Computer Networks 53 (2009) 2939–2965

Contents lists available at ScienceDirect

Computer Networks

journal homepage: www.elsevier.com/locate/comnet

Data center evolution
A tutorial on state of the art, issues, and challenges

Krishna Kant

Intel Corporation, Hillsboro, Oregon, USA

A R T I C L E I N F O

Keywords:
Data center
Virtualization
InfiniBand
Ethernet
Solid state storage
Power management

A B S T R A C T

Data centers form a key part of the infrastructure upon which a variety of information technology services are built. As data centers continue to grow in size and complexity, it is desirable to understand aspects of their design that are worthy of carrying forward, as well as existing or upcoming shortcomings and challenges that would have to be addressed. We envision the data center evolving from owned physical entities to potentially outsourced, virtualized and geographically distributed infrastructures that still attempt to provide the same level of control and isolation that owned infrastructures do. We define a layered model for such data centers and provide a detailed treatment of state of the art and emerging challenges in storage, networking, management and power/thermal aspects.

1. Introduction

Data centers form the backbone of a wide variety of services offered via the Internet including Web-hosting, e-commerce, social networking, and a variety of more general services such as software as a service (SAAS), platform as a service (PAAS), and grid/cloud computing. Some examples of these generic service platforms are Microsoft's Azure platform, Google App engine, Amazon's EC2 platform and Sun's Grid Engine. Virtualization is the key to providing many of these services and is being increasingly used within data centers to achieve better server utilization and more flexible resource allocation. However, virtualization also makes many aspects of data center management more challenging.

As the complexity, variety, and penetration of such services grows, data centers will continue to grow and proliferate. Several forces are shaping the data center landscape and we expect future data centers to be lot more than simply bigger versions of those existing today. These emerging trends – more fully discussed in Section 3 – are expected to turn data centers into distributed, virtualized, multi-layered infrastructures that pose a variety of difficult challenges.

In this paper, we provide a tutorial coverage of a variety of emerging issues in designing and managing large virtualized data centers. In particular, we consider a layered model of virtualized data centers and discuss storage, networking, management, and power/thermal issues for such a model. Because of the vastness of the space, we shall avoid detailed treatment of certain well researched issues. In particular, we do not delve into the intricacies of virtualization techniques, virtual machine migration and scheduling in virtualized environments.

The organization of the paper is as follows. Section 2 discusses the organization of a data center and points out several challenging areas in data center management. Section 3 discusses emerging trends in data centers and new issues posed by them. Subsequent sections then discuss specific issues in detail including storage, networking, management and power/thermal issues. Finally, Section 8 summarizes the discussion.

E-mail address: krishna.kant@intel.com

1389-1286/$ - see front matter © 2009 Published by Elsevier B.V.
doi:10.1016/j.comnet.2009.10.004

2. Data center organization and issues

2.1. Rack-level physical organization

A data center is generally organized in rows of "racks" where each rack contains modular assets such as servers, switches, storage "bricks", or specialized appliances as shown in Fig. 1. A standard rack is 78 in. high, 23–25 in. wide and 26–30 in. deep. Typically, each rack takes a number of modular "rack mount" assets inserted horizontally into the racks. The asset thickness is measured using an unit called "U", which is 45 mm (or approximately 1.8 in.). An overwhelming majority of servers are single or dual socket processors and can fit the 1U size, but larger ones (e.g., 4-socket multiprocessors) may require 2U or larger sizes. A standard rack can take a total of 42 1U assets when completely filled. The sophistication of the rack itself may vary greatly – in the simplest case, it is nothing more than a metal enclosure. Additional features may include rack power distribution, built-in KVM (keyboard–video–mouse) switch, rack-level air or liquid cooling, and perhaps even a rack-level management unit.

For greater compactness and functionality, servers can be housed in a self-contained chassis which itself slides into the rack. With 13 in. high chassis, six chassis can fit into a single rack. A chassis comes complete with its own power supply, fans, backplane interconnect, and management infrastructure. The chassis provides standard size slots where one could insert modular assets (usually known as *blades*). A single chassis can hold up to 16 1U servers, thereby providing a theoretical rack capacity of 96 modular assets.

The substantial increase in server density achievable by using the blade form factor results in corresponding increase in per-rack power consumption which, in turn, can seriously tax the power delivery infrastructure. In particular, many older data centers are designed with about 7 KW per-rack power rating, whereas racks loaded with blade servers could approach 21 KW. There is a similar issue with respect to thermal density – the cooling infrastructure may be unable to handle the offered thermal load. The net result is that it may be impossible to load the racks to their capacity. For some applications, a fully loaded rack may not offer the required peak network or storage bandwidth (BW) either, thereby requiring careful management of resources to stay within the BW limits.

Fig. 1. Physical organization of a data center.

2.2. Storage and networking infrastructure

Storage in data centers may be provided in multiple ways. Often the high performance storage is housed in special "storage towers" that allow transparent remote access to the storage irrespective of the number and types of physical storage devices used. Storage may also be provided in smaller "storage bricks" located in rack or chassis slots or directly integrated with the servers. In all cases, an efficient network access to the storage is crucial.

A data center typically requires four types of network accesses, and could potentially use four different types of physical networks. The client–server network provides external access into the data center, and necessarily uses a commodity technology such as the wired Ethernet or wireless LAN. Server-to-server network provides high-speed communication between servers and may use Ethernet, InfiniBand (IBA) or other technologies. The storage access has traditionally been provided by Fiber Channel but could also use Ethernet or InfiniBand. Finally, the network used for management is also typically Ethernet but may either use separate cabling or exist as a "sideband" on the mainstream network.

Both mainstream and storage networks typically follow identical configuration. For blade servers mounted on a chassis, the chassis provides a switch through which all the servers in the chassis connect to outside servers. The switches are duplexed for reliability and may be arranged for load sharing when both switches are working. In order to keep the network manageable, the overall topology is basically a tree with full connectivity at the root level. For example, each chassis level (or level 1) switch has an *uplink* leading to the level 2 switch, so that communication between two servers in different chassis must go through at least three switches. Depending on the size of the data center, the multiple level 2 switches may be either connected into a full mesh, or go through one or more level 3 switches. The biggest issue with such a structure is potential bandwidth inadequacy at higher levels. Generally, uplinks are designed for a specific *oversubscription ratio* since providing a full bisection bandwidth is usually not feasible. For example, 20 servers, each with a 1 GB/s Ethernet may share a single 10 GB/s Ethernet uplink for a oversubscription ratio of 2.0. This may be troublesome if the workload mapping is such that there is substantial non-local communication. Since storage is traditionally provided in a separate storage tower, all storage traffic usually crosses the chassis uplink on the storage network. As data centers grow in size, a more scalable network architecture becomes necessary.

