better to pass the IP information from the BIOS to the kernel and run DHCP only once.

Console and logging over Ethernet

To reduce space and cabling requirements, the SDS blades do not have standard KVM or serial ports. To support normal console access to the SDS blades, we developed *console over Ethernet*, which extends the recently developed Linux netconsole feature [21]. Here we provide an overview of our console over Ethernet support; a detailed description is available in [22].

We have extended netconsole in three important ways. First, console over Ethernet is built into the kernel and enabled at boot time, and thus starts operation much sooner than netconsole so that messages generated during system boot can be observed remotely. Messages generated prior to network activation are buffered and transmitted once the network is available. We also added console over Ethernet support to our Linux BIOS code to relay power-on self-test and BIOS messages over Ethernet. This is useful for fault isolation. Second, we have added `tty` support, so that we can connect to the system console through a slightly modified terminal emulator program that allows full console input and output, including special console key combinations, escape sequences, and `tty` functions. Third, while netconsole transmits messages as UDP packets, our implementation sends messages using a special Ethernet frame type. This means that console over Ethernet does not depend on the correct configuration and operation of the full network stack. A console monitor program receives and displays messages contained in these special Ethernet packets and allows us to selectively monitor any particular blade or all of them.

Much of the disk output generated by the SDS blades during normal operation goes to system and application log files. In the Linux-DSA environment, these log files reside in the per-blade `/var` file system on the MetaServer and are accessed by the blades using NFS. This allows each blade to have its own set of log files. Postprocessing techniques can merge these files into a single set if desired. With the exception of the Web-server logs, most of this log information passes through the `syslog` daemon, which can alternatively be configured to filter and relay the log messages over the network.

Distributed file cache

To remain cost-competitive with traditional servers, blade servers must contain significantly less memory per server. This can limit the performance of blade servers when running memory-intensive applications such as static Web serving for large Web sites. To address this issue, we implemented a set of Linux file cache and Web server modifications to maximize the effectiveness of the limited amount of on-board memory.

The primary goal of our distributed file cache design was to coordinate the contents of the file cache on the blades within a blade cluster to improve the cache hit ratio for applications such as Web servers. The file system containing the content to be cached is accessed through the Linux network block device (NBD) [23]. Additionally, each blade also makes the dataset available to every other blade in the cluster through NFS. We modified the TUX Web server [21] to hash and rename incoming requests so that a particular request is always accessed through one SDS blade in the event of a local file cache miss; when a miss occurs, TUX uses NFS to obtain the file from its associated blade. We also augmented the Linux file cache so that one of three rules would be applied to each file in the file cache: never evict, aggressively cache, aggressively evict. An interface we provide through the `/proc` file system is used to associate files with rules. Then we modified TUX to use this mechanism to prevent large files from displacing smaller, more popular files that it wants to retain in the file cache.

4. Experimental analysis

This section describes our evaluation of our prototype hardware and software.

Benchmarks

Industrial benchmarks for three-tiered Web sites focus on server throughput subject to conformance to a specified maximum response time [24, 25]. These benchmarks are designed to represent high-intensity workloads that stress the capacity of a Web site. The response-time requirement corresponds to the quality of service stipulations in typical service-level agreements. Following standard research practice, we use modified versions of industry-standard

benchmarks[5] to drive our experimental evaluation. These benchmarks are widely available and understood, allowing others in the field to put our results in a familiar context. While there may be some debate on whether industrial benchmarks represent actual workloads, they nevertheless serve as a fair vehicle for comparing different designs and implementations under the same conditions.

Our first benchmark, ⟨tpc-w⟩, was a modified version of the e-commerce benchmark, TPC-W**, from the Transaction Processing Council [24]. TPC-W is a benchmark for evaluating sites supporting e-commerce and is modeled after an online retail bookstore. We used ⟨tpc-w⟩ to evaluate the different server alternatives for the application server tier. Our implementation was based on the TPC-W specification, with a few modifications to provide us with a general-purpose tool for testing our servers running work typical of the application server tier. We used the open-source database software MySQL [26] v3.23.49a as the database server and Apache [27] v1.3.23 with PHP [28] v4.1.2 as the content-generation module. We used the in-kernel TUX Web server/cache [29] for serving the static image content. We used query result caching [30] on the database server and the APC [31] extension to PHP for script compiling and caching on the application servers for increased scalability. All of our tests used a separate machine as the database server. Our evaluation used a TPC-W scale factor of 100K entries and the ⟨shopping⟩ workload. We used multiple client machines emulating concurrent users to the site, as outlined in TPC-W. Our comparisons were based on the peak performance of the different server alternatives defined as the maximum number of WIPS (Web interactions per second) while meeting the 90th-percentile total response time constraints specified for all of the client–site interactions. Since our evaluation focused on the first two tiers of the Web site, we eliminated the database tier from affecting our comparisons by always maintaining a fixed load of 2000 concurrent Web site users, resulting in a CPU utilization of ~37% on the database server. This was done by using additional client and application service machines to maintain the appropriate background load on the database tier.

Our second benchmark, ⟨web99⟩, was a modified version of SPECweb**99 [25]. We used the metrics, request types, and conformance rules of the benchmark. In this benchmark, one or more clients generate connections to the HTTP server that consist of a sequence of requests in the pattern "sleep—HTTP request—wait for response." Server throughput was measured as the number of simultaneous connections per second that satisfy the bandwidth requirement of 40 000 bytes per second. The sleep interval in the request pattern was determined dynamically to achieve a connection bandwidth between 40 000 and 50 000 bytes per second. The bandwidth requirement and the way in which it is implemented enforces a *de facto* responsiveness requirement of 0.03 seconds per operation. The primary difference between SPECweb99 and our ⟨web99⟩ benchmark is the distribution of request types issued by the client. In our benchmark, 50.4% of requests were dynamic GETs, 9.6% were dynamic POSTS, and 40.0% were static GETs. We used the TUX in-kernel Web server [21] from the standard Linux distribution, modified as described in the section on the distributed file cache, as the Web server when running the ⟨web99⟩ benchmark.

