# KnightShift: Shifting the I/O Burden in Datacenters to Management Processor for Energy Efficiency

Sabyasachi Ghosh, Mark Redekopp, Murali Annavaram

# KnightShift: Shifting the I/O Burden in Datacenters to Management Processor for Energy Efficiency

Sabyasachi Ghosh, Mark Redekopp, Murali Annavaram
Electrical Engineering Department, University of Southern California

## Abstract

Data center energy costs are growing concern. Many datacenters use direct-attached-storage architecture where data is distributed across disks attached to several servers. In this organization even if a server is not utilized it can not be turned off since each server carries a fraction of the permanent state needed to complete a request. Operating servers at low utilization is very inefficient due to the lack of energy proportionality. In this research we propose to use out-of-band management processor, typically used for remotely managing a server, to satisfy I/O requests from a remote server. By handling requests with limited processing needs, the management processor takes the load off the primary server thereby allowing the primary server to sleep when not actively being used; we call this approach KnightShift. We describe how existing management processors can be modified to handle KnightShift responsibility. We use several production datacenter traces to evaluate the energy impact of KnightShift and show that energy consumption can be reduced by 2.6X by allowing management processors to handle only those requests that demand less than 5

## 1 Introduction

Datacenter power consumption is growing at a rapid pace to meet the demands of the service oriented architecture, where every computing need is delivered as a service over the cloud. Electricity costs for powering servers are a significant part of the total operating cost of a datacenter. It is well known that servers in a datacenter are rarely operated at 100% utilization. Due to lack of energy proportionality [4] servers consume significant energy even when they operate at low utilization. Server virtualization coupled with workload migration can reduce the number of active servers by consolidating many low utilization servers to a few servers operating at high utilization. Consolidation reduces the energy costs since servers that are idle, with no assigned workloads, can be turned off or placed in low power modes such as the PowerNap mode [11].

Powering down servers is not always possible even when they are idle, especially because of the direct-attached-storage (DAS) architecture used in many datacenters where data is distributed across disks attached to several servers. In almost all data-intensive applications, such as web search, e-commerce, and social networking, request processing requires accessing and modifying data stored on a disk. For instance, in an e-commerce website which is supported by a datacenter at the back-end, a user may browse the catalog of available items. In a direct-attached-storage architecture it is likely that some of the catalog data may be available on a server other the one assigned to process the request, in which case several I/O requests must be made to that server to retrieve the data.

Direct-attached-storage architecture scales well with the growth in the data size as cheap commodity disks can be used to scale storage without the need for expensive I/O solutions. The disadvantage of a direct-attached-storage architecture is that even if a server is at near 0% utilization it can not be turned off since each server carries a fraction of the permanent state needed to complete a request. A request that is being serviced by one server may need data that is distributed across multiple servers. Due to long server wakeup latencies datacenter operators may be forced to leave the machines ON even when there is no workload assigned to the server. It is in this context we propose KnightShift, a mechanism to shift the burden of servicing remote I/O requests from the primary server system to a specialized low power system that is closely coupled with the primary system. We call the low power system as the Knight. For the Knight to handle the remote requests it must satisfy the following requirements:

- Knight must be closely attached to the primary server so that it can concurrently access the disk storage of the primary server with low latency.
- In spite of the close proximity, Knight must be electrically isolated from the primary server and hence should be independently powered ON irrespective of the primary server's state.
- Knight must have enough functional capabilities to receive, interpret, service, and finally send a response to the I/O request.
- The requesting server must see an identical response when it is serviced by Knight compared to what would have been generated by the primary server.

If the above requirements are satisfied it is possible to power down the primary server completely during the times when the server is only processing remote server I/O requests. By letting Knight handle I/O requests it is possible to make the data attached to the primary server always available to a remote node. We can extend the capabilities of Knight further by allowing it to even handle certain non critical requests that are assigned to the primary server. By shifting the responsibility of satisfying simple requests to Knight the primary system can be powered down for even longer periods of time.