2.3. Management infrastructure

Each server usually carries a management controller called the BMC (baseboard management controller). The management network terminates at the BMC of each server. When the management network is implemented as a "sideband" network, no additional switches are required for it; otherwise, a management switch is required in each chassis/rack to support external communication. The basic functions of the BMC include monitoring of various hardware sensors, managing various hardware and software alerts, booting up and shutting down the server, maintaining configuration data of various devices and drivers, and providing remote management capabilities. Each chassis or rack may itself sport its own higher level management controller which communicates with the lower level controller.

Configuration management is a rather generic term and can refer to management of parameter settings of a variety of objects that are of interest in effectively utilizing the computer system infrastructure from individual devices up to complex services running on large networked clusters. Some of this management clearly belongs to the baseboard management controller (BMC) or corresponding higher level management chain. This is often known as *out-of-band* (OOB) management since it is done without involvement of main CPU or the OS. Other activities may be more appropriate for *in-band* management and may be done by the main CPU in hardware, in OS, or in the middleware. The higher level management may run on separate systems that have both in-band and OOB interfaces. On a server, the most critical OOB functions belong to the preboot phase and in monitoring of server health while the OS is running. On other assets such as switches, routers, and storage bricks the management is necessarily OOB.

2.4. Electrical and cooling infrastructure

Even medium-sized data centers can sport peak power consumption of several megawatts or more. For such power loads, it becomes necessary to supply power using high voltage lines (e.g., 33 KV, 3 phase) and step it down on premises to the 280–480 V (3 phase) range for routing through the uninterrupted power supply (UPS). The UPS unit needs to convert AC to DC to charge its batteries and then convert DC to AC on the output end. Since the UPS unit sits directly in the power path, it can continue to supply output power uninterrupted in case of input power loss. The output of UPS (usually 240/120 V, single phase) is routed to the power distribution unit (PDU) which, in turn, supplies power to individual rack-mounted servers or blade chassis. Next the power is stepped down, converted from AC to DC, and partially regulated in order to yield the typical ±12 and ±5 V outputs with the desired current ratings (20–100 A). These voltages are delivered to the motherboard where the voltage regulators (VRs) must convert them to as many voltage rails as the server design demands. For example, in an IBM blade server, the supported voltage rails include 5–6 V (3.3 V down to 1.1 V), in addition to the 12 V and 5 V rails.

Each one of these power conversion/distribution stages results in power loss, with some stages showing efficiencies in 85–95% range or worse. It is thus not surprising that the cumulative power efficiency by the time we get down to voltage rails on the motherboard is only 50% or less (excluding cooling, lighting, and other auxiliary power uses). Thus there is a significant scope for gaining power efficiencies by a better design of power distribution and conversion infrastructure.

The cooling infrastructure in a data center can be quite elaborate and expensive involving building level air-conditioning units requiring large chiller plants, fans and air recirculation systems. Evolving cooling technologies tend to emphasize more localized cooling or try to simplify cooling infrastructure. The server racks are generally placed on a raised plenum and arranged in alternately back-facing and front-facing aisles as shown in Fig. 2. Cold air is forced up in the front facing aisles and the server or chassis fans draw the cold air through the server to the back. The hot air on the back then rises and is directed (sometimes by using some deflectors) towards the chiller plant for cooling and recirculation. This basic setup is not expensive but can also create hot spots either due to uneven cooling or the mixing of hot and cold air.

2.5. Major data center issues

Data center applications increasingly involve access to massive data sets, real-time data mining, and streaming media delivery that place heavy demands on the storage

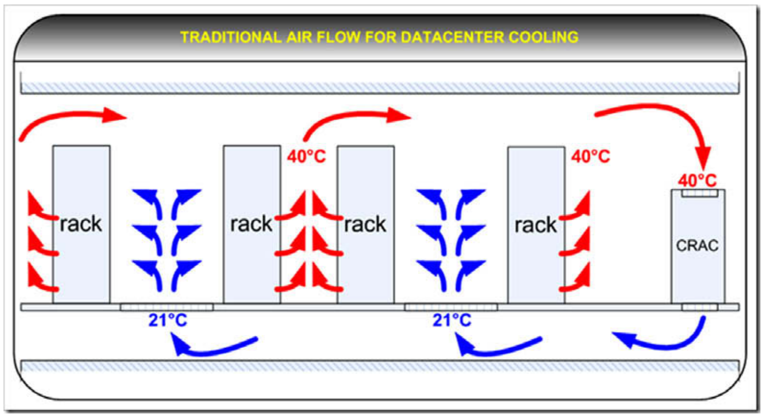

Fig. 2. Cooling in a data center.

infrastructure. Efficient access to large amounts of storage necessitates not only high performance file systems but also high performance storage technologies such as solid-state storage (SSD) media. These issues are discussed in Section 5. Streaming large amounts of data (from disks or SSDs) also requires high-speed, low-latency networks. In clustered applications, the inter-process communication (IPC) often involves rather small messages but with very low-latency requirements. These applications may also use remote main memories as ‘‘network caches” of data and thus tax the networking capabilities. It is much cheaper to carry all types of data – client–server, IPC, storage and perhaps management – on the same physical fabric such as Ethernet. However, doing so requires sophisticated QoS capabilities that are not necessarily available in existing protocols. These aspects are discussed in Section 4.

Configuration management is a vital component for the smooth operation of data centers but has not received much attention in literature. Configuration management is required at multiple levels, ranging from servers to server enclosures to the entire data center. Virtualized environments introduce issues of configuration management at a logical – rather than physical – level as well. As the complexity of servers, operating environments, and applications increases, effective real-time management of large heterogeneous data centers becomes quite complex. These challenges and some approaches are discussed in Section 6.

The increasing size of data centers not only results in high utility costs [1] but also leads to significant challenges in power and thermal management [82]. It is estimated that the total data center energy consumption as a percentage of total US energy consumption doubled between 2000 and 2007 and is set to double yet again by 2012. The high utility costs and environmental impact of such an increase are reasons enough to address power consumption. Additionally, high power consumption also results in unsustainable current, power, and thermal densities, and inefficient usage of data center space. Dealing with power/thermal issues effectively requires power, cooling and thermal control techniques at multiple levels (e.g., device, system, enclosure, etc.) and across multiple domains (e.g., hardware, OS and systems management). In many cases, power/thermal management impacts performance and thus requires a combined treatment of power and performance. These issues are discussed in Section 7.

As data centers increase in size and criticality, they become increasingly attractive targets of attack since an isolated vulnerability can be exploited to impact a large number of customers and/or large amounts of sensitive data [14]. Thus a fundamental security challenge for data centers is to find workable mechanisms that can reduce this growth of vulnerability with size. Basically, the security must be implemented so that no single compromise can provide access to a large number of machines or large amount of data. Another important issue is that in a virtualized outsourced environment, it is no longer possible to speak of ‘‘inside” and ‘‘outside” of data center – the intruders could well be those sharing the same physical infrastructure for their business purposes. Finally, the basic virtualization techniques themselves enhance vulnerabilities since the flexibility provided by virtualization can be easily exploited for disruption and denial of service. For example, any vulnerability in mapping VM level attributes to the physical system can be exploited to sabotage the entire system. Due to limited space, we do not, however, delve into security issues in this paper.