[5] For legal reasons, we cannot publish official benchmark results except for generally available products.

Evaluation methodology

System configurations

We used the benchmarks described in the preceding section to carry out a comparative study between the SDS and a traditional server configuration. To conduct a fair and conservative comparison, we fixed the deployment space and configured both servers to represent the respective resource densities possible within that space. For the traditional server configuration, we chose a server with a 1.2-GHz Pentium III Intel processor, 512-KB cache, 2 GB of RAM, and one 1-Gb/s Ethernet interface. This server has a 1-U form factor, and it is hereafter referred to as the 1-U server. The 1.2-GHz processor was the best available server-grade processor at the time of the SDS design. Although the 1-U server contains a local disk, we used it only for system files and swapping, following the industry trend to use remote disks accessed by NFS for all data storage. The requirements of some of the software that we used on the 1-U server (for example, VMware ESX 1.5 [32]), precluded fully diskless operation. Our SDS design allows a standard 42-U rack to contain up to 360 blades, so an SDS configuration that uses space equivalent to a 1-U server would contain between eight and nine SDS blades. Thus, we configured the SDS deployment to consist of eight SDS blades, each configured to run the processor at its low speed of 300 MHz, with 256 MB of RAM and one active 100-Mb/s Ethernet connection. In several of our experiments, we used clustering mechanisms to make our eight server blades appear as a single server to network clients. We refer to this configuration as a *blade cluster*.

Power measurements

We measured power by sampling the ac current and voltage levels using a sense resistor in-line with the power cord of the system being measured. Throughout this paper, the term *watts* refers to ac wattage input to the ac-to-dc converting power supplies. Voltage and current were

Table 1 Performance of ⟨tpc-w⟩.

Server	*WIPS* (W)	*Power*	*WIPS/MHz*	*WIPS/watt*	*CPU* (%)	*Network* (Mb/s)
SDS	117	104	0.049	1.13	76 (avg)	7.6 (avg)
Traditional server	68	102	0.059	0.67	96	28.4

sampled at 10 kHz with a National Instruments** SCXI-1102C filter module[6] and transmitted to a PC employing a National Instruments PCI6052E measurement card. Software running on the PC computes cumulative energy consumption from the measured voltage and current and broadcasts a packet over the local subnet containing the cumulative energy consumed and other data, including a time stamp, at a rate of once per second. Simultaneously, the monitoring application (see the section on blade monitoring) running on each blade sends out a packet containing performance and utilization data at a rate of once per second. The power and performance measurements were collected and stored by a separate system. These measurements were the basis for the results presented in this paper. The measured power data and performance data were independently validated by other instruments and benchmarks for accuracy and reliability.

In collecting our power data, we made some decisions about which components to include and exclude in the measurements of both our SDS prototype and the 1-U servers. In the SDS configuration, we excluded the power consumed by the MetaServer, the PARD server, the NFS servers holding the test data, the network switches, the chassis management blades, and the fans. We did this for two reasons. First, and most significantly, although the design point of our prototype allows us to support at least 36 blades with the parts and servers that we excluded, all of our measurements use an eight-blade configuration; thus, they cannot take into account how the power consumed by the excluded items is amortized across a much larger number of blades in a fully configured environment. Second, the prototype environment imposed some practical constraints on the collection of the power-consumption data. In the case of the MetaServer, the PARD server, and the NFS servers, the systems are housed in different chassis than the blades, and the fans in the SDS chassis are powered separately from the blades. To exclude the power consumed by the network switches and the chassis management blades, we measured a chassis with no server blades powered on and subtracted the result from our measurements with one or more blades active. As a result of these exclusions, we measured the power consumed by all of the blades that were powered on. In the case of the 1-U servers, we measured the total power consumed by the servers excluding that consumed by the disk drives; we excluded the disk power by connecting the drives to a separate power supply. We did this because none of the power consumed by the disks used to support our SDS was included in our power measurements of it.

[6] National Instruments Corporation, Austin, TX.

Efficiency metrics

We employed two efficiency metrics in our evaluation to quantify the performance of our server configurations with respect to their processing capacity and power usage. We measured the *performance efficiency* as delivered performance, expressed in terms of the benchmark rating, divided by raw processing power, in aggregate CPU MHz. For ⟨tpc-w⟩, performance efficiency is expressed as WIPS/MHz; for ⟨web99⟩, it is expressed as connections/MHz (conn/MHz). We also measured *power efficiency* as the delivered performance, in terms of the benchmark rating, divided by average power consumption in watts. Power efficiency is expressed in WIPS/watt for ⟨tpc-w⟩ and connections/watt (conn/watt) for ⟨web99⟩. For both efficiency metrics, a higher value indicates a more efficient configuration.

Results for e-commerce workloads

This section reports the results of our evaluation using e-commerce workloads.

Results for ⟨tpc-w⟩

Table 1 summarizes the performance of SDS and traditional server configurations on the application service workload ⟨tpc-w⟩. SDS obtains 1.72 times the performance of the traditional server at comparable energy costs. The CPU is the performance-limiting factor for both the SDS blades and 1-U server for ⟨tpc-w⟩. Thus, the key advantage of SDS for this application workload is the superiority of its CPU density (8 × 300 MHz) over that of the 1-U server (1.2 GHz). This result is somewhat surprising, since the conventional wisdom is that blades are good only for the edge-server type of workload (tier 1), and that application servers require far more resources than can be found on dense blades. Table 1 also presents the performance and power efficiency results for our two server configurations. The SDS configuration has a slightly lower performance efficiency, primarily because

Table 2 Details for ⟨tpc-w⟩ for both servers.

	Traditional server			*SDS*		
	CPU (%)	*Network* (Mb/s)	*Disk* (ops/s)	*CPU* (%)	*Network* (Mb/s)	*Disk* (ops/s)
NFS server	2	0.60		5	11.1	128
Database server	37	5.5	49.9	38	5.7	47.3

CPU utilization is lower for SDS than for the traditional server. Note that the SDS system is significantly more power-efficient, by almost a factor of 2. Additional experiments indicate that roughly 3 W per blade could be conserved during system idle time simply by halting the processor in the idle loop. We could not implement this power-conservation strategy because of a problem with our prototype hardware. Voltage and frequency scaling could have reduced our energy consumption further during idle periods. In short, we feel that these results provide a conservative estimate of the energy efficiencies possible under varying workloads.

Looking in greater detail into the performance outlined in **Table 2**, we see significantly greater NFS server activity for the SDS blades as opposed to the 1-U server. The image data set for our evaluation problem size, 100K, would fit into 4 GB of memory, the amount available on a fully configured 1-U server, which it could share with a second processor on a 1-U server. To avoid penalizing the 1-U server by the choice of our comparison point, we reduce the image data set size to that for 50K while keeping the problem size the same. After warmup, this allows the 1-U server to serve all of the images from its memory, resulting in NFS activity for only server logs. Each blade has only 256 MB of memory, which could cache only about a fourth of the uniformly accessed image data set, resulting in significantly higher NFS server activity. The increased NFS usage to serve images also explains the lower average CPU utilization figures for the blades when compared to the 1-U server—they are waiting for NFS. There are techniques for avoiding this, such as static partitioning of the data set with clever request routing at the switch or connection handoff. Thus, our numbers present a conservative picture of the power–performance advantage enjoyed by SDS over traditional servers. It is worth noting that despite the dependence of the blades on the network due to diskless operation and the use of only one network interface by our prototype hardware, the per-blade network utilization is quite low; blade network bandwidth is not the performance-limiting factor for ⟨tpc-w⟩.

The 1-U server poses no load on the NFS server, since the 2 GB of RAM is sufficient to cache all files. Only log writes contribute to the traffic. The log writes of the blades force 128 disk I/O operations per second, or one I/O operation every 8 ms on average. Thus, we can conclude that NFS is not a bottleneck.

Time-varying ⟨tpc-w⟩ results

In reality, servers seldom operate at full load for long periods of time. Several studies have found that Web workloads are bursty in nature, and support the intuitive notion that Web servers tend to be busiest during some peak hours during the day and almost idle in others [33]. Our SDS design enables fine-grained power management of a blade cluster to achieve energy savings during periods of low to moderate workload. Our blade cluster uses the power-aware request distribution (PARD) described in Section 3 to lower the energy consumed by the blade cluster. The ability to turn blades off and bring them back up quickly allows our SDS design to achieve much higher power efficiency than traditional servers.

To evaluate the benefits of SDS systems equipped with PARD, we developed a time-varying workload based on ⟨tpc-w⟩. **Figure 4(a)** illustrates the variation in request rate over the course of this workload. This request pattern is based on the diurnal logs of a major financial Web site. The 24-hour request log was reduced to two hours by averaging over consecutive 12-minute intervals in order to produce short, repeatable runs. Further, its peak load was scaled to the peak sustainable by the SDS system to avoid overstating the energy savings by having excess capacity that is always off. To determine the effects of closing down connections in this accelerated time frame, the ⟨tpc-w⟩ average user session length was also reduced, from 15 minutes to two minutes. This reduction is still significantly less than the 12× reduction in wall-clock time, giving conservative estimates on the power savings.

Figure 4(b) shows the power consumed by the blade cluster over the course of the workload both with and without the PARD mechanism. We did not include the traditional server in this analysis because there is no straightforward analog of PARD for a monolithic server that operates as a single host. Note that power consumed by the blade cluster without PARD is relatively constant over the course of the run. In contrast, power for the blade cluster with PARD is significantly lower during periods of light to moderate user activity. The total energy

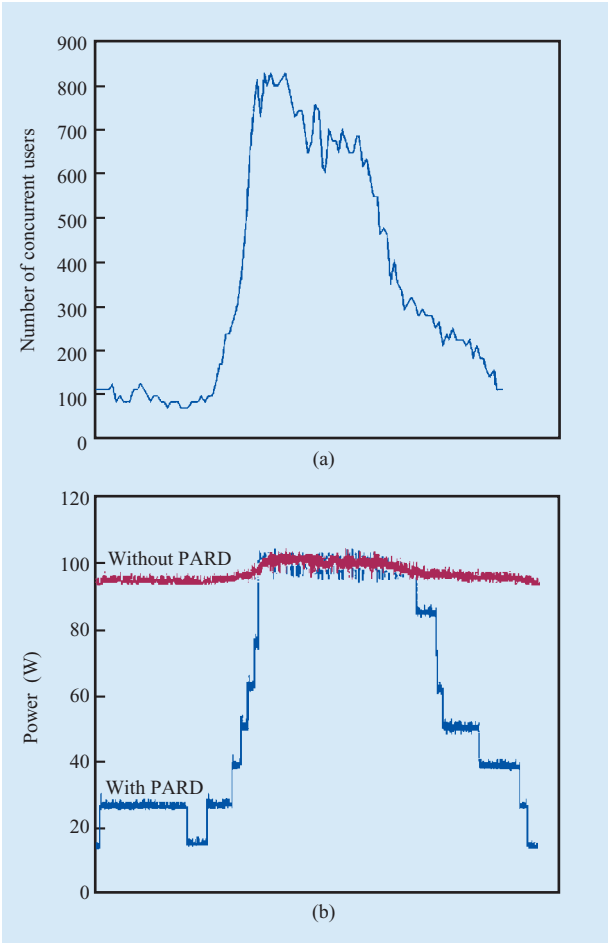

Figure 4

Time-varying ⟨tpc-w⟩: (a) Request rate profiles; (b) power consumption.

consumed during the run is the area under each curve, which is 698 joules without PARD and 415 joules with PARD. Thus, our PARD mechanism generates 40% savings in energy for the time-varying ⟨tpc-w⟩ workload. Additional tuning with more aggressive LVS timeouts to speed up the detection of inactive connections and tuning the *connection_capacity* value to match expected workloads would further improve the energy savings. This experiment provides evidence of the viability of even simple schemes to provide significant energy savings using SDS servers for time-varying workloads.

Results for Web-serving workloads

We used ⟨web99⟩ to study the performance of our blades on Web workloads containing a mix of static and dynamic content. We performed two sets of experiments. The first set of experiments simulated a data center environment that hosts a large number of small, independent Web sites. In this environment, the workload for each Web site is typically small, and even though engineers configure such sites for peak activity, the peak workload is typically insufficient to drive a single traditional server near its capacity. In this experiment, each SDS blade handled one independent Web site. The traditional server, on the other hand, has to "scale down" by using partitioning or virtualization. We use VMware ESX 1.5 to partition the traditional server into up to eight logical partitions. Each of these handles an independent workload, at the expense of the virtualization overhead.

Table 3 presents the performance in number of connections per second (conn/s) and power consumption in watts for a single blade and single logical partition. Power consumption for the logical partition was computed by dividing the aggregate measurement by 8. Table 3 also shows the performance and power efficiency for both configurations. To compute the performance efficiency of a logical partition, we used 1200/8 = 150 MHz as the effective MHz. These results demonstrate that blades are a better solution than logical partitioning on a per-unit basis because each blade can sustain a larger number of connections per second than a partition. A blade is also more power-efficient by about a factor of 2. The superiority of the blades in this environment is due to the ability to pack twice as much processor capacity into the same space as the 1-U server. Under this load, the blades need not communicate with one another to serve the requests. We also note that during this experiment, no significant paging activities were detected on either a blade or a partition, and that the NFS server workload was similar between the two systems.

Note that a logically partitioned server, however, provides a "safety valve" in situations in which the actual workload exceeds the configured peak capacity for one partition. If one partition undergoes a load spike that exceeds its peak capacity while the other partitions are underloaded, the overloaded partition can steal CPU cycles from the other partitions to handle the spike. In a separate measurement, we determined that one logical partition can handle a load spike of up to 420 connections per second when the remaining seven partitions are idle.

The second set of experiments model the case in which a single Web site has enough traffic to warrant dedicating an entire 1-U server to the site. Here we compare a single dedicated 1-U server and a blade cluster collectively servicing the site. It does not make sense to group virtualized servers together here. In the blade case, we examined two different configurations. The first was typical of most off-the-shelf software solutions found in practice, where the blades access the data set of the Web site through NFS. The second configuration employed the distributed file cache implementation described in

Table 3 Multiple-site ⟨web99⟩ performance.

Server	*Conn/s*	*Power* (W)	*Conn/MHz*	*Conn/watt*	*CPU* (%)
SDS blade	125	12.3	0.42	10.2	22.9
Logical partition	65	12.7	0.43	5.1	18.8

Table 4 Results for single-site ⟨web99⟩ workload for two blade-cluster configurations and a traditional server.

Server		*Power*	*Conn/MHz*	*Conn/watt*	*CPU* (%)	*Network* (Mb/s)	*NFS disk* (ops/s)
Blade cluster (NFS)	385	95.3	0.16	4.6	9 (avg)	21 (avg)	130
Blade cluster (Custom)	850	98.6	0.35	8.6	25 (avg)	49 (avg)	188
Traditional server	1250	95.9	1.04	13.0	91	478	89

Section 3. **Table 4** shows the results for our single Web site ⟨web99⟩ workload for a blade cluster using NFS, a blade cluster using our distributed file cache, labeled Custom, and a traditional 1-U server.

In contrast to smaller independent Web sites, we find that large Web sites are best hosted using 1-U servers. The SDS blade cluster using NFS delivers less than one third the performance of the 1-U server for this workload. The distributed file cache support improves the performance of the blade cluster by more than 100%, but still achieves only 68% of the performance of a 1-U server. The power efficiency of the SDS blade cluster is about 34% less than that of the 1-U server. When serving 1250 connections, the CPU is the bottleneck for the 1-U server.

The low rate of NFS disk requests for the traditional server relative to the connections served indicates that it retains most of the data set in its file cache. In contrast, disk access is the constraining factor for both the NFS and custom blade cluster configurations. Because the custom blade cluster aggressively evicts large files from its page cache, most of the disk requests in this configuration are for large files, which enables the disk to attain a higher number of I/O operations served per second. In the NFS blade cluster configuration, the file system cache on the blade servers routinely becomes polluted, so that both large and small files must be fetched from the remote disk. Network bandwidth does not limit performance for any of our configurations.

5. Qualitative assessment and discussion

The previous section characterized quantitatively the benefits of energy-efficient, dense servers in terms of performance and energy consumption. In this section we focus on how systems software contributed to improvements in other key areas such as reliability, manageability, price/performance, and deployment choices. We first discuss how our software is an enabler for energy-efficient servers and then speculate on future opportunities for system-software research on dense servers.

We believe that the operating system and middleware enhancements we developed were instrumental in building a working SDS prototype that was physically dense and easy to administer and had superior performance and energy characteristics. One of our fundamental design decisions was to configure the blades without local disks. We made this decision because a local disk would increase power consumption, decrease server density because of increased volume and cooling requirements, and negatively affect blade reliability by introducing a mechanical component into an otherwise completely solid-state design. Another important design decision was to provide each blade with only network connections and no other I/O devices. Since the network connection is carried on the backplane of the CompactPCI chassis, this effectively eliminates any direct cable connections to the server blades. Eliminating cable connections decreases cost and improves system reliability. Cabling is a nontrivial problem in environments with large numbers of densely packed systems: It is hard to find places to put all of them, easy to connect them incorrectly, and even easier to disconnect or loosen them accidentally. Fixing a cabling problem can be a slow and labor-intensive procedure.

Linux-DSA provides the operating system support required by this design at a minimum overall system cost by allowing all of the server blades in an enclosure to share a single read-only root file system stored on a single disk attached to the management blade. At the same time, this approach improves system manageability by centralizing much of the system maintenance and configuration activity on the MetaServer. Our console over Ethernet support allows the consoles for all of the blades

to appear at a single, conveniently located workstation, reducing the need to go to the blade enclosure itself to perform low-level administrative tasks. Moreover, our use of console over Ethernet eliminates the need for graphics hardware and KVM connections on the blades, allowing a more space-efficient design.

Even though the SDS blades are prototypes and not production hardware, they are constructed from high-volume components and therefore could have a price/performance ratio competitive with those of traditional 1-U servers. Moreover, server blades should offer lower cost of operation than traditional servers because of lower power and cooling expenses. This suggests that new metrics are needed to properly evaluate new server architectures that offer different tradeoffs between purchase price and operating costs. One possible metric is the ratio of the sum of purchase price and projected operating costs to the performance of the machine.

6. Future work

There are a number of opportunities for further research in the area of energy-efficient servers. We believe that the ability to partition a server at relatively fine granularity into multiple components reduces the cost of any single failure, since only a small fraction of the entire cluster becomes unavailable. Our very short boot time makes a fail-and-recover strategy attractive at the blade level.

For reliability, since the CompactPCI architecture requires a management blade, two system management blades are configured in each CompactPCI chassis such that they act as a backup for each other with capability of hot swap and instantaneous fail-over. The hardware-level high-availability mechanisms and algorithms are properties of the Ziatech hardware [15] we are using for the system management blades. However, to avoid making the MetaServer a single point of failure, we have to be able to use the second management blade as a MetaServer backup. To do this, we must be able to

- Detect MetaServer failure.
- Bring up DHCP on the backup MetaServer.
- Switch the blades' mounts of their root and `/var` file systems from the original MetaServer to the backup.
- Change swap servers if necessary.

Of these items, the most difficult is switching the mounts. Some of aspects of this process may be easier if EBD rather than NFS is used to access the root and `/var` file systems, since the switch is done at the device rather than the file system level. Given our very short shutdown and reboot times, restarting the blades may remain a better alternative in some environments. In addition, the PARD server is also a single point of failure. Fail-over techniques similar to those used for the MetaServer should work reasonably well, but the feasibility of preserving the state of any open connections and the best techniques for doing so are topics for further investigation.

Conventional wisdom holds that systems with many components are harder to manage and increase the total cost of ownership. A major challenge for the SDS design is to reduce the cost and complexity of managing a large number of servers. Our Linux-DSA environment is a significant achievement in this area, but further improvements are possible by centralizing additional management tasks, providing more granular performance management, and enabling graceful degradation of service in the event of failures.

One possibility for future research would be to apply the basic ideas of SDS to other areas such as RAID storage devices or clustered database servers. We have applied these ideas primarily to edge and middle-tier servers, but we believe massive parallelism can provide similar benefits of high performance and reduced total energy consumption for other cluster-style architectures.

It would be interesting to explore the benefits of allowing one blade to borrow the under-utilized resources of another. For example, instead of completely powering down a blade whose processing power is not currently needed, a means might be provided for another blade to use its memory as a disk cache. This could enable more flexible configuration of SDS blades for hosting a set of independent Web sites.

Finally, our evaluation has shown that traditional servers provide better performance than a blade cluster for workloads in which processing or data are not easily partitioned, as in the case of our single-site ⟨`web99`⟩ workload. Future research should focus on mechanisms for efficient distribution of work to the blade with the correct resources or transparent resource sharing, with the goal of identifying ways to make more effective use of the aggregate resources of our blade cluster. In particular, it would be interesting to explore an extension of our PARD mechanism that incorporates locality-aware request distribution [34].

7. Related work

Chase et al. [6] have studied aspects of cluster-based power management in server systems. They have used a scheme inspired by economic theory to decide which servers could be turned off or on depending on workload variations and desired quality of service. Their report describes a prototype implementation using traditional servers and shows up to 29% savings in measured energy consumption without compromising quality of service or performance. Pinheiro et al. [35] have described a similar scheme. Rajamani et al. [18] have examined the issues

involved in the design and evaluation of power-aware request distribution (PARD) schemes for server clusters. They have identified the key system-workload factors for PARD schemes and have quantified their impact on energy-saving strategies. Elnozahy et al. [8] have presented a theoretical analysis of other power management schemes for server clusters that support voltage scaling of the individual servers. Their analysis showed that energy savings of up to 60% could be possible using coordinated voltage scaling in conjunction with turning machines on and off depending on workloads. In a different paper, Elnozahy et al. [36] have advocated request batching as a method to achieve energy savings during low-intensity workloads that would otherwise prevent machines from being turned off. The work presented here differs from these efforts in that it focuses on densely packed systems built out of low-power components and studies the performance and deployment of such systems as an alternative to traditional servers.

Dense servers made their product debut in 2001, subsequent to the initial design of our prototype. RLX Technologies**[7] used the Transmeta** Crusoe** chip[8] [37] to implement its low-power ServerBlade** [4], but later switched to low-power Intel processors [14]. More recently, Amphus**,[9] Compaq** (prior to its merger with Hewlett-Packard**), Hewlett-Packard, and others have offered dense servers that claim excellent energy efficiency [1–3]. We are not aware of any performance evaluation of these servers or how they compare to traditional alternatives. Our blade prototype is similar in spirit to these offerings in that it uses low-power components and can potentially pack up to 320 systems in a standard server 19-inch rack.

After the work described here was completed, several vendors (including IBM) introduced a new class of "function-dense" blades which have the same specifications as 1-U servers but with slightly higher density. To contrast with this style of server architecture, the blades described in this paper are now commonly referred to as "node-dense" blades. Clearly, function-dense blades are suitable for all workloads that can run on two-way or smaller servers. We expect that node-dense blades will retain their density and power advantage for suitable workloads. In a market in which all two-way and smaller servers are blades, the question changes from whether blades or conventional servers should be used to what kind of blades should be used.

Power management has been studied extensively for pervasive computing devices and portable computers. Lorch and Smith have provided a survey of software techniques for power management [38], and Ellis has made the case for involving the software in power management [39]. Vahdat et al. have described operating system techniques for managing the power of dynamic memory [40], while Flinn and Satyanarayanan have described several techniques for adapting the system to application profiles in mobile systems [41]. There are also several standards that specify the interfaces between power management software and the hardware [42, 43]. The difference between all of these efforts and the work presented here is that they have remained focused on portable applications, whereas we have focused on server workloads. Owing to the relatively short time that dense servers have been in existence, it is not yet clear whether the differences in application structure and resource demand will imply the need for additional power management features and methods, although our initial work has led us to believe that this will be the case.

8. Conclusions

We have demonstrated that dense servers offer better performance and lower energy consumption on CPU-intensive as well as multiple small-scale, independent workloads than do alternative deployments using traditional servers. For TPC-W-style workloads, our prototype SDS implementation, configured as a blade cluster, outperforms a traditional server by almost a factor of 2 at comparable energy costs. Similarly, for small-scale, independent workloads, a group of dense servers running as individual systems offers better performance and lower energy consumption than a virtualized server handling the same workload. On the other hand, the relatively small amount of memory on each prototype SDS blade makes it difficult to scale up the performance of a cluster of dense servers on memory-intensive workloads without either specialized software or static partitioning of the data. Even then, a traditional server offers better performance. We conclude that the choice between dense and traditional servers currently depends on the intended usage, and in many cases, installations may be best served by a mixture of the two. However, the technology trends are that components designed for low-power processing continue to increase in the amount of computing resource they offer. Since our work was conducted, both blades and traditional servers have increased in performance in roughly equal proportion; as of the completion of this paper, the densest blades used 900-MHz Pentium III CPUs and up to 1 GB of RAM, while 1-U servers had two 2.8-GHz Xeon CPUs and up to 8 GB of RAM. Because the relative performance of blades and traditional servers has been essentially unchanged, we expect that our conclusions will continue to hold. If these trends continue, we predict that the advantages of dense servers will grow while their limitations gradually disappear.

[7] RLX Technologies Corporation, The Woodlands, TX.
[8] Transmeta Corporation, Santa Clara, CA.
[9] Amphus Corporation, San Jose, CA.

Acknowledgments

We would like to thank Jessie Gonzalez, David Pruett, Chuck Lanier, and Alan Van Antwerp for their help during server bringup. John Carter had several useful suggestions concerning the methodology of evaluation. Chandler McDowell helped in setting up the power measurement. Alison Smith and Ravi Kokku helped in the implementation of the PARD infrastructure. Patrick Bohrer, Bishop Brock, David Cohn, and Hazim Shafi had several useful comments on the paper and the early stages of this work. This research was supported in part by the Defense Advance Research Projects Agency under Contract No. F33615-00-C-1736. We acknowledge the trademarks and copyrighted material mentioned here as the property of their owners.

*Trademark or registered trademark of International Business Machines Corporation.

**Trademark or registered trademark of Oracle Corporation, Microsoft Corporation, Intel Corporation, PCI Industrial Computer Manufacturers Group, Linus Torvalds, The Open Group, Red Hat, Inc., Ziatech Corporation, Transaction Processing Performance Council, Standard Performance Evaluation Corporation, National Instruments, Inc., RLX Technologies, Inc., Transmeta Corporation, Amphus Corporation, Compaq, Inc., or Hewlett-Packard Company.

References

1. Amphus Corporation, San Jose, CA, "Virgo: A ManageSite-Enabled, Fully Manufacturable, Ultra-Dense Server Design," 2001.
2. Hewlett-Packard Company, Palo Alto, CA, "Proliant BL10e Server" (formerly a product of Compac, Inc.), January 2002.
3. Hewlett-Packard Company, Palo Alto, CA, "HP bc1100," December 2001.
4. RLX Technologies, The Woodlands, CA, "Redefining Server Economics," May 2001.
5. See *http:www-1.ibm.com/servers/eserver/bladecenter/*.
6. J. Chase, D. Anderson, P. Thakar, A. Vahdat, and R. Doyle, "Managing Energy and Server Resources in Hosting Centers," *Proceedings of the 18th Symposium on Operating Systems Principles* (*SOSP*), October 2001, pp. 103–116.
7. P. Bohrer, E. Elnozahy, T. Keller, M. Kistler, C. Lefurgy, C. McDowell, and R. Rajamony, "The Case for Power Management in Web Servers," *Power-Aware Computing*, R. Graybill and R. Melhem, Eds. Kluwer Academic Publishers, New York, January 2002.
8. E. Elnozahy, M. Kistler, and R. Rajamony, "Energy-Efficient Server Clusters," presented at the Workshop on Power-Aware Computing Systems, Cambridge, MA, February 2002; workshop proceedings to be published as an issue of *Lecture Notes in Computer Science* (*LNCS*), Springer-Verlag, Heidelberg, Germany.
9. See *http://www-3.ibm.com/software/info1/websphere/*.
10. Oracle Corporation, Redwood Shores, CA, "Application Server"; see *http://www.oracle.com/ip/deploy/ias/*.
11. Microsoft Corporation, Redmond, WA, "Microsoft .Net"; see *http://www.microsoft.com/*.
12. J. Gray and A. Reuter, *Transaction Processing: Concepts and Techniques*, Morgan Kaufmann Publishers, Inc., San Francisco, CA, 1993.
13. PCI Industrial Computer Manufacturing Group, "CompactPCI Specifications, PICMG 2.16," October 2001; see *http://www.picmg.org/*.
14. Intel Corporation, Santa Clara, CA, "SpeedStep"; see *http://developer.intel.com/mobile/Pentium III/*.
15. Ziatech Corporation, Rochester, NY, "Redundant CPU Architecture for High Availability Systems," 2001.
16. E. V. Hensbergen and F. Rawson, "Revisiting Link-Layer Storage Networking," *Technical Report 22602*, IBM Austin Research Laboratory, Austin, TX 78758, 2002.
17. R. Pike, D. Presotto, K. Thompson, H. Trickey, and P. Winterbottom, "The Use of Name Spaces in Plan 9," *Oper. Syst. Rev.* (reprinted from *Proceedings of the 5th ACM SIGOPS European Workshop*) **27**, No. 2, 72–76 (1992).
18. K. Rajamani and C. Lefurgy, "On Evaluating Request-Distribution Schemes for Saving Energy in Server Clusters," *Proceedings of the IEEE International Symposium on Performance Analysis of Systems and Software*, March 2003, pp. 111–122.
19. W. Zhang, "Linux Virtual Server for Scalable Network Services," presented at the Ottawa Linux Symposium, 2000.
20. R. Minnich, J. Hendricks, and D. Webster, "The Linux BIOS," *Proceedings of the Fourth Annual Linux Showcase and Conference*, October 2000, pp. 73–79.
21. Red Hat Corporation, Raleigh, NC, "Netconsole," 2002; see *http://www.redhat.com/software/rhel/notes/ws/*.
22. M. Kistler, E. Van Hensbergen, and F. Rawson, "Console over Ethernet," *Proceedings of the FREENIX Track: 2003 USENIX Technical Conference*, June 2003, pp. 125–136.
23. Linux Network Block Device, "NBD," 2002; see *http://nbd.sourceforge.net/*.
24. W. Smith, "TPC-W: Benchmarking, an Ecommerce Solution," The Transaction Processing Performance Council, February 2000; see *http://www.tpc.org/tpcw/*.
25. Standard Performance Evaluation Corporation, "An Explanation of the SPECweb99 Benchmark," 1999; see *http://www.spec.org/*.
26. P. DuBois, *MySQL*, New Riders Publishers, Indianapolis, IN, December 1999.
27. Apache Software Foundation, "The Apache HTTP server"; see *http://httpd.apache.org/*.
28. Apache Software Foundation, "PHP," 2002; see *http://www.php.net/*.
29. Red Hat Corporation, Raleigh, NC, "TUX 2.1," 2001; see *http://www.redhat.com/docs/manuals/tux/TUX-2.1-Manual/*.
30. K. Rajamani, "Multi-Tier Caching of Dynamic Content for Database-Driven Web Sites," Ph.D. thesis, Department of Electrical and Computer Engineering, Rice University, Houston, TX, October 2001.
31. APC Community Connect, "APC: Alternate PHP cache"; see *http://apc.communityconnect.com/*.
32. VMware, Inc., Palo Alto, CA, "VMware ESX Server 1.5," 2002.
33. M. Crovella and A. Bestavros, "Self-Similarity in World Wide Web Traffic: Evidence and Possible Causes," *Proceedings of the 1996 SIGMETRICS Conference on Measurement and Modeling of Computer Systems*, May 1996, pp. 160–169.
34. V. S. Pai, M. Aron, G. Banga, M. Svendsen, P. Druschel, W. Zwaenepoel, and E. M. Nahum, "Locality-Aware Request Distribution in Cluster-Based Network Servers," *Architectural Support for Programming Languages and Operating Systems* (*ASPLOS*), October 1998, pp. 205–216.
35. E. Pinheiro, R. Bianchini, E. V. Carrera, and T. Heath, "Load Balancing and Unbalancing for Power and Performance in Cluster-Based Systems," presented at the Workshop on Compilers and Operating Systems for Low Power, September 2001.
36. E. Elnozahy, M. Kistler, and R. Rajamony, "Energy Conservation Policies for Web Servers," *Proceedings of the*

4th USENIX Symposium on Internet Technologies and Systems (*USITS'03*), March 2002, pp. 99–112.
37. M. Fleischmann, "Crusoe Power Management: Cutting x86 Operating Power Through LongRun," presented at the Embedded Processor Forum, June 2000.
38. J. Lorch and A. Smith, "Software Strategies for Portable Computer Energy Management," *IEEE Personal Commun. Magazine*, pp. 60–73 (June 1998).
39. C. Ellis, "The Case for Higher-Level Power Management," *Proceedings of the 7th IEEE Workshop on Hot Topics in Operating Systems (HotOS-VIII)*, 1999, pp. 162–167.
40. A. Vahdat, A. Lebeck, and C. Ellis, "Every Joule Is Precious: The Case for Revisiting Operating System Design for Energy Efficiency," *Proceedings of the 9th ACM SIGOPS European Workshop*, September 2000.
41. J. Flinn and M. Satyanarayanan, "Energy-Aware Adaptation for Mobile Applications," *Proceedings of the 17th ACM Symposium on Operating Systems Principles* (*SOSP'99*), 1999, pp. 48–63.
42. Intel, Microsoft, and Toshiba, "Advanced Configuration and Power Management Interface ACPI Specification," 1999; see *http://www.intel.com/ial/WfM/design/pmdt/acpidesc.htm*.
43. *PC99 System Design Guide*, Microsoft Press, Microsoft Corporation, Redmond, WA, 1999.

Received November 10, 2002; accepted for publication March 28, 2003

Wesley M. Felter *IBM Research Division, Austin Research Laboratory, 11501 Burnet Road, Austin, Texas 78758 (wmf@us.ibm.com).* Mr. Felter joined IBM in 2001 after receiving a B.S. degree in computer science from the University of Texas at Austin. He is currently working on low-power software and systems.

Tom W. Keller *IBM Research Division, Austin Research Laboratory, 11501 Burnet Road, Austin, Texas 78758 (tkeller@us.ibm.com).* Dr. Keller received a B.S. degree in physics with honors in 1971 and a Ph.D. degree in computer science in 1976, both from the University of Texas at Austin. He currently manages the Power-Aware Systems Department at the Austin Research Laboratory. Before joining IBM in 1989, he helped inaugurate the first Cray-1 computer at the Los Alamos Scientific Laboratory and prototyped a parallel database computer at MCC, where his performance group created the well-known TPC-C benchmark. Dr. Keller has served as Associate Director of the University of Texas Computation Center and Chair of ACM Sigmetrics. He is the author of 39 refereed publications and several patents.

Michael D. Kistler *IBM Research Division, Austin Research Laboratory, 11501 Burnet Road, Austin, Texas 78758 (mkistler@us.ibm.com).* Mr. Kistler received a B.A. degree in computer science from Susquehanna University in 1982 and an M.S. degree in computer science from Syracuse University in 1990. He joined IBM in 1982 and has held technical and management positions in MVS, APPC, and OS/2 development. He also worked (for two years) at Lotus/Iris on clustering technology for the Lotus Domino product. He joined the Austin Research Laboratory in 2000 and is currently working on projects focusing on the management and performance of clusters of dense servers. He is also currently a Ph.D. student in the Department of Computer Science at the University of Texas at Austin. Mr. Kistler's research interests are parallel and cluster computing, particularly for large commercial applications such as Web application servers.

Charles Lefurgy *IBM Research Division, Austin Research Laboratory, 11501 Burnet Road, Austin, Texas 78758 (lefurgy@us.ibm.com).* Dr. Lefurgy received B.S.E. (1994), M.S.E. (1996), and Ph.D. (2000) degrees in computer engineering from the University of Michigan. His dissertation work focused on compressed memory systems for embedded computers. He then joined the Austin Research Laboratory to work on software optimizations for energy-efficient server design. Dr. Lefurgy's other research interests include microarchitecture, memory management, and simulation of computers.

Karthick Rajamani *IBM Research Division, Austin Research Laboratory, 11501 Burnet Road, Austin, Texas 78758 (karthick@us.ibm.com).* Dr. Rajamani is a Research Staff Member in the Power-Aware Systems Department at the IBM Austin Research Laboratory. He is a graduate of the Indian Institute of Technology, Madras, and holds master's and doctoral degrees in electrical and computer engineering from Rice University, Houston, Texas.

Ram Rajamony *IBM Research Division, Austin Research Laboratory, 11501 Burnet Road, Austin, Texas 78758 (rajamony@us.ibm.com).* Dr. Rajamony received a Ph.D. degree in electrical and computer engineering from Rice University in 1998 and a B.Tech. degree in electrical engineering from the Indian Institute of Technology, Madras, in 1989. At the IBM Research Division, which he joined in 1998, his research interests have spanned the areas of server power management, computer architecture, and operating systems. He is a co-inventor of six patents and has published more than 15 papers at venues such as USENIX, ISCA, and SIGMETRICS. In 2002, Dr. Rajamony received an IBM Outstanding Innovation Award, and in 1997, the ACM SIGMETRICS Best Student Paper Award.

Freeman L. Rawson *IBM Research Division, Austin Research Laboratory, 11501 Burnet Road, Austin, Texas 78758 (frawson@us.ibm.com).* Mr. Rawson joined IBM in 1973 in San Jose, California, after graduating from Michigan State University and Stanford University. In 1976 he transferred to the IBM facility in Boca Raton, Florida, where he worked on systems software development for the IBM Series/1 and its successors and on the development of OS/2. In 1990, he joined the group responsible for the development of future kernel technologies for IBM low-end and mid-range systems products and became the Technical Team Leader of the Core Kernel Technology group. In 1995, he moved to the IBM Austin facility with that group, and in 1996, to the Austin Research Laboratory. Mr. Rawson is currently working on the analysis, prediction, and reduction of power consumption in server systems.

Bruce A. Smith *IBM Systems Group, Austin Research Laboratory, 11501 Burnet Road, Austin, Texas 78758 (bruces@us.ibm.com).* Mr. Smith received a B.S.E.E. degree from the University of Pittsburgh in 1976. He joined IBM in 1977 and has worked mainly on Intel Architecture systems, beginning with the PC/AT. He was the system lead on seven high-volume microchannel desktop systems and is currently involved in x-series development, working on advanced telco servers. He is the inventor or co-inventor of nine patents, two of which were recognized in 2000 via an IBM Distinguished Contribution Award.

Eric Van Hensbergen *IBM Research Division, Austin Research Laboratory, 11501 Burnet Road, Austin, Texas 78758 (bergevan@us.ibm.com).* After receiving a B.S. degree in computer science from the Rochester Institute of Technology in 1997, Mr. Van Hensbergen joined Bell Laboratories to work on the Inferno operating system kernel. Two years later he continued his work at the Lucent Corporation, providing kernel and operating system support work for the PathStar access server converged network platform. He joined the Austin Research Laboratory in 2001 to work on the Super Dense Server Project and contribute to efforts on other low-power systems.