Recently, manufacturers have come up with remote man-

agement interfaces to improve server manageability, like Intel's Advance Management Technology (AMT) [10], IBM's Integrated Management Module [8], HP's integrated Lights Out (iLO) [7]. These are based on the Intelligent Platform Management Interface (IMPI) standard [6] that was defined by a consortium of industrial players. System administrators can remotely connect to these interfaces to monitor server health and perform management actions like turning the server ON or OFF, software installation etc. These systems are capable of using the network interface even when the primary server is off and may also use the same IP address. n this research, we explore how these remote management engines can be used as a Knight to satisfy I/O requests from a remote server even when the primary server is powered down.

The rest of the paper is organized as follows. Section 2 provides an overview of IPMI describing details that are relevant to understanding KnightShift. We then discuss design modifications to IPMI that may be needed to support remote requests in Section 3. Section 4 describes our experimental setup that is based on large scale datacenter traces collected from USC production facility over a period of 9 days. We present some preliminary results showing the potential energy improvement of KnightShift. Section 5 describes previous studies and we conclude in Section 6.

## 2 Overview of Intelligent Platform Management Interface

Figure 1 shows a high level overview of IPMI. At the core of the IPMI hardware there is a baseboard management controller (BMC) which is a low power microprocessor or even a simple microcontroller is integrated into the server baseboard. It provides all the intelligence necessary for remote server management. BMC is connected to the network interface of the primary server using a side-band interface and can send/receive data even if the primary server is powered off. It can even power ON/OFF the primary server. This ability to remotely access the primary server through BMC, even when the primary server is not accessible due to a system crash, is called out-of-band management.

BMC is typically used to collect and log data from various sensors to monitor server health. These logs may be accessed remotely by system administrators to do root cause analysis in case of a failure. Sensors can also get BMC's attention using events. For example, a temperature sensor may generate an event when the temperature reaches some critical value. The network card may generate an event when a packet is received. Several recent IPMI implementations can perform complex functions like acting as a network filter to detect viruses and denial-of-service attacks and blocking the network packets from entering or leaving the primary server if needed.

Communication between the system administrator and IPMI, more specifically BMC, is done through IPMI commands that are sent as messages over standard internet protocol (IP). The primary server and BMC not only share the physical network interface but may also share the same IP address. It is this feature of a shared network interface with the same IP address that provides interesting avenues to service I/O requests originating from a remote server.

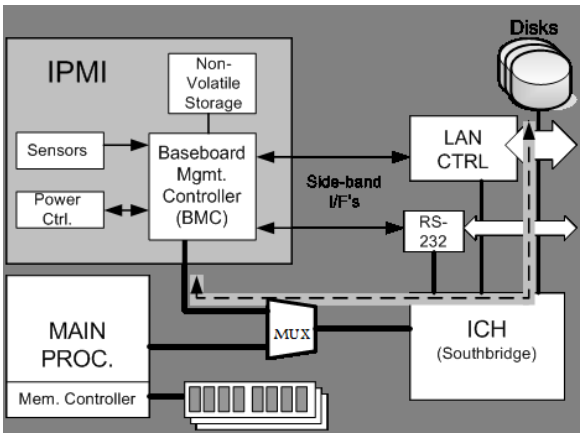


**Figure 1. An Overview of IPMI Organization**

## 3 Enhancing IPMI to act as Knight

In a traditional datacenter with direct-attached storage, a load balancer assigns a request to a server (let us say, $Server_A$). $Server_A$ may need to access multiple data blocks that may potentially be dispersed over many physical servers. $Server_A$ then initiates I/O requests to a remote server, say $Server_B$ that holds the necessary data. I/O requests from $Server_A$ enter the incoming request queue of $Server_B$ through its network interface. $Server_B$ fetches the request from the incoming queue and initiates a disk access request through it's I/O hub. Data from the disk is temporarily stored in main memory buffer pool using DMA. Once disk access is complete the buffer is then transferred to the $Server_A$ again using the network interface. This simple flow is modified with KnightShift. Let us assume that $Server_B$ is in a low power state (or even turned OFF) and it is unable to access the disk without moving to a higher power state. KnightShift intercepts all the network data entering $Server_B$ and places the data in a incoming request queue associated with BMC (say $BMC_B$). $BMC_B$ interprets the network request and recognizes the request as a data access request from $Server_A$. Instead of waking up $Server_B$ to perform I/O access it directly initiates a disk read request and buffers the data in memory associated with $BMC_B$. Since $BMC_B$ services the request, $Server_B$ can continue to sleep without being disturbed. $Server_B$ is woken up only when a request that can not be handled by $BMC_B$ is assigned to the server.