3. Future directions in data center evolution

Traditional data centers have evolved as large computational facilities solely owned and operated by a single entity – commercial or otherwise. However, the forces in play are resulting in data centers moving towards much more complex ownership scenarios. For example, just as virtualization allows consolidation and cost savings within a data center, virtualization across data centers could allow a much higher level of aggregation. This notion leads to the possibility of ‘‘out-sourced” data centers that allows an organization to run a large data center without having to own the physical infrastructure. Cloud computing, in fact, provides exactly such a capability except that in cloud computing the resources are generally obtained dynamically for short periods and underlying management of these resources is entirely hidden from the user. Subscribers of virtual data centers would typically want longer-term arrangements and much more control over the infrastructure given to them. There is a move afoot to provide *Enterprise Cloud* facilities whose goals are similar to those discussed here [2]. The distributed virtualized data center model discussed here is similar to the one introduced in [78].

In the following we present a 4-layer conceptual model of future data centers shown in Fig. 3 that subsumes a wide range of emergent data center implementations. In this depiction, rectangles refer to software layers and ellipses refer to the resulting abstractions.

The bottom layer in this conceptual model is the *Physical Infrastructure Layer* (PIL) that manages the physical infrastructure (often known as ‘‘server farm”) installed in a given location. Because of the increasing cost of the power consumed, space occupied, and management personnel required, server farms are already being located closer to sources of cheap electricity, water, land, and manpower. These locations are by their nature geographically removed from areas of heavy service demand, and thus the developments in ultra high-speed networking over long distances are essential enablers of such remotely located server farms. In addition to the management of physical computing hardware, the PIL can allow for larger-scale consolidation by providing capabilities to carve out well-isolated sections of the server farm (or ‘‘server patches”) and assign them to different ‘‘customers.” In this case, the PIL will be responsible for management of boundaries around the server patch in terms of security, traffic firewalling, and reserving access bandwidth. For example, set up and management of virtual LANs will be done by PIL.

The next layer is the *Virtual Infrastructure Layer* (VIL) which exploits the virtualization capabilities available in individual servers, network and storage elements to support the notion of a *virtual cluster*, i.e., a set of virtual or real nodes along with QoS controlled paths to satisfy their

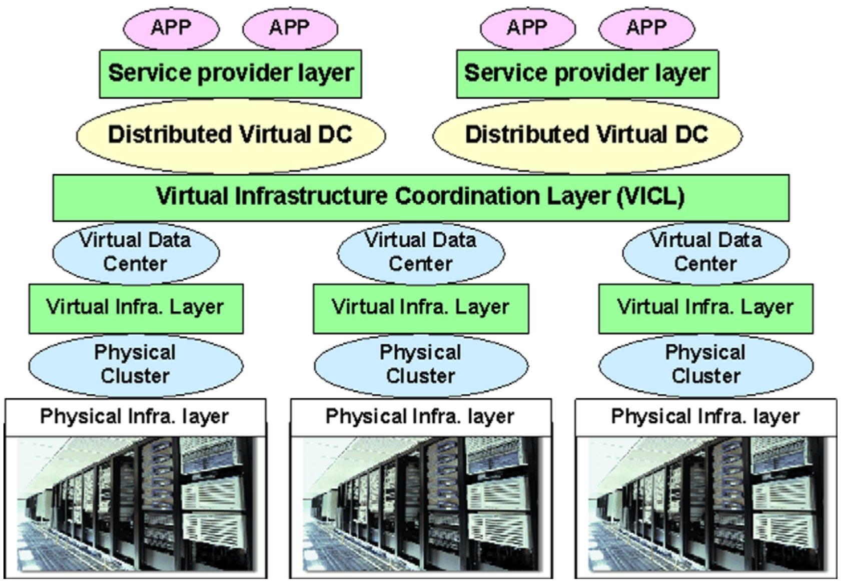

Fig. 3. Logical organization of future data centers.

communication needs. In many cases, the VIL will be internal to an organization who has leased an entire physical server patch to run its business. However, it is also conceivable that VIL services are actually under the control of infrastructure provider that effectively presents a *virtual server patch* abstraction to its customers. This is similar to cloud computing, except that the subscriber to a virtual server patch would expect explicit SLAs in terms of computational, storage and networking infrastructure allocated to it and would need enough visibility to provide its own next level management required for running multiple services or applications.

The third layer in our model is the *Virtual Infrastructure Coordination Layer* (VICL) whose purpose is to tie up virtual server patches across multiple physical server farms in order to create a geographically distributed virtualized data center (DVDC). This layer must define and manage virtual pipes between various virtual data centers. This layer would also be responsible for cross-geographic location application deployment, replication and migration whenever that makes sense. Depending on its capabilities, VICL could be exploited for other purposes as well, such as reducing energy costs by spreading load across time-zones and utility rates, providing disaster or large scale failure tolerance, and even enabling truly large-scale distributed computations.

Finally, the *Service Provider Layer* (SPL) is responsible for managing and running applications on the DVDC constructed by the VICL. The SPL would require substantial visibility into the physical configuration, performance, latency, availability and other aspects of the DVDC so that it can manage the applications effectively. It is expected that SPL will be owned by the customer directly.

The model in Fig. 3 subsumes everything from a non-virtualized, single location data center entirely owned by a single organization all the way up to a geographically distributed, fully virtualized data center where each layer possibly has a separate owner. The latter extreme provides a number of advantages in terms of consolidation, agility, and flexibility, but it also poses a number of difficult challenges in terms of security, SLA definition and enforcement, efficiency and issues of layer separation. For this reason, real data centers are likely to be limited instances of this general model.

In subsequent sections, we shall address the needs of such DVDC's when relevant, although many of the issues apply to traditional data centers as well.

4. Data center networking

4.1. Networking infrastructure in data centers

The increasing complexity and sophistication of data center applications demands new features in the data center network. For clustered applications, servers often need to exchange inter-process communication (IPC) messages for synchronization and data exchange, and such messages may require very low-latency in order to reduce process stalls. Direct data exchange between servers may also be motivated by low access latency to data residing in the memory of another server as opposed to retrieving it from the local secondary storage [18]. Furthermore, mixing of different types of data on the same networking fabric may necessitate QoS mechanisms for performance isolation. These requirements have led to considerable activity in the design and use of low-latency specialized data center fabrics such as PCI-Express based backplane interconnects, InfiniBand (IBA) [37], data center Ethernet [40,7], and lightweight transport protocols implemented directly over the Ethernet layer [5]. We shall survey some of these developments in subsequent sections before examining networking challenges in data centers.

4.2. Overview of data center fabrics

In this section, we provide a brief overview of two major network fabrics in the data center namely, Ethernet and InfiniBand (IBA). Although Fiber channel can be used as a general networking fabric as well, we do not discuss it here because of its strong storage association. Also, although the term "Ethernet" relates only to the MAC layer, in practice, TCP/UDP over IP over Ethernet is the more appropriate network stack to compare against InfiniBand or Fiber-channel which specify their own network and transport layers. For this reason, we shall speak of "Ethernet-stack" instead of just the Ethernet.

4.2.1. InfiniBand

InfiniBand architecture (IBA) was defined in late 1990s specifically for inter-system IO in the data center and uses the communication link semantics rather than the traditional memory read/write semantics [37]. IBA provides a complete fabric starting with its unique cabling/connectors, physical layer, link, network and transport layers. This incompatibility with Ethernet at the very basic cabling/connector level makes IBA (and other data center fabrics such as Myrinet) difficult to introduce in an existing data center. However, the available bridging products make it possible to mix IBA and Ethernet based clusters in the same data center.

IBA links use bit-serial, differential signaling technology which can scale up to much higher data rates than the traditional parallel link technologies. Traditional parallel transmission technologies (with one wire per bit) suffer from numerous problems such as severe cross-talk and skew between the bit timings, especially as the link speeds increase. The differential signaling, in contrast, uses two physical wires for each bit direction and the difference between the two signals is used by the receiver. Each such pair of wires is called a *lane*, and higher data throughput can be achieved by using multiple lanes, each of which can independently deliver a data frame. As the link speeds move into multi GB/s range, differential bit-serial technology is being adopted almost universally for all types of links. It has also led to the idea of *re-purposable PHY*, wherein the same PHY layer can be configured to provide the desired link type (e.g., IBA, PCI-Express, Fiber Channel, Ethernet, etc.).

The generation 1 (or GEN1) bit-serial links run at 2.5 GHz but use a 8–10 byte encoding for robustness which effectively delivers 2 GB/s speed in each direction. GEN2 links double this rate to 4 GB/s. The GEN3 technology does not use 8-10 byte encoding and can provide 10 GB/s speed over fiber. Implementing 10 GB/s on copper cables is very challenging, at least for distances of more than a few meters. The IBA currently defines three lane widths – 1X, 4X, and 12X. Thus, IBA can provide bandwidths up to 120 GB/s over fiber using GEN3 technology.

In IBA, a layer-2 network (called *subnet*) can be built using IBA switches which can route messages based on explicitly configured routing tables. A switch can have up to 256 ports; therefore, most subnets use a single switch. For transmission, application messages are broken up into "packets" or message transfer units (MTUs) with maximum size settable to 256 bytes, 1 KB, 2 KB, or 4 KB. In systems with mixed MTUs, subnet management provides endpoints with the Path MTU appropriate to reach a given destination. IBA switches may also optionally support multicast which allows for message replication at the switches.

There are two important attributes for a transport protocol: (a) connectionless (datagram) vs. connection oriented transfer, and (b) reliable vs. unreliable transmission. In the Ethernet context, only the reliable connection oriented service (TCP) are unreliable datagram service (UDP) are supported. IBA supports all four combinations, known respectively as Reliable Connection (RC), Reliable Datagram (RD), Unreliable Connection (UC), and Unreliable Datagram (UD). All of these are provided in HW (e.g., unlike TCP/UDP) and operate entirely in user mode which eliminates unnecessary context switches or data copies.

IBA implements the "virtual interface" (VI) – an abstract communication architecture defined in terms of a per-connection send–receive queue pair (QP) in each direction and a completion queue that holds the operation completion notification. The completion queue may be exploited for handling completions in a flexible manner (e.g., polling, or interrupts with or without batching). The virtual interface is intended to handle much of the IO operation in user mode and thereby avoid the OS kernel bottleneck. In order to facilitate orderly usage of the same physical device (e.g., NIC) by multiple processes, a VI capable device needs to support "doorbell", by which each process can post the descriptor of a new send/receive operation to the device. The device also needs to support virtual-to-physical (VtP) address translation in HW so that OS intervention is avoided. Some OS involvement is still required: such as for registration of buffers, setup of VtP translation tables, and interrupt handling. Virtual interface adapter (VIA) is a well known networking standard that implements these ideas [13].

VI is a key technology in light-weight user-mode IO and forms the basis for implementing *Remote Direct Memory Access* (RDMA) capability that allows a chunk of data to be moved directly from the source application buffer to the destination application's buffer. Such transfers can occur with very low-latency since they do not involve any OS intervention or copies. Enabling the direct transfer does require a setup phase where the two ends communicate in order to register and pin the buffers on either side and do the appropriate access control checks. RDMA is appropriate for sustained communication between two applications and for transfers of large amounts of data since the setup overhead gets amortized over a large number of operations [6]. RDMA has been exploited for high performance implementations of a variety of functionalities such as virtual machine migration [19], implementation of MPI (message passing interface) [28], and network attached storage [33].

One interesting feature of IBA link layer is the notion of "virtual lanes" (VLs).[1] A maximum of 15 virtual lanes (VL0–VL14) can be specified for carrying normal traffic over an IBA link (VL15 is reserved for management traffic). A VL can be

[1] A virtual lane has nothing to do with the physical "lane" concept discussed above in the context of bit-serial links.

designated as either high priority or low priority and all VLs belonging to one priority level can be further differentiated by a weighted round-robin scheduling among them. Messages are actually marked with a more abstract concept called Service Level (SL), and SLto VL mapping table is used at each switch to decide how to carry this message. This allows messages to pass through multiple switches, each with a different number of VL's supported.

IBA uses a credit based flow control for congestion management on a per virtual lane basis in order to avoid packet losses [3,36]. A VL receiver periodically grants "credit" to the sender for sending more data. As the congestion develops, the sender will have fewer credits and will be forced to slow down. However, such a mechanism suffers from two problems: (a) it is basically a "back-pressure" mechanism and could take a long time to squeeze flows that transit a large number of hops and (b) If multiple flows use the same virtual lane, the flows that are not responsible for the congestion could also get shut-down unnecessarily. To address these issues, IBA supports a Forward Explicit Congestion Notification (FECN) mechanism coupled with endpoint rate control.

FECN tags the packets as they experience congestion on the way to destination and this information is returned back in the acknowledgement packets to the source. Up to 16 congestion levels are supported by IBA. A node receiving congestion notification can tell if it is the source or victim of congestion by checking whether its credits have already been exhausted. A congestion source controls the traffic injection rate successively based on received congestion indications. A subsequent rate increase is based on a time-out that is relative to the latest congestion notification.

IBA also supports Automatic Path Migration (APM) which allows a queue pair (QP) to be associated with two independent paths to the destination. Initially, the first path is used, but a changeover is effected in case of failure or significant errors. A changeback to the original path can be effected under software control.

4.2.2. Ethernet stack

Since the Ethernet stack is quite well known, we only mention some of its salient points relative to the data center environment. Let us start with the MAC layer (i.e., Ethernet per se). Ethernet layer is crucial in data centers since a typical data center sports far more (layer-2) switches than (layer-3) routers. This is a result of much lower cost, latency and configuration simplicity of a layer-2 switch as compared to a router. However, this immediately implies that the things that IP layer can do reasonably well (e.g., routing, QoS, filtering, and security) are not well supported in a data center. Moreover, if we were to simply implement all these mechanisms in layer-2 directly, switches would become as complex, slow and hard to configure as the routers.

One unique capability introduced in the Ethernet in IEEE 802.1q standard is the notion of virtual LAN or VLAN. The VLAN mechanism allows the traffic to be tagged with a 12-bit VLAN id. In a simple static assignment case, VLAN id's are statically mapped to switch ports. This allows the VLANs to provide a strong isolation in that the traffic belonging to a VLAN cannot be directed to ports that are not assigned to that VLAN. A dynamic assignment scheme also exists which can map a VLAN to a unique set of ports depending on source MAC address or other attributes of the traffic.

Given a set of layer-2 endpoints (e.g., servers or routers) connected via a network of switches, a layer-2 routing mechanism is essential to provide a low-latency delivery of Ethernet frames without any loops or complex configuration. The original design of layer-2 routing focused primarily on avoiding routing loops by defining a single spanning tree to cover all endpoints and switches. This spanning tree protocol (STP) (described in IEEE 802.1D standard) disables all links that are not a part of the spanning tree and hence their available bandwidth is wasted. Also, the tree structure results in a very uneven traffic distribution over the used links. Several enhancements have been made to 802.1D to address these issues including (a) Per VLAN spanning tree so that it is possible to use a different subset of links for each VLANs and thereby spread out the traffic, (b) Rapid STP (RSTP) that quickly detects failed links and reconfigures the spanning tree to minimize dropped frames, and (c) Multiple STP (MSTP) which uses several "regional" trees connected via a higher central spanning tree (CST). Other ways of using multiple trees include (a) each switch port acting as the spanning tree root for the traffic incoming at that port, and (b) directing traffic from the same source among multiple trees according to some criteria [20]. In spite of these mechanisms, balancing traffic among various links could still be challenging.

On the QoS front, the Ethernet mechanisms started out as quite primitive, but have been enhanced subsequently. In particular, the VLAN mechanism also includes a 3-bit CoS (*class of service*) field in the extended Ethernet header for differentiating VLAN flows. This is exploited by the data center Ethernet [40] for differentiating between different types of traffic (e.g., storage vs. inter-process communication vs. client–server). The IEEE task force on Ethernet congestion management, known as 802.1Qau (www.ieee802.org/1/pages/802.1au.html), is currently examining ways of improving congestion notification and management [7]. The main objectives of this effort are to enable switches to mark packets and allow endpoint layer-2 to do 802.1x type link flow control at the level of individual CoS classes. (The default Ethernet link flow control happens at the level of entire link and thus is not very useful when multiple traffic types are involved.)

Although IP layer provides a rich set of mechanisms for QoS control, it adds significant additional latency both at routers and at endpoints. Similarly, the traditional TCP sockets interface can incur large latencies especially with the traditional kernel based implementations. The adoption of VI architecture coupled with necessary HW support can reduce end-to-end latencies under 10 μs range [35,16,21,22], but this may still not satisfy emerging real-time financial data mining and modeling applications that require latencies as low as 1 μs. RDMA enabled network interfaces can reduce the latencies further, however, IBA with native RDMA generally provides significantly lower latencies [34]. One difficulty in implementing RDMA over TCP is the need for an intermediate layer called MPA to close the gap between the byte-stream nature of TCP and message oriented transfer expected by RDMA.

The main advantage of Ethernet stack is the TCP/UDP interface over which most applications operate. However, TCP was designed for highly variable open Internet environment – rather than data centers – and has numerous deficiencies from a data center perspective [24]. In particular, it is well known that achieving good QoS is very difficult with TCP since multiple competing TCP flows will tend to divide up the available bandwidth equally, rather than according to the specified fractions [36,11]. Similarly, TCP congestion control can be unnecessarily heavy-duty for data centers. In particular, TCP provides elaborate schemes to deal with packet losses, which rarely arise in well configured data centers. Packet losses can also be highly undesirable at high data rates in that they can substantially degrade application performance. Delay based TCP implementations [29] such as TCP-Vegas are much more appropriate for data centers but such versions are unfortunately not very popular.

TCP also suffers from a number of other weaknesses that have been addressed by other TCP compatible protocols such as SCTP (stream control transmission protocol). SCTP grew out of the need to emulate Signaling System No. 7 (SS7) capabilities in the Internet [8,25]. Although SCTP is an even more heavy-duty protocol than TCP and thus may be difficult to scale to high-speeds, it does offer a number of features that can be useful in data centers. These include:

1. Multi-homing, which allows a connection to use alternate paths in case of primary path failure. This is similar to IBA's automatic path migration (APM) feature.
2. Better resistance against denial of service (DoS) attacks by delaying memory allocation for connection information and challenge–response type of verification. In particular, SCTP does not suffer from the well known "SYN attack" of TCP.
3. Better robustness due to 32-bit CRC (vs. 16-bit for TCP) and built-in heart-beat mechanism. At high data rates, 16-bit CRC may lead to undetected errors quite frequently.
4. Protocol extensibility via the "chunk" mechanism, which allows introduction of new control message types.
5. Preservation of upper layer message boundaries, which simplifies RDMA implementation.
6. More flexible delivery (ordered or unordered, and control over number of retransmissions). For example, ordered delivery is unnecessary for RDMA.

SCTP also supports the concept of a "stream", which is a logical flow within a connection with its own ordering constraints. The stream concept allows different but related types of data to be transmitted semi-independently without having to establish and manage multiple connections. Unfortunately, most SCTP implementations do not optimize this feature [25], and its usefulness is unclear.

4.3. Data center networking challenges

In this section, we identify the networking requirements imposed by evolution in data centers and then expose the deficiencies of available fabrics according to those requirements. Ref. [23] discusses the requirements and approaches from the perspective of transport layer; here we discuss these in a more general setting.

The notion of distributed virtualized data centers (DVDC) discussed in Section 3 attempts to create the abstraction of a single data center that could be geographically distributed. While this is a useful abstraction, it is crucial to take advantage of the 2-level structure lying underneath: high BW, low-latency, nearly error free communication within a physical server patch, and much higher-latency and lower-speed communication environment between data centers. In particular, at the middleware level, resource allocation and migration should automatically account for this discrepancy. Similarly, at the transport level the protocols must be self-adjusting and capable of working well both for paths that stay entirely within a server patch and those that go across server patches.

Since intra and inter server patch communications have very different characteristics, they result in very different challenges. For example, a simple credit-based flow control (e.g., such as the one used in IBA) is appropriate for intra server-patch communications because of small round-trip times (RTTs), rare packet drops, and need for very low CPU overhead. On the other hand, for inter server patch communications, good throughput under packet drops due to congestion or errors is very important and hence a sophisticated control (as in TCP or SCTP) may be required.

Although wired networking technologies (copper and/or fiber versions) are expected to remain dominant in data centers, wireless technologies such as Wi-Fi, Ultra-wideband (UWB), and free-space optical are finding niche applications as their available BW increases. For example, the available wireless bandwidths may be adequate for low-end data centers running compute intensive applications. Even in larger data centers, wireless may be quite adequate as a management fabric. Wireless technologies have the important advantage of eliminating the wire management problem, allow for ad hoc addition/deletion to the infrastructure, and provide a convenient broadcast (as opposed to point to point) communication medium that can be exploited in clever ways. To support this diversity in MAC layers, it should be possible to choose the congestion control mechanism depending upon the MAC layers traversed [23,38]. For a connection oriented protocol, the congestion control can be negotiated during the connection setup; however, in some cases, automated dynamic adjustments may also be necessary.

As the MAC technologies evolve, they are marching towards unprecedented data rates. For example, a 12X GEN3 IBA link can support bandwidths of 120 GB/s, and 100 GB/s Ethernet is actively under development [31]. At 100 GB/s, an averaged size 1000 byte packet must be processed in less than 80 ns. With a complex protocol, it is very difficult to complete MAC, network and transport layer processing in 80 ns, particularly when memory accesses are involved. It is clear that ultimately the network speed bumps will be limited by the "memory-wall" phenomenon (see Section 7.1). Thus, in addition to direct placement of data in caches, It is thus necessary to go beyond the VI architecture and make the protocols as lean as possible. The leanness flies directly in the face of greater functionality required to address security, flexibility and other issues. At very high data rates, the entire protocol stack including MAC, network and transport layers must be thinned out. This can pose significant challenges in maintaining compatibility with standards.

As discussed in Section 4.1, the traditional tree network architecture provides limited cross-section bandwidth that could become a problem in large data centers. Ref. [15] addresses this problem by using many more but lower capacity switches at higher levels of the hierarchy arrange in a Clos or fat-tree topology [27]. One issue with such an approach is increase in the number of assets to be managed. Ref. [32] instead does away with spanning tree protocol by exploiting a centralized fabric manager for the entire network. This fabric manager uses hierarchical location based pseudo-MAC addresses to control routing while the edge switches translate between pseudo and real MAC addresses. Yet another approach is explored in [39] where standard Ethernet switches are replaced by new types of switches called "Axons" to which unmodified Ethernet hosts can connect. Axons use source routing based on the routing table at the ingress Axon. The routing table maintenance is done in SW running on Intel® Atom processor; the actual routing is performed in HW using FPGA. Another such attempt called Ethane [9] provides centralized control over the entire network. Note that the centralized control solutions can be vulnerable to failures and attacks and may have scalability issues of their own.

Section 3 introduced the concept of a *virtual cluster* (VC), which requires provisioning QoS controlled communication paths between virtual nodes. To enable this, it is necessary to tag all communications within a virtual cluster with tag so that it is possible to differentiate between multiple virtual clusters sharing the same communication paths. Also, it is necessary to think of QoS in terms of the overall application needs rather than the needs of an individual flow between two endpoints. This is the main distinction between the type of QoS we discuss here and the traditional QoS notions. The tags can be exploited to ensure that competing virtual clusters on a shared path are allocated bandwidth either according to some fixed criteria (e.g., relative priority or type of application being run on the virtual cluster) or based on dynamically changing needs of the applications. One way to estimate the bandwidth need dynamically is to keep track of actual bandwidth usage during uncongested periods and then divide up the available bandwidth in that proportion during congestion periods [10,24].

The tagging and corresponding bandwidth control can be implemented at various levels of network stack with different consequences. Tagging at the MAC level ensures that (layer-2) switches can participate in tag examination and BW management [26]. The data center Ethernet project in IEEE [40] is basically concerned with exploiting the existing three CoS (Class of Service) bits in Ethernet frame for such a tagging and bandwidth management. Expansion of such a mechanism to full-fledged virtual clusters would require significant perturbations to the existing Ethernet standard and would still require a mechanism at the IP layer to handle virtual clusters going across layer-2. At layer-3, the MPLS (multi-protocol label switching) already provides sophisticated mechanisms to tag flows [30,17] and the corresponding resource reservation mechanisms such as RSVP-TE [4] can be used to automate the setup. These can be used for inter server-patch path setups, but are not useful within a data center because of abundance of layer-2 switches. Finally, tagging at the transport layer is easy to implement but will have only endpoint significance. That is, while the tags can be used for congestion control of connections belonging to different virtual clusters, no enforcement will occur in the network itself. Ref. [23] proposes such a tagging mechanism along with the notion of collective bandwidth control described in [24] that automatically determines the needs of competing workloads and then attempts to allocate bandwidth proportionately during congestions. In general, an accurate control over bandwidths provided to competing applications can be quite challenging with TCP-like congestion control mechanisms.

In a virtualized environment, communication between nodes needs a mechanism to automatically detect when two nodes are located on the same platform and thus can communicate without involving the external network. Recent advances such as XenLoop [41] provide a transparent mechanism to automatically intercept packets of co-resident VMs and shepherd them via the shared memory interface. A somewhat similar issue arises for local vs. non-local communication. For example, if the intra-chassis fabric is different from the mainstream fabric (e.g., PCI-Express vs. Ethernet), it may be necessary to transparently switch between transport protocols appropriate for the media. Providing a low-overhead and transparent communication mechanism between VM's that may be migrated dynamically and hardware support of it remains a challenging problem.

As data centers move from owned physical entities to the DVDC model discussed here, it becomes much harder to protect them against denial of service and other types of attacks. Therefore, security mechanisms such as those adopted by SCTP become essential in spite of their substantial overhead. The big challenge is to ensure the scalability of these protection mechanisms at very high data rates that are likely to be seen in the future. Similarly, support for high availability mechanisms such as multi-homing, connection migration, path diversity, and path control become critical in this environment, but devising scalable solutions for them can be very challenging.

Data center networks often deploy a variety of network appliances or middle-boxes such as domain name servers, firewalls, load-balancers, Network address translation (NAT) devices, virtual private network (VPN) gateways, malware scanning appliances, protocol accelerators, etc. The deployment, configuration, traffic engineering, and keeping them up to date is often a very challenging task and continues to increase in complexity as data centers grow, but has not received much attention in the literature. Managing these devices in a DVDC environment can be particularly challenging since the middle boxes themselves might need to be virtualized without compromising the security, isolation, and performance features they are designed to provide.

5. Data center storage

In spite of tremendous growth in storage capacity in the last decade, the data tsunami shows no sign of abating; in fact, powered by new large scale-applications, higher-than-Moore's-law growth in computational capacity, and expanding global connectivity, the storage growth continues to accelerate. According to IDC estimates, the data volume continues to increase 50–70% per year. These trends make storage management in data centers extremely challenging. In this section, we provide an overview of emerging storage technologies and application needs and then discuss major challenges.

5.1. Storage basics

Until recently much of the storage relied upon rotating magnetic media, and the storage architectures developed around this technology. In this section we focus primarily on this media; the emerging solid state disk (SSD) technologies are addressed in Section 5.2.

Storage in data centers may take one (or a combination) of the following three forms: Direct attached storage (DAS), storage area network (SAN), and network attached storage (NAS). DAS refers to block-oriented storage directly attached to a server. SAN provides block-oriented storage that resides across a network. NAS also provides access to storage residing across a network but accessible via a higher level interface such as files or objects.

The dominant DAS technology has been the hard disk drive for quite some time and has continued to scale in performance. However, there are several shortcomings inherent to hard disks that are becoming harder to overcome as we move into faster and denser design regimes. In particular, the mechanical movement implies that disks will remain significantly faster for sequential accesses than for random accesses and the gap will only grow. This can severely limit the performance that hard disk-based systems are able to offer to workloads with significant random access component or lack of locality. Such performance deterioration is likely to be a problem in data centers where consolidation can result in the multiplexing of unrelated workloads imparting randomness to their aggregate. Although an individual state-of-the-art hard disk consumes significantly less power than other components of a server (e.g., about 12 W vs. 150 W for the processor subsystem), the large number of storage devices means that a 20–30% of the data center power could be consumed by storage.

The traditional secondary storage interface in the server world has been SCSI (Small Computer System interface). SCSI can handle up to 15 hard drives and throughput rates of up to 320 MB/s. Although a set of 15 simultaneously streaming hard drives can put out a much higher data rate, this is generally not a problem since even small fractions of random access patterns can seriously degrade the throughput of a traditional hard disk. However, an array of 15 solid-state or hybrid drives could easily exceed the limit, thereby implying the need for faster interfaces. The Serial-attached SCSI (SAS) interface is already replacing the traditional parallel SCSI interface with a serial link interface (as opposed to bus interface for parallel SCSI). (See Section 4.2.1 on more details regarding serial interfaces.) With 6 GB/s links, a SAS drive can provide throughput rates of up to 2.4 GB/s. Clients traditionally have used the parallel ATA interface, which too are being replaced by the serial version called SATA.

Although direct attached storage (DAS) on each server can provide the fastest access, it has numerous limitations in terms of size and flexibility. Consequently, per server storage is generally small and reserved for local data such as boot image and swap space. The sharable storage is generally provisioned separately in a ''storage tower'' and accessed via NAS or SAN. NAS provides a convenient file or object level access to the servers and can use traditional networking fabric such as Ethernet. However, the high level access may be too slow or unsuitable for applications that prefer to do their own storage management (e.g., database systems). Both NAS and SAN (discussed next) tap out at 8-16 TB storage limit because of 32-bit disk block addressing used in the implementations.

SAN provides the block-level access to remote storage and has traditionally used Fiber-Channel (FC) as the preferred networking technology. Although FC is specifically designed for storage access, the need for separate networking infrastructure and limited familiarity among the administrators makes it expensive to operate and maintain. iSCSI (Internet SCSI) is an alternative to FC and allows remote access to SCSI drives over the mainstream Ethernet. (The hardware interface could be serial or parallel and does not matter at the protocol level.) iSCSI typically runs on top of TCP and hence is easy to implement but the resulting heavy-duty layering can be significantly less performant than FC. This issue can be partly addressed by iSCSI cards that implement iSCSI and underlying TCP/IP in hardware. The emergence of inexpensive 10 GB/s Ethernet has also made iSCSI considerably more attractive.

Because of the prevalence of FC, many applications that use low-level storage access (e.g., database management systems) are designed for FC storage. Thus, even with an eventual trend towards the much cheaper iSCSI or similar Ethernet-based solutions, FC interfaces will be required for quite some time. The 8–16 TB limit for a FC SAN almost guarantees multiple islands of FC storage that need to be connected. Several standard protocols have been designed for interconnecting the FC and TCP/IP worlds. The FCIP protocol encapsulates FC packet into TCP packets for transmission across an IP network. The FCoE (FC over Ethernet) encapsulates FC packets into Ethernet frames and is thus not routable. The iFCP protocol is a gateway to gateway protocol that allows a direct transmission of the FC packet payload over an intervening TCP/IP network.

In general, a *storage volume* could be spread over multiple physical or logical storage devices, and a consistent view requires ''storage virtualization''. Storage virtualization could be host-based, network-based or storage device-based. A widely deployed host-based solution is *Logical Volume Manager* (LVM) where the host OS manages storage volumes spread over devices under its control. Network based virtualization instead accomplishes the same task using a few ''appliances'' directly connected to the server network. In a more sophisticated version of this approach the data and meta-data paths may go over separate networks. In yet another variation, the virtualization functionality may be integrated in the SAN switch itself (perhaps using ASICs), so that the switch can direct the request to the appropriate storage volume and thereby reduce the number of network hops. Finally, the storage device itself (or a front-end processor connected to a "storage tower") may provide this functionality. With most of these solutions the virtualization extends only to the scope of the controlling agent (host OS, appliance, switch, etc.) and interoperability becomes difficult since different OSes, appliances and storage devices may implement virtualization differently.

Unification of multiple virtualized storage subsystems requires a higher level entity to coordinate access across these subsystems. This unification is being required increasingly due to 16 TB limit for traditional NAS/SAN storage. A popular example of such coordination is the clustered file system (CFS). CFS consists of a number of "cluster heads" each of which is a specialized server that manages the storage under its control, provides cross-cluster mapping of files and objects, and supports transparent access to the files and objects stored anywhere and across cluster nodes. Nominal storage functions such as striping and mirroring need to be provided by the CFS across the cluster in a transparent manner. CFS also needs to provide resilience in the face of storage devices across clusters experiencing failures. Some examples of CFS, designed for large scale HPC environments, are Lustre, parallel virtual file system (PVFS) and IBM's general parallel file system (GPFS).

5.2. Solid-state and hybrid storage

The continued improvement in the cost and performance of flash based storage [43] has made solid state disks (SSDs) a viable technology in data centers. Furthermore, there are a host of other non-volatile RAM (NVRAM) technologies under development that may significantly alter the storage landscape in the near future. Some of the more prominent NVRAM technologies include magnetic RAM (MRAM), phase-change memory (PRAM or PCM), and Ferroelectic RAM (FeRAM) [54,50]. NVRAM technologies offer several advantages over the rotating magnetic media: lower and more predictable access latencies for random requests, smaller form factors, lower power consumption, lack of noise, and higher robustness to vibrations and temperature. Table 1 presents the key characteristics of the important NVRAM technologies that exist today — some are more mature than others. Since the NAND flash memory (simply flash henceforth) is the most mature and popular of these at this time, we will use it as the representative technology to drive our discussion.

We begin with some characteristics of flash based storage. Flash devices require an *erase* operation before data can be written, and erases can only be done in units of a *block*. A block comprises 64 or 128 physically contiguous "pages." A page is the granularity of individual reads and writes and is typically 2KB in size. An erase operation is not only very slow (about 2 ms for 128 K block) but also results in a slight degradation of the flash, thereby limiting the useful lifetime of a block. Each block typically has a lifetime of 10 K–100 K erase operations [46]. Wear-leveling techniques that distribute the physical block location such that erasures are evenly spread across the entire flash are an essential aspect of flash usage.

Each flash page can be in one of three different states: (i) *valid*, (ii) *invalid* and (iii) *free/erased*. When no data has been written to a page, it is in the erased state. A write can be done only to an erased (or clean) *page*, changing its state to valid. This restriction forces an written page to be left alone and instead write page updates to a different location. Such out-of-place writes result in pages whose contents are invalid (i.e., obsolete). A garbage collector (GC) runs periodically and identifies blocks that only contain invalid pages and erases them. During periods of GC, the throughput offered by a flash device can decrease significantly. The frequency of GC and its computational overheads worsen with increased randomness in writes.

Finally, flash memory cells could be either Single-Level-Cell (SLC) or Multi-Level-Cell (MLC). As the name implies, SLC stores one bit per cell and MLC stores more than one. MLC obviously provides higher density and thus lower overall cost, however, this comes at the expense of slower speed, significantly lower lifetime, and lower operating temperature (due to more likelihood of errors caused by leakage current at higher temperatures). Consequently, solid state drives (SSDs) invariably use SLC, with MLC more common in consumer applications such as thumb drives. The data given in Table 1 corresponds to SLC.

Table 1
Sample characteristics of some storage technologies.

	Latency			Power	Lifetime	Cost
	Read	Write	Erase	Consumption	(Write cycles)	($/MB)
SRAM [45]	2–3 ns	2–3 ns	N/A	1 W	N/A	1.24
SDRAM	40–75 ns	40–75 ns	N/A	2–10 mW	10^{15}	0.0073
NOR [57]	85 ns	6.5 μs	700 ms/blk	375–540 mW	100 K	0.9111
NAND [48]	16 μs	200 ns	2 ms/blk	.06–2.5 W SSD	100 K	0.0049
MRAM [45]	35 ns	35 ns	None	24 mW	10^{15}	36.66
FeRAM [50]	85 ns	85 ns	None	3.6 mW	10^{15}	47.04
PCM [51,59]	62 ns	300 ns	N/A	480 μW	$> 10^7$	N/A
Magnetic disk	1–5 ms	1–5 ms	None	5–15 W	MTTF = 1.2 Mhr	0.003

In order to maintain compatibility with hard drives, SSDs are designed to interface to standard I/O busses such as SCSI or SATA. An embedded processor implements the so called Flash Translation Layer (FTL) to hide the idiosyncrasies of flash so that the same software can work with both hard drives and SSDs. The key functionality implemented by the FTL includes: (i) translation from logical to physical addresses to allow for wear leveling, (ii) out-of-place updates and garbage collection, and (iii) wear-leveling policies. The quality of FTL implementation is a key to SSD performance; for example, it is found that for certain random write-dominated workloads (e.g., DBMS workloads explored in [52]), the overheads of GC and wear-leveling can sometimes make SSDs slower than HDDs. For sequential accesses, HDDs can easily outperform SSDs. Nevertheless, SSDs hold a lot of potential for higher and more predictable performance than HDDs.

Although SSDs may be useful as stand-alone secondary storage for very high throughput and low-latency applications, they are generally expected to remain in the supporting role for hard disks in the foreseeable future. Many papers have explored SSD as an intermediate layer in the storage hierarchy between main memory and HDD based secondary storage [55,53]. However, many other issues in the integration of SSD's and HDDs for faster and more consistent performance remain to be resolved.

5.3. Challenges in data center storage

Although virtualization and clustering provide mechanisms to handle large amounts of storage, the ever-increasing volume of stored data will continue to pose scalability challenges at multiple levels. Managing a large number of storage devices (which may be further divided into multiple clusters) do pose significant challenges in terms of performance and availability. Another issue concerns the efficient management of a huge number of objects (such as files) that a large storage system will be expected to host. The object sizes themselves could vary over a huge range. In particular, the typical Zipf-like file-size distribution implies that (a) data centers will have a large number of small files and (b) the files on the large end could be extremely large in size and could be spread over multiple devices or even clusters. Keeping track of large numbers of files involves challenges in how to efficiently represent, manage, and manipulate file meta-data. However, we cannot simply design mechanisms that work well only for very large file systems. The number of objects managed by various file system instances is itself likely to follow Zipf-like distributions. Consequently, meta-data management should be designed to take advantage of small file system sizes whenever that is the case. Similar issues apply with respect to file sizes as well. The design should be able to provide efficient mapping, access, and updates to not only huge files running into petabytes but also to small files that are only a few hundred bytes.

Many emerging applications involve working with large amounts of data – both permanent and transient (or temporary) kind. An adaptive trade-off between computation and storage can be useful in working with such data. For example, infrequently accessed data could be compressed or regenerated each time by running the appropriate transformation, simulation or filtering program. Quantification of such computation/storage trade-offs requires addressing issues such as (a) storage power vs. CPU power consumed, (b) performance impact of storage saving techniques, and (c) data placement and migration across the storage hierarchy.

Due to their significantly different operational characteristics from both HDDs and main memory technologies, SSDs require novel modeling approaches. In particular, out-of-place updates, garbage collection and wear-leveling perturb the access characteristics of the incoming traffic and need to be addressed in the modeling [44]. Also, as a SSD gets increasingly worn out with time, its erase operations slow down considerably, requiring more retries and bad block remapping, thereby reducing the effective throughput of the device [12]. The GC and wear-leveling algorithms also affect power consumption and lifetime in complex ways that are non-trivial to model.

A related issue with respect to SSD – and more generally NVRAM based storage – is to re-examine the traditional distinction between main memory and secondary storage access. When a HW thread stalls for disk access, the OS takes control and switches the thread to another SW process since the latency of IO completion is very large compared with the cost of a context switch. However, such a switch does not make sense for a memory access. With extremely fast solid state storage, such a distinction may no longer hold and an adaptive context switch mechanism may be required. Furthermore, a fast storage access exposes the high overhead of the traditional file-system layer, and it is necessary to reexamine traditional file-access model to make it substantially leaner. In this context, a relevant question to ask is whether intermediate storage layer should really be accessed as secondary storage or simply as a higher level memory, or perhaps as something in between? In this context, