DSWITCH: A Dual Mode Direct and Network Attached Disk

Quanlu Zhang
Peking University
zql@net.pku.edu.cn

Yafei Dai
Peking University
dyf@pku.edu.cn

Lintao Zhang
Microsoft Research
lintaoz@microsoft.com

Abstract

Putting computers into low power mode (*e.g.,* suspend-to-RAM) could potentially save significant amount of power when the computers are not in use. Unfortunately, this is often infeasible in practice because data stored on the computers (*i.e.,* directly attached disks, DAS) might need to be accessed by others. Separating storage from computation by attaching storage on the network (*e.g.,* NAS and SAN) could potentially solve this problem, at the cost of lower performance, more network congestion, increased peak power consumption, and higher equipment cost. Though DAS does not suffer these problems, it is not flexible for power saving. In this paper, we present DSWITCH, an architecture that, depending on the workload, allows a disk to be attached either directly or through network. We design flexible workload migration based on DSWITCH, and show that a wide variety of applications in both data center and home/office settings can be well supported. The experiments demonstrate that our prototype DSWITCH achieves a power savings of 91.9% to 97.5% when a disk is in low power network attached mode, while incurring no performance degradation and minimal power overhead when it is in high performance directly attached mode.

Categories and Subject Descriptors D.4.2 [*Storage Management*]: Storage hierarchies

General Terms Design, Performance

Keywords Power proportional, NAS/SAN, Energy, Service migration, Network, ARM

1. Introduction

Power consumption of computing infrastructure has become one of the major concerns in households, offices, and data center environments. Modern PCs support various low power modes, such as ACPI states S3 (suspend-to-RAM) and S4 (hibernate) [1]. These low power modes can significantly reduce the power consumption of a computer when it is idle, which normally might use up to 60% of the peak power.

Still, users tend to leave the machines on even when they are not actively used. Among the reasons why a machine is left on [38], a main one is the need to access data stored on its disks. In an office setting, colleagues who work in a different time zone might want to access shared files, while background activities such as backup services and virus scans often need to access the disks during off hours. At home, family members might want to watch streaming video on TV or look at photos on tablets, even though the media files are stored on home PC. Moreover, P2P clients such as BitTorrent might be downloading files from the Internet. Though such applications only need minimal computational power, they still force the PC to be in the wakeup state and thus consume significant amount of energy.

The situation is similar in data center and enterprise settings. Data center power consumption is undergoing alarming growth. Study [30] shows that much energy is wasted by idle servers. Most online services hosted in data centers show significant diurnal variation in load levels [38]. This causes low server utilization in the range of only 20-30% in typical deployments [16, 19]. Virtualization and live VM migration techniques have been developed to migrate computing tasks to consolidate servers when workload is low [13, 17, 23]. However, modern data centers are filled with commodity servers with their own computational and storage resources [26]. It is relatively easy to migrate computation transparently, but it is not practical to migrate data daily for load consolidation.

Directly attached storage (DAS) co-locates storage with computation. To decouple these two resources, network storage (such as network attached storage and storage area network) has long been studied and deployed. Network storage uses dedicated storage servers to make data available on the network, thus allows computational servers to be taken offline freely. However, network storage systems come with their own downsides. Network storage is expensive, often requires its own dedicated network fabric. The performance

SoCC '15, August 27 - 29, 2015, Kohala Coast, HI, USA
 ISBN 978-1-4503-3651-2/15/08$15.00
DOI: http://dx.doi.org/10.1145/2806777.2806850.

of network storage is limited by the network performance, which is often not as good as modern local storage systems (such as fast SSDs). When mixed with regular network traffic, networked storage may increase network congestion and add latency unpredictability. Moreover, for a well provisioned system, network storage may require higher power during peak load, due to extra energy spent on pushing the bits through network. Due to these limitations, dedicated network storage is not a replacement of DAS. At home, even though cheap NAS solutions are available, most home PCs still come with local disks for better performance, which is mostly unused when the PCs are offline. In data centers, many cloud computing facilities use commodity servers with directly attached disks, and build network file systems as well as computation capacity on top of these servers [26].

This paper describes DSWITCH, a novel storage attachment architecture that allows a disk to be operated and switched between DAS mode and NAS mode, to achieve high performance under load while still allowing the decoupling of storage and computation when idle. When the workload is high, DSWITCH works in the DAS mode to provide the performance of a directly attached disk with negligible power overhead. When the workload is low, DSWITCH can be switched into the NAS mode, to allow access of the data stored on the disk through the network, while the computer sleeps to save power.

We built a DSWITCH prototype leveraging a SATA multiplexer and an auxiliary low power computer. SATA multiplexer allows a regular SATA disk to be electrically connected with either the main computer (MC) or the low power computer (LC), so that the disk can be mounted by either one (but not simultaneously). MC and LC are both connected to the network. Both MC and LC can initiate mode switch and they can wake up each other from sleeping. Mode switching has minimal impact on other computers. From a remote host's point of view, often a mode switch is just a temporary network service outage. In our prototype, mode switching takes less than 30 seconds to complete for most cases. For many workloads such as file sharing and BitTorrent, reconnection mechanism can help automatically recover from the temporary failure.

DSWITCH achieves significant power saving when the workload is low. The low power computer in our prototype only consumes 2.8 watts in peak load, while the main computer consumes tens of watts when idle. In our experiments, we find that DSWITCH can save 91.9% to 97.5% of power while still allowing the machine to be reachable for lightweight services such as remote desktop connection, P2P file sharing, video streaming, and background downloading.

This paper makes the following contributions:

- We present DSWITCH, a novel storage attachment architecture that combines the benefit of directly attached and network attached storage. When workload is high, DSWITCH provides the same high performance as directly attached storage with negligible power overhead. When workload is low, DSWITCH allows a computer to freely enter low power mode to save energy, while still allowing remote hosts to access data through network.
- We implemented a DSWITCH prototype with off-the-shelf components and extensively investigated its power and performance profile.
- We evaluated DSWITCH in multiple application scenarios in home, office and data centers, and find that the architecture introduces minimal service disruptions and provides sufficient performance as well as significant power savings for various practical applications.

2. Why DSWITCH ?

Power consumption has become considerable expenditure of home/enterprise and data centers [32]. With the price drop of hardware, power saving would become crucially important. In order to reduce power consumption of computers, several power management techniques have been developed over the years.

Dynamic Voltage and Frequency Scaling (DVFS) technique adjusts CPU voltage and frequency based on varying workload. Though CPU becomes power-proportional, it actually draws only about 25% of the total power for a typical computer [39]. Other components, such as motherboard, memory, fans, still consume substantial power, which is very challenging to be power-proportional. The burgeoning heterogeneous multi-core processor also suffers the same problem. Therefore, switching the main computer as a whole to a low power computer becomes a favourable approach, especially given the fact that the low power computer is extremely power-efficient, consuming about the same power as a PC in S3 state (see Table 1), while providing enough computational resource for disk service and light-weight workload.

Modern computers also support Advanced Configuration and Power Interface (ACPI) [1]. However, though ACPI state S3 (suspend to RAM) and S4 (hibernation) offer great reduction in power draw, a machine in S3 or S4 totally loses the presence on the network, thus cannot provide service (*e.g.,* storage access) any more. Techniques such as Wake-on-LAN (WoL) [7], dedicated proxy machine [31], can wake up a sleeping machine remotely. However, it is usually overkill to wake up the machine for data access or light-weight workload. Also this approach does not work for the typically interactive servers in data centers, most idle intervals of which are under one second although 60% fraction of time is idle [30].

Local data access is essential for the performance of applications in desktop scenario, as well as the efficiency of computing jobs (*e.g.,* big data analytics) in data centers, due to its high throughput and low latency. DSWITCH guarantees this benefit, while also having the remarkable properties of NAS. That is, DSWITCH can flexibly couples and decouples

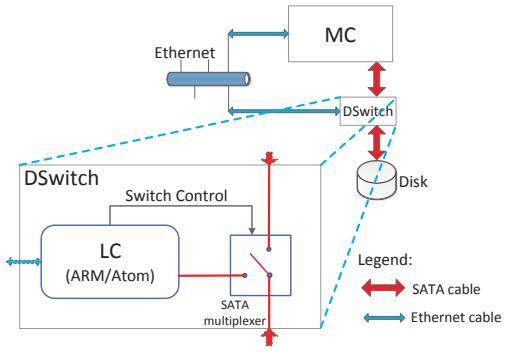


Figure 1: DSWITCH architecture.

storage and computation, which makes it possible to design much more efficient power saving mechanisms for various scenarios without introducing additional overhead (see § 4).

3. Design of DSWITCH

DSWITCH is a disk attachment architecture, the design of which is primarily motivated to achieve the following goals:

- When the workload is low, allow the host computer to sleep or be turned off to save power, while still maintaining data accessibility on the network with reasonable performance.
- When workload is high, achieve the same performance and power profile as a directly attached disk.
- The switching process should only have minor disruption on the availability of data and services. More importantly, it should not cause data corruption on the disk.

In the following discussion, we assume that the disks have serial ATA (SATA) interface, which is by far the most widely used interface for hard disks and SSDs. DSWITCH for other disk interfaces, such as SAS, can be implemented similarly.

3.1 Design Overview

DSWITCH is the co-design of hardware architecture and software management. On the hardware side, in order to minimize the modification to current machine, we design DSWITCH as a plug-in module, connected to the machine through a SATA cable as shown in Figure 1. Specifically, DSWITCH sits between a disk and a host computer (*i.e.,* MC), and adds an alternative path from the disk to the network. Inside DSWITCH, there are two main components: a low-power computer (*i.e.,* LC) and a SATA multiplexer. The disk is connected to the downstream of the SATA multiplexer, while the two upstreams connect to LC and MC respectively. LC has a network interface that connects itself to the network, it also controls the SATA multiplexer.

DSWITCH works in one of two modes in Figure 2. In the directly attached mode, the SATA multiplexer connects the disk to MC directly, guaranteeing high local data access, while LC can sleep. In the network attached mode, the SATA

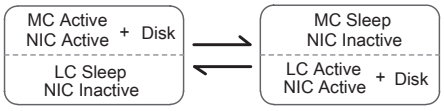


Figure 2: State transition.

multiplexer connects the disk to LC, while MC is put to sleep to save power. We could (but not always need to) take advantage of the fact that MC's network interface is inactive in S3 state, and let LC assume the same network identity (*i.e.,* name, IP and MAC), in order to reduce disruption of the services to the external world.

It is challenging to design the software part for this new hardware platform. First, services on MC should be able to be offloaded flexibly, even facing heterogenous hardware and operating systems between MC and LC. Second, the induced service interruption should also be controlled within an acceptable range for different scenarios. Therefore, we design two switch modes: *service migration* and *VM migration*.

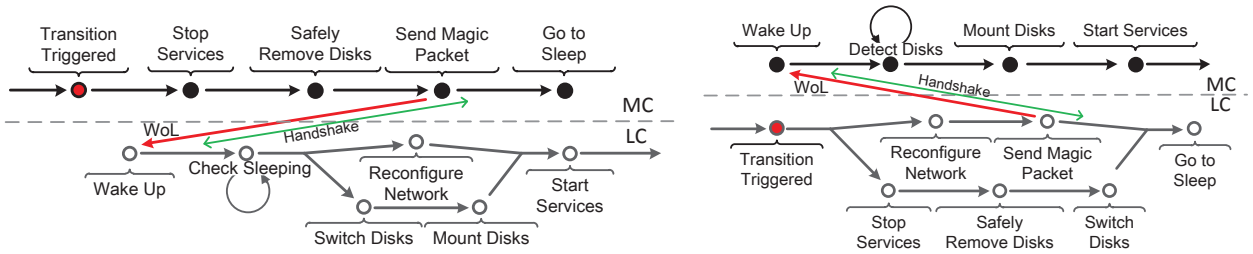
Service migration directly migrates services from MC to LC, which is mainly for light-weight and cross-platform applications, including CIFS file sharing (Samba)/NFS, BitTorrent, web server, and so on. Since LC is a full blown computer, it can run many of these standard services without any issues. Such service migration is often sufficient for home and office.

VM migration migrates services from MC to another MC, while LC only serves data access. Because LC has a vastly inferior CPU with a potentially different instruction set. It might run a different OS that is not compatible with the OS on the MC. For platform dependent software or the services that require high CPU performance (*e.g.,* database server), another computer with similar configuration as MC is more preferable. In this case, LC will expose the disk through a storage protocol such as NFS, iSCSI, and a remote virtual machine can mount this disk and provide the required services.

3.2 Switch with Service Migration

It is non-trivial to design a smooth and robust transition between MC and LC. First, there is no dedicated channel for them to communicate with each other and coordinate their actions. Second, they have to use the same network identity to provide services, which means they cannot be online simultaneously. Third, their actions should be carefully organized to reduce service interruption and avoid data corruption.

We use the disk as a communication medium which allows MC and LC to exchange information, such as the list of to-be-resumed applications and their states. Currently, the migration is performed on a service-by-service basis. The programmer has to write the code to generate the script for



(a) From MC to LC (b) From LC to MC

Figure 3: Switch with service migration.

service resuming. It is easy for stateless services, just the command of starting the service is enough. However, for stateful services, the script should include state recovery which needs to be supported by the service (*e.g.,* wget). This migration design is more general and practical than live process migration, especially on different CPU architectures.

On the other hand, we relax the restriction that MC and LC should be online mutually exclusively, since it makes the switch process very vulnerable. The idea is we give two different network identities to them, thus, they can communicate with each other during the switch process, guaranteeing correct execution or implementing roll-back if necessary, which greatly narrows the failure window.

The actions of MC and LC are carefully scheduled and we add necessary parallelism to them, as shown in Figure 3. The trigger event can be generated by the user or a load monitoring daemon. Moreover, since MC and LC are not equal (*i.e.,* only LC controls the SATA multiplexer), the operation sequences are different in the two directions.

From MC **to** LC**.** The steps of MC and LC are shown in Figure 3(a). The disk is safely removed (using *HotSwap!* [3] in Windows or *umount* in Linux) before switching. In the whole process, MC need not change its network identity. For LC, it wakes up after receiving the magic packet. At this time, it is still using the network identity of its own. Then LC reconfigures its network identity to be the same as MC, after making sure that MC has become unreachable. At the same time, it sends signals through the GPIO pins to the SATA multiplexer to switch the disk. Then it is able to start the corresponding services.

From LC **to** MC**.** Switching from LC to MC is a little different from the process above. As shown in Figure 3(b), when LC receives the switching command, it reconfigures its network back to its own network identity, and sends a magic packet to wake up MC. Meanwhile, it stops the services, and then removes the disk safely and switches the disk back to MC. At last, LC enters sleep state. On MC's end, it is waken up by the magic packet and starts to detect the disk. Then MC mounts the disk and starts the services.

We guarantee the correct execution of switch operation through a handshake between MC and LC, and implement rolling back if failure occurs. Taking switching from MC to LC for example, after LC wakes up, it builds a socket connection with MC and reports its healthiness, then MC can go to sleep. If a failure occurs after this handshake, LC will execute the reverse switch to roll back.

The operation flow can handle the most common case when the disk is not used for OS files of MC. For the OS disk, we have to make sure that the disk is switched to LC after MC turns into S3 state and that MC is woken up after the disk is switched back. An alternative approach for dealing with OS disk would be to run MC root from a RAMdisk, loaded from a network service.

3.3 Switch with VM Migration

The switch process shown above is designed for a single DSWITCH unit, it is suitable for desktop PCs at home or in office, and is sufficient to support lightweight services that can be provided by both MC and LC. Though the same process is applicable in data centers as well for lightweight services, we take a different approach for workloads that are CPU and memory demanding (*e.g.,* database service for business oriented queries), or are platform dependent. In this case, we might want to enlist a spare computer that has compatible configuration as MC to carry out the workload. Most often, the spare computer would be another computer in the data center that is also lightly loaded.

We use the idea of virtual machine consolidation which has been widely applied in data centers for power saving. We migrate the workload (*e.g.,* services or VMs) from the physical server where it originally resides, called origin-server, to another physical server called step-server. There are two ways to implement the consolidation. For the first way, the origin-server runs VMs to provide service, and we simply stop the VMs on the origin-server and restart them on a step-server. For the second way, the service runs directly on the origin-server, and we migrate this origin-server to a step-server to be a VM. We describe the switch process of the second way in detail below, the first one can be implemented similarly.

Here, a coordinator is employed to coordinate the actions of servers. It makes switch decisions based on pre-

determined policies, and sends commands to the machines accordingly. Moreover, it monitors the switch process in order to find failures timely. When a failure occurs, the origin-server, step-server and LC are coordinated to implement rolling back using the reverse direction switch process. On the other hand, LC in this case only takes charge of data service (*e.g.,* expose the disk through iSCSI), thus, it uses its own network identity all the time.

The detailed switch processes are shown in Algorithm 1 and Algorithm 2. For switching from MC to LC, the coordinator triggers the switch operation and picks a machine as the step-server. Then the origin-server as well as its services is migrated to the step-server, running as a VM. There is no communication between origin-server and step-server, they exchange the information, such as network identity, iSCSI Qualified Name (iqn), through the coordinator, which makes it easy to control their actions. The reverse switch is a similar process.

Algorithm 1 Switch from MC to LC

```
1: Origin-Server (MC):
2:     receive switch command from Coordinator
3:     stop services; safely remove disks
4:     send magic packet to wake up LC
5:     handshake with LC; go to sleep
6: LC:
7:     wake up; switch and mount disks
8:     start iSCSI target to expose the disks
9:     report the completion to Coordinator
10: Step-Server:
11:    receive network conf. of MC from Coordinator
12:    start a VM configured with the network conf.
13:    the VM: mount the disks through iSCSI initiator
14:    the VM: start the services
15:    the VM: report the completion to Coordinator
```

Algorithm 2 Switch from LC to MC

```
1: Step-Server:
2:     receive switch command from Coordinator
3:     shut down the VM; notify LC
4: LC:
5:     receive the notification
6:     stop iSCSI target; safely remove and switch disks
7:     send magic packet to wake up MC
8:     handshake with MC; go to sleep
9: Origin-Server (MC):
10:    wake up; detect and mount disks
11:    start services
12:    report completion to Coordinator
```

Stopping services and removing disk safely are sufficient to avoid data corruption. However, on-the-fly operations inevitably will fail for a short period of time during switching operation. Depending on the applications, this outage could potentially be mitigated by temporarily redirecting requests to replicas before switching.

4. Switch Scheduling

Though in high performance mode disks are directly attached to servers, DSWITCH decouples them by adding additional path for disks onto the network. Usually both computers and clusters have to provision enough resource for the peak load, which actually induces over-provision most of the time. Thus, DSWITCH architecture can achieve significant power savings in various scenarios, while still guaranteeing equal or better performance (due to data locality). The scheduling mechanisms based on DSWITCH tend to be much less complex, since there is no need to worry about data availability, consistency, and transfer.

Here we explicitly elaborate three representative use cases and their scheduling mechanisms to show the easy usage and significant power efficiency of DSWITCH.

4.1 Office Scenario

As illustrated at the beginning, computers in companies tend to be left on for working at home or sharing files with colleagues. Several approaches have been proposed to address energy waste in such situation [13, 17], however, they need data transfer, or resource virtualization and migration.

DSWITCH makes power saving in office scenario very simple and efficient. Note that data is the central resource, while computing resource (*i.e.,* computers) exists almost everywhere. Thus, we can work with any PC, as long as the data is available, rather than powering on the powerful computer in the office. In this case, the computer in the office is switched to LC mode, which exposes the storage on the network with significantly low power. Then we can freely mount the storage remotely using any PC.

However, for those "always-on" [13] but platform-dependent applications and services, we would like to employ an alternative approach, *i.e.,* setting up a server in the company for desktop consolidation, which can almost deal with all situations in office scenario. The consolidation is much easier to be implemented compared to that in other works. Specifically, when a computer turns into LC mode, its storage is exposed by LC. Then the server starts a VM which mounts the storage and runs the required applications. Each VM possesses limited resources (*e.g.,* CPU, memory), if its resource utilization exceeds a threshold, it will be shut down and the corresponding MC will be waken up. The switch operation can be triggered by users as well. Actually, the two approaches above can be deployed together, users can freely choose the one they want.

We can estimate the expected power savings (see measured numbers in § 7.1). The power saving of DSWITCH in the first approach is $\frac{E_M - E_L}{E_M}$, where E_M is the power consumption of a computer in idle and E_L is LC's power consumption. For the second approach, since a physical machine's power draw can be estimated by a linear function of its load [21], we assume the server consumes $E_{idle} + nE_i$, where E_{idle} is the idle power of the server, E_i is the added

power draw of one VM on average and n is the number of the computers using this approach. Thus, the power saving can be expressed as $\frac{nE_M-(E_{idle}+nE_i+nE_L)}{nE_M}$.

4.2 Three-Replica Storage

Three-replica has been widely applied in modern systems, and many works proposed power proportional mechanisms based on changing the number of online data replicas [14, 38, 42]. For example, Sierra [38] rearranges the placement of replicas, in order to power off additional data copies (*i.e.,* low gear mode) when the load of the cluster is low without sacrificing data availability. It achieves significant power savings. However, Sierra has to use distributed virtual log (DVL) to support writing to those off-line nodes, in order not to violate the placement rule of replicas. This means the data should be first recorded on the online nodes in three-replica and be written back when the corresponding nodes are online again, which inevitably induces the waste of network and disk I/O resource and possibly impacts online services.

DSWITCH can achieve significant power saving even without modifying such systems, since it makes data accessibility not a concern any more when powering off nodes. We demonstrate the implementation of Hadoop on DSWITCH platform as an example. Each node in Hadoop runs a tasktracker and a datanode. We determine the number of online nodes according to the load of the cluster, using the load prediction technique [38, 43]. When a node should be turned off, we stop the tasktracker and migrate the datanode onto LC.

The online nodes are randomly selected. When the cluster is shifted up, we randomly select some off-line nodes to join the online group. When the cluster is shifted down, we select some nodes randomly from the online group to power off. Though it is possible to involve data locality in the node selection for better performance, the random node selection is usually enough (see § 7.5.3 for detailed numbers).

On the other hand, if we use extremely low power ARM as LC that can only provide limited throughput for data access. The write performance would be degraded greatly if the datanode in LC mode serves the write, and so does the read performance. In this case, we can modify namenode of HDFS to schedule write only to the online nodes. This still maintains the random data placement since the online nodes are randomly selected. Also it is possible to serve read using multiple replicas in parallel. Therefore, the total read throughput can be expressed as $nT_M + (N - n)T_L$, where n and N are the numbers of online nodes and all nodes respectively, T_M and T_L are the read throughput provided by MC and LC. The total write throughput becomes $nT_w/3$, where T_w is the write throughput of MC. We do not implement this special design, since according to our evaluation the LC that can provide the throughput of 30MB/s already shows good performance.

4.3 VM Consolidation

VM consolidation is another typical scenario in modern data centers. It is often implemented based on NAS to avoid cumbersome disk migration, but NAS has the drawbacks as discussed before and sacrifices high local data access. DSWITCH, however, supports VM consolidation without suffering these drawbacks. Here we do not propose a new consolidation mechanism, since there have been lots of such mechanisms [18, 35, 41] which can be deployed on DSWITCH platform seamlessly. More importantly, data locality is greatly improved in our implementation. We consolidate VMs among each rack and randomly select the online nodes. So, n/N VMs on average have local data access, and other VMs can obtain rack locality.

In the current implementation, VM consolidation is implemented by simply stopping and restarting the VMs. Live VM migration has not been supported (see § 5).

5. Discussion and Future Work

Security and Management. We have not covered security issues in the discussion above. Data security and access control are very important issues that must be addressed in practice. However, these issues are largely out of the scope of this paper. Since both LC and MC are complete computers running full blown operating systems, they can implement any security and data access control measure demanded by IT policy. Similarly, computers in large offices and data centers often have management utilities to facilitate tasks such as network configuration and policy deployment. Again, we are not discussing machine management issues here because they can be implemented on DSWITCH in a similar way as on regular PC.

Choice of LC. There are many different low-power computers available in the market [37]. If we only use LC for file sharing, a low performance ARM computer is enough. For applications and services that demand high performance, low-power X86 computers like those based on Atom CPUs can be used. Running the same OS on CPUs with the same instruction set on both LC and MC opens doors for many opportunities, such as supporting the same software release. The choice of LC's computational power largely depends on the supported services and their workload patterns. But, usually the LC, that can provide similar data access performance to MC, is able to support most services in data centers.

Live VM Migration. Currently DSWITCH is not suitable for the services that cannot tolerate interruption. A possible way is running them in VMs, and implementing live VM migration. DSWITCH could support live VM migration, if we use a layer to mask disk switch operation. Take NFS for example, when switching a disk, the corresponding NFS clients first block the I/Os to this disk. Then the disk is switched and exposed again, after which the NFS clients update their NFS connections and resume the blocked I/Os.

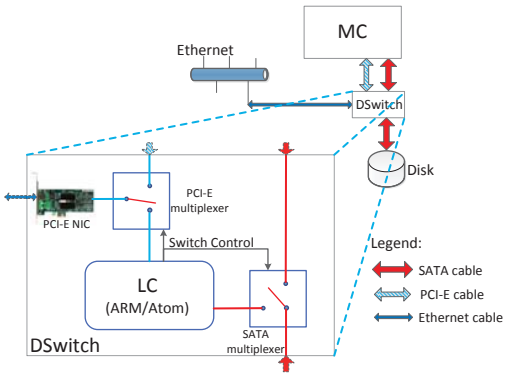


Figure 4: Alternative DSWITCH architecture.

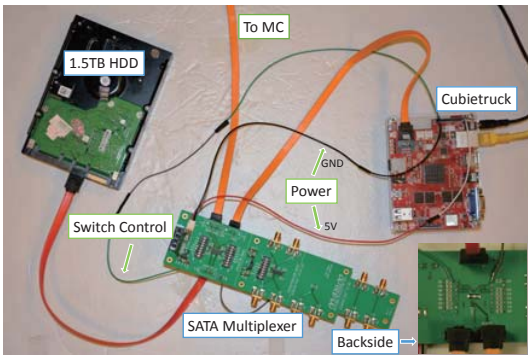


Figure 5: DSWITCH prototype.

This can be accomplished by modifying NFS, we leave this in future work.

Reducing Networking Cost. Current DSWITCH design has two network interfaces belonging to MC and LC respectively, which means a system needs two NICs and two network switch ports. This could be expensive in certain cases, *e.g.,* for 10GbE deployments or in data centers where the cost of the network switch is based on port count. Here we propose an alternative design whose implementation will be left for future work. As shown in Figure 4, MC and LC share the same NIC and the NIC can be switched between them with a PCI-E multiplexer (*e.g.,* MAXIM MAX4969). LC controls the PCI-E multiplexer in the same way as it controls the SATA multiplexer. The switch process might be simpler, since we can omit network reconfiguration and MAC address spoofing, which could also greatly reduce the switching time. However, we cannot use WoL to wake up machines in this design. A different side-channel, such as a USB cable between MC and LC is needed for the machines to wake up and communicate with each other. It is also possible to integrate DSWITCH in motherboard or NIC directly, and use the side-channel for communication and notification to simplify switch process and reduce switch latency.

6. Prototype Implementation

We implemented a DSWITCH prototype using an ARM-based low power computer called Cubietruck [2] due to its support of Gigabit Ethernet and native SATA port. It is equipped with AllWinnerTech A20 dual-core Cortex-A7 ARM CPU, 2GB DDR3 SDRAM, and is supported by several Linux distributions. We run Lubuntu-12.10-Desktop on Cubietruck for most of the tests. Due to lack of good iSCSI support in Lubuntu, we install Fedora19 on Cubietruck for iSCSI tests and also use Fedora on MC for the same set of tests. Cubietruck has tens of general purpose input and output (GPIO) pins. We use GPIO to control the SATA multiplexer. Both Cubietruck and MC are connected to a 1 Gbit/s switch. Cubietruck do not currently support WoL, so we leave it on even when the disk is switched away. A daemon is listening on a port on Cubietruck. When MC wants to wake up Cubietruck, it sends socket message to this port instead of sending WoL messages.

We use MAX4986 [1] Evaluation Kit from MAXIM as the SATA multiplexer in our prototype. We move the three SATA connectors to the backside of the EV kit in order to operate it with two hosts and one device (the original board is configured for one host and two devices). We connect the control switch (SEL) on the EV kit to a Cubietruck's GPIO pin. The SATA port in MC needs to be set to AHCI mode in BIOS to support hot swap. Since Cubietruck supports disk hot swap, no additional setting is required. The power supply of the EV kit is provided by the 5V pin on Cubietruck.

7. System Evaluation

In this section, we evaluate the power and performance of DSWITCH, and also evaluate various applications and services implemented on DSWITCH. Then we evaluate the performance and power savings that can be achieved by DSWITCH platform when running Hadoop cluster.

For most of the experiments, we use a typical office PC (DELL OptiPlex 9010, with Intel Core i7-3770, 16GB RAM and 1 Gbit/s Ethernet) as MC. We benchmark the I/O performance of MC and LC using Iometer [5] and IOzone [4].

The disk to be switched in the tests is a Seagate 1.5TB 7200RPM 3.5 inch disk. The disk is powered by a separate power supply. We measure the disk power consumption to be 6.66 watts when active, 4.71 watts when idle and 0.05 watts when spun down. We do not add disk power consumption in all the reported numbers in our evaluation.

7.1 Power Consumption

To understand the power consumption of normal desktops and servers, we evaluate the power consumption of a typical

[1] It supports SATA 3.0 speed at 6Gb/s, and its MTTF is estimated to be more than 42700000 hours using Accelerated Life Test [11].

Computers	Lightweight I/O	Idle	S3	Save
Workstation	218.7	211.2	3.3	97.5%
Game machine	116.3	107.5	2.4	95.9%
OptiPlex 9010	56.0	41.5	1.3	91.9%
Cubietruck	2.8	1.6	-	-

Table 1: Power consumption of different machines (watt). The power consumption of idle state is measured after machines have been idle for 5 minutes. The power consumption of lightweight I/O workload is measured when they are providing 1MB/s sequential data access through Samba. The last column is the power savings of DSWITCH compared with computers' idle power. The power consumption of DSWITCH consists of the power consumptions of Cubietruck in idle, MC in S3 and the SATA multiplexer.

desktop (DELL OptiPlex 9010), a Game machine (Intel i7-4770, 32GB RAM, NVidia GeForce 760) and a powerful workstation (Dell T5500, Dual Xeon X5650, 48GB RAM, NVidia GeForce 660Ti), as shown in Table 1. The machines all run Ubuntu desktop 12.04. As we can see from the table, compared with their respective power consumptions in S3 state, machines awake consume significantly more power even when they are idle or providing lightweight services [25]. Cubietruck, on the other hand, consumes very little power even with I/O workload.

We measure the power consumption of the SATA multiplexer to be around 0.45W. Therefore, DSWITCH working in the low power network attached mode can achieve a power saving of 91.9% to 97.5% (excluding disk power consumption). When working in the high performance directly attached mode, the power overhead of DSWITCH is only 2.05W. It could potentially be reduced to be as little as 0.45W when sleep and wakeup are supported in future revision of Cubietruck.

7.2 Price Comparison

Here we estimate monetary cost of DSWITCH, and verify its cost-efficiency in both home and data center scenarios. SATA multiplexer is estimated as $15 each [10], and a low power computer for home scenario (*e.g.,* CubieBoard, Raspberry Pi) can be less than $33, thus, a DSWITCH is about $48. Assuming a computer is idle for 50% of the time (running 24 hours a day) [31], it only takes about 3.3, 6.5, 17.3 months respectively for the three computers in Table 1 to earn back the cost of DSWITCH [2]. Note that with DSWITCH, there is no need to setup NAS any more.

For data centers, not only does DSWITCH provide better data access performance, the deployment of DSWITCH is also much cheaper than those commercial NAS/SAN solutions. Here we use Cubietruck ($80) as LC for better performance, and one additional 1Gb/s port is $4, so, the price of a DSWITCH becomes $99 accordingly. We compare it with DELL PowerVault MD3260i [8], a SAN storage array, which is already cheaper than other similar commercial products. MD3260i without disks costs $23607.68, thus, the amortized cost for one disk is $393, 3 times higher than DSWITCH. Moreover, if LC allows more disks to be attached or we use wholesale prices, the capital expenditure of deploying DSWITCH will be further reduced.

[2] We use the electricity price in New York, *i.e.,* 19.29 Cents per Kilowatthour [9].

Events	Min	Avg	Max
Enter S3	2.6	2.8	3.2
Network Resumes from S3	4.0	4.9	5.1
Network MC to LC	3.0	3.5	4.0
Network LC to MC	7.0	13.9	22.0
Service MC to LC	9.2	12.7	15.3
Service LC to MC	11.0	17.0	22.8

Table 2: Disruption of network and services (second). Each event is tested for 10 times and the table shows minimum, average and maximum numbers.

7.3 Network and Services Reachability

During switch process, the network and the services are disrupted. We carry out a series of experiments to understand the factor that contribute to the length of the service disruption. The summary of the evaluation is shown in Table 2.

We measure the time for MC to enter S3, which is the time between the request and the power LED's first blink, to be 2.8s on average. We use ICMP ECHO (ping) messages to detect the availability/unavailabilty of the network. When leaving S3, the network of MC needs about 5s to resume after the wake-up event is triggered. The network is disrupted for only 3.5s when switching from MC to LC. Switching from LC to MC disrupts the network a little longer, on average about 13.9s. The reason is because LC has to first reconfigure the network back to its own network identity, this is the time when the network of the unit becomes unreachable. Then it sends magic packet to wake up MC, which needs an additional 5s or so to resume its network.

Lastly, we use Samba as an example to measure end-to-end service disruption. When switching from MC to LC, the time is mainly consumed by detecting and mounting the disk, which takes about 5s. Because the disk has to be spun down and cleanly unmount right before MC enters S3, so LC has to spin up the disk again. Both starting and stopping Samba service takes around 1s. If a service requires longer time to stop or start, the service unreachable time would be correspondingly longer. For switching from LC to MC, the time of the service being unreachable is about 17s, longer than the time for network recovery.

The service interruption is reasonable and acceptable. Home/office scenario usually does not have stringent demand for very short interruption. The switch operation is often executed after users finish using MC, for example, leaving the office. In data center scenario, most services have their own fault tolerance mechanism, such as three replicas, thus, the interruption can be tolerated seamlessly. However, if a service cannot tolerate interruption, the mechanism discussed in § 5 could be applied.

7.4 Disk Performance

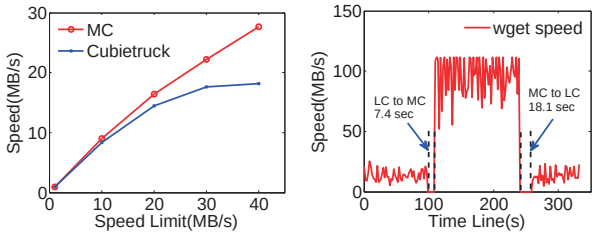
In order to evaluate disk performance, Cubietruck and MC expose the disk on the network using Samba and iSCSI respectively. We also tested local disk read/write performance for comparison using standard C++ library.

For iSCSI test, we installed iSCSI target (*i.e., scsi-target-utils*) on both machines. The iSCSI target runs with *write-cache* off in the configuration since different cache sizes would have different impact on the performance. Then we tested disk performance with various workloads using Iometer and used one worker to generate these workloads. In order to get the real networked disk's performance, we set the test file's size 4 times larger than the system memory (*i.e.,* we use 8GB file for Cubietruck and 64GB file for MC) to avoid the whole test file being cached in the system memory.

The result is shown in Table 3. For sequential read and write, the throughput increases with the increase of request size for MC, while for Cubietruck it seems that it can support about 33 MB/s for sequential read and 16 MB/s for sequential write at most. For 4KB sequential write and 4KB random read/write, Cubietruck shows the same performance with MC. The performance of the two machines does not have much difference especially for random access and small request size.

To deeply understand the performance result, we recorded the CPU utilization of the iSCSI target service during the test. We can see that the CPU utilization of MC is very low. If we consider the resource of all 8 processors, the CPU utilization of the desktop is even less than 3%. On the other hand, though Cubietruck has 2 processors, the CPU is also not fully utilized, at the maximum of 122.7%. For some workloads, it is because of the bottleneck of the disk. While for other workloads, none of disk, network and CPU reaches its own bottleneck (Cubietruck can reach 600 Mbit/s network speed with nearly 100% CPU utilization). We also tried multiple workers generating workloads simultaneously. The performance increases by 1∼5MB/s, however, still below the performance of MC. Thus we suspect it is because of bad task scheduling. Disk, network and CPU are not well pipelined.

For Samba service, we tested the disk performance using IOzone as Samba provides file-level data service. We also set the file size 4 times bigger than the system memory to avoid the whole file being cached in the memory. As shown in Table 3, the performance of sequential read is stable for different request sizes, probably because Samba has read-ahead mechanism regardless of the request size. The write performance of both Cubietruck and MC is better than iSCSI service mainly because Samba returns write request right after the data is written in memory. For sequential write, the throughput of Cubietruck does not increase that fast compared to MC, we think it is because Cubietruck has relatively poor sequential write performance for local disk. Cubietruck performs sequential read as fast as MC while its sequential write is only 38MB/s.



(a) Speed of wget (b) Speed during state transition

Figure 6: Result of wget test.

7.5 Service Performance

7.5.1 Home Scenarios

We tested multiple applications on our DSWITCH prototype, such as video streaming through file sharing, background downloading using *wget*, and BitTorrent. We observe that DSWITCH provides satisfactory services in all these application scenarios. In this section, we report some detailed results on background downloading using *wget*.

In the setup, an Apache Webserver is deployed in the same LAN to provide a fixed upload throughput. We report the *wget* performance of both MC and LC, as well as the behavior during switch operation. As shown in Figure 6(a), the download speed of MC increases almost linearly with the increase of throughput limit at the source, while Cubietruck can only support a maximum download speed of around 18MB/s. We perform switch in the middle of download (no bandwidth limit) and plot the downloading speed of this period as shown in Figure 6(b). The interrupted time for switching from LC to MC is about 7.4s, while switching from MC to LC takes 18.1 seconds.

7.5.2 Database

We also evaluated the performance of TPC-H and TPC-C [24] on our prototype using MySQL database which runs in a virtual machine (KVM, 4 cores, 4 GB memory, virtio enabled). The disk that stores this VM's image is mounted to the host machine (MC) of this VM through three ways: locally, NFS share provided by another MC, and NFS share provided by LC (*i.e.,* Cubietruck).

Workloads	4KSR	4KSW	4KRR	4KRW	1MSR	1MSW	1MRR	1MRW
MC local	117/7.2	105/12.0	0.43/1.0	0.38/1.0	120/6.1	185/12.8	48.9/2.5	36.2/1.5
LC local	113/90.1	37.2/48.5	0.41/2.1	0.41/2.6	110/91.5	38.4/41.2	43.5/38.7	23.2/17.1
MC iSCSI	11.9/22.7	0.46/1.7	0.53/2.0	0.24/1.0	98.6/13.6	40.1/5.5	45.9/7.5	20.4/3.5
LC iSCSI	7.1/88.1	0.46/15.3	0.46/14.2	0.24/9.9	32.5/122.7	15.8/72.0	25.2/94.9	12.1/55.2
MC Samba	44.6/6.2	7.0/20.0	0.44/1.3	2.24/2.6	46.4/6.7	88.5/36.4	29.1/4.9	53.1/21.2
LC Samba	22.5/85.6	5.2/89.3	0.51/7.4	2.19/31.3	22.5/87.0	24.5/90.7	20.0/77.5	29.6/97.3

Table 3: Disk performance of different workloads. local, iSCSI and Samba are local disk read/write, iSCSI service and Samba service respectively. The name of the workloads consists of 3 parts: **4K** or **1M**, **S**equential or **R**andom, **R**ead or **W**rite. There are two numbers separated by '/' in each cell, the left one is throughput (MB/s), the right one is the corresponding CPU utilization compared to the resource of a single processor.

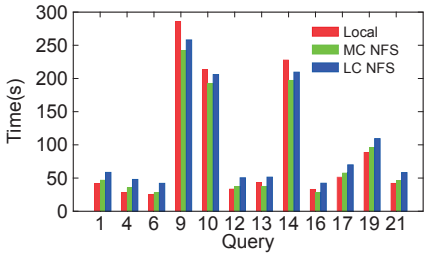


Figure 7: Result of TPC-H benchmark. Scale factor of TPC-H equals 1 in this test.

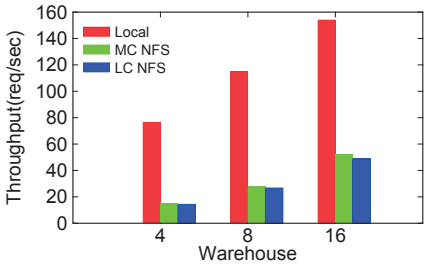


Figure 8: Result of TPC-C benchmark.

For TPC-H, before executing each query, we drop the cache of the VM, host machine and NFS server, and also restart MySQL. We only show several queries in Figure 7, other queries have similar results. Since TPC-H is CPU intensive benchmark, the three modes present similar query latency. For low latency queries, local mode is a little better. While, for long latency queries, MC NFS performs better, probably because NFS client prefetches more data from NFS server during the execution of the queries, increasing cache hit. The query latency of LC NFS is only a little lower than that of MC NFS.

For TPC-C, we use InnoDB as the storage engine, and change the number of warehouses to test the performance. The result is shown in Figure 8. NFS mode performs much worse than Local mode, due to the high network latency it induces. Moreover, MC NFS and LC NFS have similar throughput, since the bottleneck locates in disk I/O and transaction implementation rather than CPU and network.

7.5.3 Hadoop Cluster

We implemented Hadoop on three nodes, one of them is attached with the DSWITCH prototype. When switching this node to LC, its datanode is migrated to another node, running in a VM which mounts the disk exposed by LC. We switch the disk during data read and write. The interrupted interval of datanode heartbeats is 22s when switching from LC to MC, and 41s from MC to LC, both are much lower than the default timeout 10 minutes. We do not show detailed numbers for the sake of space.

Then, we implement Facebook Hadoop workload provided in [6, 22] on a Hadoop cluster with 19 machines. Each machine has 64GB of memory and 2.6GHz AMD Opteron 4180 Processor (12 cores). All these machines are connected by a 1 Gbit/s switch. The trace in our evaluation is generated from the samples of the 2009 Facebook Hadoop cluster using SWIM [6]. We implement the first 3000 jobs in our cluster, which lasts about 14 hours. During its peak load, all the machines are heavily loaded.

The number of submitted jobs in every 5 minutes is shown in Figure 9. We use "oracle" load prediction to determine the number of online nodes, called workload aware scheduling (WAS) as shown in Figure 9. We compare WAS with full gear mode (*i.e.,* all nodes stay online) and half gear mode (*i.e.,* 9 nodes stay online). The half gear mode is implemented by simply stopping 10 nodes' tasktrackers. Here we evaluate two different implementations of WAS: first, the LC mode is simulated by stopping tasktracker, called WAS-ST; second, the LC mode is simulated by stopping tasktracker and limiting the throughput of that node to 30MB/s (*e.g., tc* in linux), called WAS-BD.

We set the full gear mode as the baseline, and show the performance of the other three modes in Figure 10. WAS-ST almost has no performance degradation and meanwhile saves $\frac{68.8(E_M-E_L)}{E_M}\%$ power consumption (*e.g.,* 67.8% if we

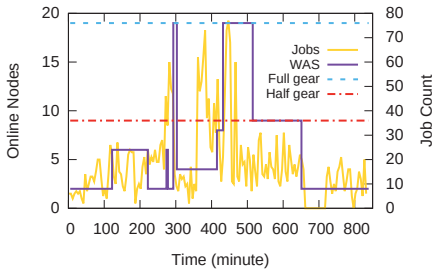


Figure 9: Load of the cluster and different scheduling modes.

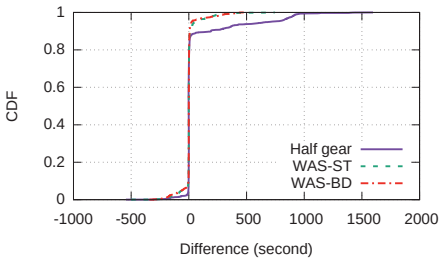


Figure 10: Performance of different scheduling modes. The lines are the CDF of the duration difference of corresponding jobs between the baseline and the other modes, *e.g.,* the duration of the 10th job in WAS-ST minus the duration of the 10th job in baseline.

use $E_M = 211.2$ and $E_L = 2.8$ in Table 1). A small part of jobs show a little higher or lower duration compared to the full gear mode, which can be seen as the random errors of job duration. However, for the half gear mode, about 10% jobs suffer significant performance degradation, because the cluster cannot provide enough computational resource during peak load. WAS-BD on the other hand shows almost the same performance as WAS-ST, which illustrates that less powerful Lc can still provide reasonable performance in such systems.

8. Related Work

Various works have investigated methods to enter computers into sleep state when they are idle without losing their presence on the network. In order to support always-on semantics, remote proxies [31] have been designed. Works such as [13, 17, 23] propose VM migration, which is mainly used in data centers to consolidate idle computers. These works have to wake up the computers for storage access even for very light workload. Some works like [12, 20, 36] propose multi-tiered hardware architecture and each tier has a different power consumption profile. Somniloquy [12] and Turducken [36] target laptops in household and enterprise scenarios, they equip a single computer with a little proxy (*i.e.,* a low-power mini-processor) which maintains the network presence for the computer. Another work [29] moves a step further to build a sharing and consistent file system ZZFS based on Somniloquy. However, the low-power tier still cannot operate the storage of the sleeping computer directly, which is required for normal execution of many applications. This means the computer has to wake up for many trivial tasks. Power-Agile [20] explores the configuration space of heterogeneous devices for great power efficiency.

Many works have been done to deal with power proportionality in data centers. Since workloads in data centers show significant peak-to-trough ratios [38]. Some approaches [14, 40] achieve this goal by increasing replication factor, however, it demands more storage capacity and degrades write performance. Rearranging data layout carefully [14, 38] is a common approach to power off a proportion of servers, but it would generate more write workload because the data that should be written to the offline servers need to be written to the online servers temporarily. On the other hand, since erasure coding has been applied in data centers broadly [27, 34], it is becoming harder to apply these approaches. DSwitch architecture removes the strict constraint for data availability, which might offer more optimization space for these solutions.

There are other works targeting power saving in data centers. e-STAB [28] considers traffic load which is an important factor when designing their dynamic power management. PowerNap [30] minimizes idle power and the states transition time to achieve rapid reaction to workload variation, but it requires nearly all components of the system to support sleep state, which actually has not been widely supported in current servers. FAWN [15] directly replaces commodity server with low-power embedded CPU server for data-intensive workloads. However, workloads in data centers tend to be both CPU and I/O intensive, the approach of FAWN can only support certain I/O intensive workloads.

Some works target power saving of disks. The work in [33] proposes a different data layout policy which aggregates hot data onto a small number of "hot" disks and spins down the disks that store cold data. However, it introduces more latency when accessing cold data since the disks need to be spun up first. Another work in [44] uses multi-speed disks and complicate management schemes (*i.e.,* data layout and migration, disk speed settings) to achieve power proportionality. In this paper we target flexible power proportionality of computers, so the approaches for disks above can actually be co-deployed with ours.

9. Conclusion

We have presented DSwitch, an storage attachment architecture that allows a disk to be either connected to a host computer through SATA interface, or to be directly connected to the network through SAN or NAS protocol. DSwitch combines the benefit of both directly attached storage (DAS) and network attached storage (NAS

and SAN). And the DSWITCH architecture can be widely adopted in home, office and data center environments.

10. Acknowledgments

The work was supported by National Basic Research Program of China ("973") under Grant 2011CB302305, and China NSF under Grant 61232004. We thank the anonymous SoCC reviewers and our shepherd Martin Kersten for comments that helped improve this paper.

References

[1] Advanced configuration and power interface specification. http://www.acpi.info/DOWNLOADS/ACPIspec50.pdf.

[2] CubieBoard. http://cubieboard.org/.

[3] HotSwap! http://mt-naka.com/hotswap/index_enu.htm.

[4] IOzone Filesystem Benchmark. http://www.iozone.org/.

[5] Iometer. http://www.iometer.org/.

[6] Statistical Workload Injector for MapReduce (SWIM). https://github.com/SWIMProjectUCB/SWIM/wiki#overview.

[7] Wake on LAN Technology. http://www.liebsoft.com/pdfs/Wake_On_LAN.pdf.

[8] The Price of DELL PowerVault MD3260i. http://configure.us.dell.com/dellstore/config.aspx?oc=brct32&model_id=powervault-md3260i&c=us&l=en&s=bsd&cs=04.

[9] Average Retail Price of Electricity to Ultimate Customers by End-Use Sector. http://www.eia.gov/electricity/monthly/epm_table_grapher.cfm?t=epmt_5_6_a.

[10] The Price of MAX4986. http://www.maximintegrated.com/en/products/analog/analog-switches-multiplexers/MAX4986.html/tb_tab1.

[11] RELIABILITY REPORT FOR MAX4986ETO+T PLASTIC ENCAPSULATED DEVICES. http://www.maximintegrated.com/reliability/product/MAX4986.pdf.

[12] Y. Agarwal, S. Hodges, R. Chandra, J. Scott, P. Bahl, and R. Gupta. Somniloquy: Augmenting Network Interfaces to Reduce PC Energy Usage. In *Proc. USENIX NSDI*, 2009.

[13] Y. Agarwal, S. Savage, and R. Gupta. Sleepserver: A Software-Only Approach for Reducing the Energy Consumption of PCs within Enterprise Environments. In *Proc. USENIX ATC*, 2010.

[14] H. Amur, J. Cipar, V. Gupta, G. R. Ganger, M. A. Kozuch, and K. Schwan. Robust and Flexible Power-Proportional Storage. In *Proc. SoCC*, 2010.

[15] D. G. Andersen, J. Franklin, M. Kaminsky, A. Phanishayee, L. Tan, and V. Vasudevan. FAWN: A Fast Array of Wimpy Nodes. In *Proc. SOSP*, 2009.

[16] L. A. Barroso and U. Holzle. The Case for Energy-Proportional Computing. *Computer*, 40(12):33–37, 2007.

[17] N. Bila, E. de Lara, K. Joshi, H. A. Lagar-Cavilla, M. Hiltunen, and M. Satyanarayanan. Jettison: Efficient Idle Desktop Consolidation with Partial VM Migration. In *Proc. EuroSys*, 2012.

[18] N. Bobroff, A. Kochut, and K. Beaty. Dynamic Placement of Virtual Machines for Managing SLA Violations. In *Proc. 10th IFIP/IEEE International Symposium on Integrated Network Management*, 2007.

[19] P. Bohrer, E. N. Elnozahy, T. Keller, M. Kistler, C. Lefurgy, C. McDowell, and R. Rajamony. The Case for Power Management in Web Servers. In *Power aware computing*, pages 261–289. Springer, 2002.

[20] G. Challen and M. Hempstead. The Case for Power-Agile Computing. In *Proc. USENIX HotOS*, 2011.

[21] G. Chen, W. He, J. Liu, S. Nath, L. Rigas, L. Xiao, and F. Zhao. Energy-Aware Server Provisioning and Load Dispatching for Connection-Intensive Internet Services. In *Proc. USENIX NSDI*, 2008.

[22] Y. Chen, A. Ganapathi, R. Griffith, and R. Katz. The Case for Evaluating MapReduce Performance Using Workload Suites. In *Proc. 19th International Symposium on Modeling, Analysis & Simulation of Computer and Telecommunication Systems (MASCOTS)*, 2011.

[23] T. Das, P. Padala, V. N. Padmanabhan, R. Ramjee, and K. G. Shin. LiteGreen: Saving Energy in Networked Desktops Using Virtualization. In *Proc. USENIX ATC*, 2010.

[24] D. E. Difallah, A. Pavlo, C. Curino, and P. Cudre-Mauroux. OLTP-Bench: An Extensible Testbed for Benchmarking Relational Databases. In *Proc. VLDB*, 2013.

[25] D. Economou, S. Rivoire, C. Kozyrakis, and P. Ranganathan. Full-System Power Analysis and Modeling for Server Environments. In *Proc. ISCA*, 2006.

[26] U. Hoelzle and L. A. Barroso. *The Datacenter as a Computer: An Introduction to the Design of Warehouse-Scale Machines*. Morgan and Claypool Publishers, 1st edition, 2009. ISBN 159829556X, 9781598295566.

[27] C. Huang, H. Simitci, Y. Xu, A. Ogus, B. Calder, P. Gopalan, J. Li, S. Yekhanin, et al. Erasure Coding in Windows Azure Storage. In *Proc. USENIX ATC*, 2012.

[28] D. Kliazovich, S. T. Arzo, F. Granelli, P. Bouvry, and S. U. Khan. e-STAB: Energy-Efficient Scheduling for Cloud Computing Applications with Traffic Load Balancing. In *Proc. IEEE GreenCom*, 2013.

[29] M. L. Mazurek, E. Thereska, D. Gunawardena, R. Harper, and J. Scott. ZZFS: A Hybrid Device and Cloud File System for Spontaneous Users. In *Proc. USENIX FAST*, 2012.

[30] D. Meisner, B. T. Gold, and T. F. Wenisch. PowerNap: Eliminating Server Idle Power. In *Proc. ASPLOS*, 2009.

[31] S. Nedevschi, J. Chandrashekar, J. Liu, B. Nordman, S. Ratnasamy, and N. Taft. Skilled in the Art of Being Idle: Reducing Energy Waste in Networked Systems. In *Proc. USENIX NSDI*, 2009.

[32] B. Nordman. Networks, Energy, and Energy Efficiency. In *Proc. Cisco Green Research Symposium*, 2008.

[33] E. Pinheiro and R. Bianchini. Energy Conservation Techniques for Disk Array-Based Servers. In *Proc. the 18th annual international conference on Supercomputing (ICS)*, 2004.

[34] M. Sathiamoorthy, M. Asteris, D. Papailiopoulos, A. G. Dimakis, R. Vadali, S. Chen, and D. Borthakur. Xoring Elephants: Novel Erasure Codes for Big Data. In *Proc. VLDB*, 2013.

[35] A. Singh, M. Korupolu, and D. Mohapatra. Server-Storage Virtualization: Integration and Load Balancing in Data Centers. In *Proc. ACM/IEEE conference on Supercomputing*, 2008.

[36] J. Sorber, N. Banerjee, M. D. Corner, and S. Rollins. Turducken: Hierarchical Power Management for Mobile Devices. In *Proc. MobiSys*, 2005.

[37] A. S. Szalay, G. C. Bell, H. H. Huang, A. Terzis, and A. White. Low-Power Amdahl-Balanced Blades for Data Intensive Computing. *ACM SIGOPS Operating Systems Review*, 44(1):71–75, 2010.

[38] E. Thereska, A. Donnelly, and D. Narayanan. Sierra: Practical Power-Proportionality for Data Center Storage. In *Proc. EuroSys*, 2011.

[39] N. Tolia, Z. Wang, M. Marwah, C. Bash, P. Ranganathan, and X. Zhu. Delivering Energy Proportionality with Non Energy-Proportional Systems-Optimizing the Ensemble. In *Proc. HotPower*, 2008.

[40] C. Weddle, M. Oldham, J. Qian, A.-I. A. Wang, P. Reiher, and G. Kuenning. PARAID: A Gear-Shifting Power-Aware RAID. *ACM Transactions on Storage (TOS)*, 3(3):13, 2007.

[41] T. Wood, P. J. Shenoy, A. Venkataramani, and M. S. Yousif. Black-box and Gray-box Strategies for Virtual Machine Migration. In *Proc. USENIX NSDI*, 2007.

[42] L. Xu, J. Cipar, E. Krevat, A. Tumanov, N. Gupta, M. A. Kozuch, and G. R. Ganger. SpringFS: Bridging Agility and Performance in Elastic Distributed Storage. In *Proc. USENIX FAST*, 2014.

[43] Q. Zhang, M. F. Zhani, S. Zhang, Q. Zhu, R. Boutaba, and J. L. Hellerstein. Dynamic Energy-Aware Capacity Provisioning for Cloud Computing Environments. In *Proc. the 9th international conference on Autonomic computing (ICAC)*, 2012.

[44] Q. Zhu, Z. Chen, L. Tan, Y. Zhou, K. Keeton, and J. Wilkes. Hibernator: Helping Disk Arrays Sleep Through the Winter. In *Proc. SOSP*, 2005.