For KnightShift to work as described above, there are several design enhancements needed for existing IPMI.

1. **Interpreting I/O Requests:** BMC must be capable of correctly interpreting an I/O request embedded in a network packet, and translate these requests to disk accesses. Many IPMI implementations today support advanced features such as anomaly detection on network data without any support from the primary processor. BMC can decode IP packet headers and run complex computations for this purpose. Hence understanding the semantics of network I/O request is well within current BMC capabilities. BMC must also understand the file system implementation on the underlying disk. While a plethora of file systems are supported by the Oper-

ating System running on the primary processor, BMC may not have the capability to understand all these file system formats. In KnightShift BMC may support only a few popular file systems. But this is not a functional limitation since BMC can always fall back on waking up the primary server and letting it handle any request that is beyond BMC's capabilities.

2. **Direct I/O access:** The disk drives of the primary server must be independently powered ON even if the primary processor is powered down. Since current baseboard designs already use a different voltage source for disk drives, one would need small modifications to the baseboard design to keep them continuously on. In current IPMI implementations, BMC doesn't have access to the disks or the IO hub directly. For Knightshift, the IO hub must be decoupled from the primary processor. The decoupled hub must have a two-input mux connecting to the BMC and the primary processor. BMC would be selected only when the primary processor is in a low power state or OFF.

3. **Memory Isolation:** True benefits of KnightShift occur when the primary processor and its memory subsystem can both be placed in low power state when not in active use. Thus BMC must provide it's own memory for using as I/O buffer pool. As shown in Figure 1, current implementations of BMC already have an associated non-volatile storage, such as Flash, which may be used to buffer disk data during KnightShift operation. We assume that these additional writes have negligible impact on the lifetime of the Flash memory.

4. **Memory Coherency:** If the primary server is placed in low power mode with modified data residing in the I/O buffers in its memory then BMC may access stale data from disk. Hence, in KnightShift operation, before the primary server can be placed in a low power sleep state it must drain the modified data to disk.

5. **ISA Support:** Our eventual goal is to extend KnightShift to not only serve remote I/O requests but also regular requests assigned by the load balancer to the primary server. If a fraction of the requests assigned by the load balancer to the primary processor, particularly those requiring very limited compute resources, are also handled by KnightShift it can significantly increase the window of time the primary server can sleep. However, to satisfy these requests BMC must be capable of running the software layer that runs of the primary server. Such an option is feasible only if BMC uses the same ISA as the primary processor. In essence, this option significantly enhances BMC capability to make the server appear as two closely coupled heterogeneous servers, with the odd property that the two share nothing in common other than storage.

6. **Stateless Workloads:** Our approach is focused primarily on stateless request-response style workloads - i.e. workloads in which one request has no effect on how the next request is processed. This simplifies Knightshift since the Knight can process a request without worrying about any state in the memory of the primary server. Several e-commerce and web search applications use Java stateless session beans. Stateless sessions simplify recovery and process migration in datacenters. Hence stateless workload is not a severe limitation to our approach. For supporting stateful workloads, the state can be written to the disk in a pre-defined location and Knight can read that location before processing requests.

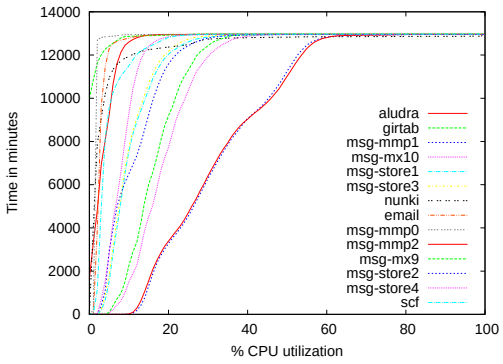


**Figure 2. CPU Utilization vs Time spent**

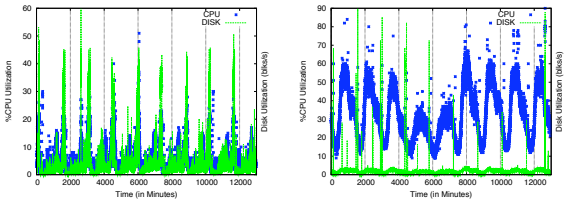


**Figure 3. CPU vs Disk Utilization for SCF (left) and MSG-MMP1 (right)**

## 4 Experimental Setup and Results

We collected minute-granularity utilization traces from USC's production datacenter for 9 days . The datacenter has several hundred servers across 95 racks and 5 chiller units (CRACs) in a raised floor organization. It serves multiple tasks, such as email, blackboard (digital support for class room teaching), distance education network (video and live streaming support of course lectures), compute intensive application hosting (such as CAD tools) and student timeshare servers. Each task is assigned to a dedicated cluster, with the data spread across multiple servers. We selected 8 clusters and 4 servers per cluster for trace collection. We were assured by our datacenter operations group that the selected servers within a cluster exhibit a behaviour representative of each of the server within that cluster. We used *sar* command with "-A" option to collect every system status that *sar* is capable of monitoring, which includes CPU utilization, memory utilization, disk utilization, paging activity, kernel and user activity.

### 4.1 Trace Overview

Figure 2 shows CPU utilization of all the servers traced in our study. We can see that some servers (aludra, nunki, scf,

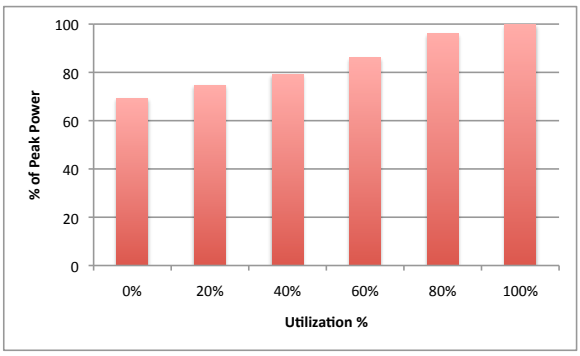


**Figure 4. Utilization vs Peak Power**

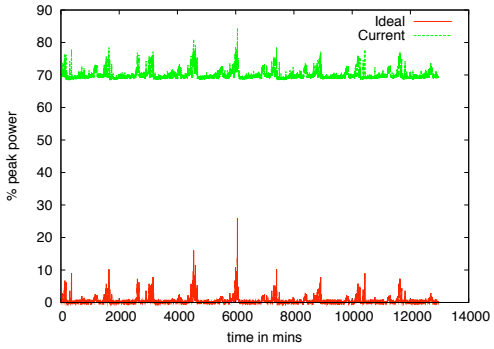


**Figure 5. Power Consumption in Current and Ideal Scenarios for SCF**

girtab, email, mmp0) run at less than 20%of CPU utilization for nearly 90% of their total operational time. Servers such as SCF are the primary NFS servers for all of the Engineering students. Figure 3 shows the CPU utilization and the number of disk blocks (512 byte) transferred per second for SCF server in Figure 3 for each one-minute interval. As can be seen clearly from this figure the CPU utilization and disk transfers are highly correlated. However, the CPU utilization necessary to service most data transfers is very low. For example, a utilization level of 10% is enough to satisfy nearly 50% of the I/O requests. Hence these servers are certain to benefit from KnightShift.

Another category of servers are those that require varying amounts of CPU during our 9 day observation window. For instance, the primary email server msg-mmp1 shown in Figure 3 graph on the right, is heavily accessed during the day time and even at the night time its utilization is never less than 10%. In the KnightShift approach since Knight may handle only low CPU utilization requests there may not be many opportunities to even turn ON the Knight to process request. In this scenario KnightShift will not provide much energy savings to the server. Note that due to space constraints we show only a few representative graphs in the paper. In general, these traces confirm that CPU utilization never reaches 100% CPU utilization but at the same time they also never go down to 0% utilization for extended periods of time, reaffirming prior studies [4, 13, 12] .

## 4.2 Energy Proportionality Impact on Energy Consumption

In current day servers, energy consumption is high even at low utilization due to lack of energy proportionality. We obtained the utilization versus power consumption data by analyzing results reported on the SPECWeb2009 webpage [9] for the best performing HP server in 2009 with solid state disks (SSD). This server uses a 3.2 GHz quad core CMP, with 96 GB of memory and 2 TB SSD. We used power workload from SpecWeb2009, which essentially characterizes server power at different CPU utilizations, to derive the energy proportionality curve shown in Figure 4 for our server. Since SSDs consume little energy when they are not being actively used, therefore power consumption curves at 0% utilization shown in Figure 4 correspond mostly to the power consumed by the processor, memory and other baseboard logic. The peak power consumption in the selected server is 715 Watts. Power consumption drops to 496 Watts at 0% utilization. We derive the power consumption at any given utilization (as a ratio of the peak power) using linear interpolation of the data shown in the energy proportionality figure.

In Figures 5, 6 we show the power consumption as a function of time for SCF and MSG-MMP1 trace. The curve labeled *Current* shows the power consumption in our baseline where the server power follows energy proportionality trends shown in Figure 4 for a current day server. In order to generate this curve we simply transformed the CPU utilization data of the trace to the power consumption using the interpolation described above. The curve labeled *Ideal* shows the power consumption of an idealized server where power consumption scales quadratically with utilization, i.e. at 0% utilization the power consumption is zero, at 10% utilization the power consumption is 1/100 of the peak power, and at 50% utilization the power consumption is 1/4 the peak power.

In Figures 7, 8 we plot KnightShift power consumption assuming that it can satisfy any request that needs 10% of the primary server. We assume KnightShift satisfies only those requests that require less than 10% of the primary CPU's power. Since the traces lack information at the request boundary it is difficult to show results on how many I/O requests Knight satisfies in our trace. Instead we just use the CPU utilization at 1 minute time interval to decide whether Knight can satisfy that utilization level. For instance, if the CPU utilization is 10% in a given time interval in the trace then we assign the jobs in that time interval to the Knight and allow the primary server to go to sleep. If during the next time interval the CPU utilization grows above 10% then Knight wakes up the primary server to process the request. We assume that there is a wakeup penalty of 1 minute. We also assume that the 10% Knight requires only 1% of the primary server power. Note that unlike scaling the primary server voltage/frequency, Knight is built from ground up to operate at a much lower voltage and frequency that satisfies all requests that require less than 10% utilization on the primary server. Hence we argue that Knight power consumption assumption is not overly aggressive.

The power consumption of SCF server (Figure 7) drops dramatically.We assume, conservatively, at the time of wakeup during the 1 minute wakeup delay the primary server

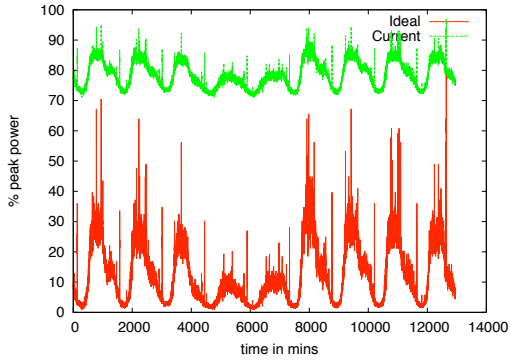


**Figure 6. Power Consumption in Current and Ideal Scenarios for MSG-MMP1**

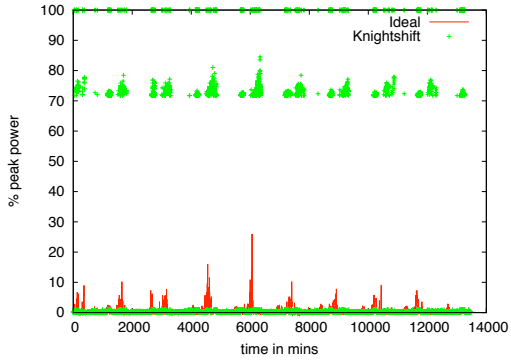


**Figure 7. Power Consumption with KnightShift and Ideal Scenario for SCF**

consumes 100% of the peak power. Hence, the occasional peaks that are visible at 100% show the time when the primary server is being woken up. Once woken up the server goes to sleep only if the utilization drops below 10%. As can be seen the 10% KnightShift curves allow the primary server to sleep for very long periods of time and allows the overall system to achieve nearly as good as ideal power consumption curve.

| Knight Capability | Time | Energy Savings |
|---|---|---|
| 5% | 1.11 | 2.59 |
| 10% | 1.04 | 5.01 |
| 15% | 1.02 | 8 |
| 20% | 1.01 | 10.38 |
| 25% | 1.00 | 12.02 |

**Table 1. Performance degradation and Energy Benefits of Knightshift w.r.t baseline for SCF**

Table 1 shows performance degradation and energy gains of KnightShift with respect to the baseline for SCF. For example, the first row says that if the Knight is capable of handling requests that need 5% of the primary server's utilization, KnightShift reduces energy consumption by 2.59X while increasing the latency of servicing the requests by 11

As chip designers and system builders realize the need for

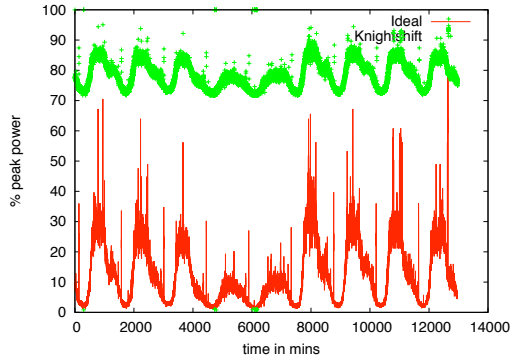


**Figure 8. Power Consumption with KnightShift and Ideal Scenario for MSG-MMP1**

energy proportionality it is only a matter of time before new approaches are developed to reduce power consumption with low utilization. For instance, memory power can be curtailed by putting DRAMs into very low power self refresh states, whenever they are not actively used. Similarly, we surmise that CPU power consumption will be reduced through techniques such as EPI throttling [3]. While these techniques will certainly help reduce the power consumption there is a fundamental limit to achieving energy proportionality if the underlying hardware has to perform well across a wide range of utilizations. Scaling voltage is already becoming restrictive. Hence DVFS is becoming just a frequency scaling technique, where power consumption reduces linearly with frequency, rather than quadratically with voltage and frequency scaling. Similarly, a large server core can be transformed into a smaller core by dynamically reducing the size of caches, branch predictors and TLBs during low utilization periods. However, the power consumption of a dynamically scaled small core is never going to be as small as building a small core from the ground up. The length of wires and routing across large blocks of logic, even when they are not used, cause the power consumption to not reduce quadratically with reduced utilization. Hence, we surmise that in future even with a perfectly proportional system the power consumption can not scale quadratically as we assumed in our Ideal scenario.

KnightShift, on the other hand, enables a new level of customization by enabling designers to focus more on providing higher performance in the primary server rather than trying to achieve energy proportionality at a range of utilizations. The designers can focus exclusively on designing the Knight system to serve the system under low utilization. Since KnightShift does not have to worry about providing high performance, it may be possible to operate the system at a much lower voltage and frequency.

## 5 Related Work

Barroso [4] showed that there is a need to make servers more energy-proportional since most enterprise servers operate at low average utilizations, at which they are not very energy-efficient. Fan et al. [13] showed that power provisioning of datacenter requires understanding the dynamic be-

havior of the servers. They showed that CPU utilization is a good proxy metric for server consumption. Ranganathan et al. [12] showed that more power savings opportunities exist at the ensemble level rather at individual server level. They take advantage of inter-server variations to show that managing the power budget at an ensemble level allows active server to steal power from inactive servers. Our research is influenced by all these prior studies. However, we believe that significant energy saving opportunities exist at the server level by making use of the management processor to allow long server sleep times.

Hybrid Datacenter designs have been proposed [5] which use heterogeneous platforms to save power under low utilization levels by migrating tasks from a high-power, high performance system and a low-power, low-performance system and shutting down the higher power servers when not needed. However, this approach may not work in the event that the high-power system has critical data on its disk which needs to be processed, as is the case in data-attached-storage architectures. PowerNap [11] is an approach to energy-efficient design in which the entire system rapidly transitions between a high and a low power state. Design goals of such a system would be minimizing idle power and transition times rather than having multiple fine-grained power states. KnightShift will benefit from PowerNap by reducing the wakeup delay but it is not necessary for KnightShift operation. Barely-alive servers [1] use a small embedded processor to only keep the memory of idle servers active so that they can be used for caching by rest of the servers in the cluster. This idea is similar to Knightshift, but in Knightshift the focus is on keeping the disks with critical data always active, and memory is shut down. FAWN [2] is a key-value storage system built using many "wimpy" flash-based nodes as an alternative to using conventional, heavyweight disk-based systems. It exploits the inherent parallelism available in the data access patterns of small-object random-access workloads leading to a more efficient design. Their approach is focused on using flash to improve energy efficiency. KnightShift does not make any assumptions of the underlying storage architecture. If a flash based storage is used KnightShift will have even more benefits than what has been shown in this research.

## 6 Conclusions

In this research we present KnightShift, a novel approach to repurposing a management processor to handle requests that require limited processing power of the primary server. By shifting the burden to the Knight system we enable the primary server to sleep for extended periods of time. We described the architecture of KnightShift and showed that only a few simple modifications are necessary to the management processor to enable KnightShift. Using a broad selection of traces collected from a production datacenter we show that KnightShift can reduce the energy consumption by nearly an order magnitude when the primary server utilization is low, with less than 12% increase the response time. In future we plan to conduct a thorough state space exploration of various design choices for KnightShift.

## Acknowledgements

This work was supported by NSF grants CCF-0834798, CCF-0834799. The authors would like to thank Partha Ranganathan for providing initial feedback.

## 7 References

[1] V. Anagnostopoulou, S. Biswas, A. Savage, R. Bianchini, T. Yang, and F. T. Chong. Energy conservation in datacenters through cluster memory management and barely-alive memory servers.

[2] D. G. Andersen, J. Franklin, M. Kaminsky, A. Phanishayee, L. Tan, and V. Vasudevan. Fawn: a fast array of wimpy nodes. In *Proceedings of the symposium on Operating systems principles*, pages 1–14, 2009.

[3] M. Annavaram, E. Grochowski, and J. Shen. Mitigating amdahl's law through epi throttling. *Proceedings of the 32nd International Symposium on Computer Architecture*, pages 298–309, June.

[4] L. Barroso and U. Hlzle. The case for energy-proportional computing. *Computer*, 40(12):33–37, 2007.

[5] B.-G. Chun, G. Iannaccone, G. Iannaccone, R. Katz, G. Lee, and L. Niccolini. An energy case for hybrid datacenters. *SIGOPS Oper. Syst. Rev.*, 44(1):76–80, 2010.

[6] download.intel.com/design/servers/ipmi/IPMIv2_0rev1_0.pdf. Ipmi v2.0 specifications document revision 1.0. Retrieved April 2010.

[7] http://h18000.www1.hp.com/products/servers/management/ilo/. Hp integrated lights-out (ilo) standard. Retrieved April 2010.

[8] http://publib.boulder.ibm.com/. Integrated management module user's guide. Retrieved April 2010.

[9] http://www.spec.org/web2009. Specweb2009. Retrieved April 2010.

[10] A. Kumar, P. Goel, and Y. Saint-Hilaire. *Active Platform Management Demystified: Unleashing the power of Intel vPro (TM) Technology*. Intel Press, 2009.

[11] D. Meisner, B. T. Gold, and T. F. Wenisch. Powernap: eliminating server idle power. *SIGPLAN Notices*, 44(3):205–216, 2009.

[12] P. Ranganathan, P. Leech, D. Irwin, and J.Chase. Ensemble-level power management for dense blade servers. *Comput. Archit. News*, 34(2):66–77, 2006.

[13] X.Fan, W. Weber, and L. Barroso. Power provisioning for a warehouse-sized computer. In *Proceedings of the 34th annual international symposium on Computer architecture*, pages 13–23, 2007.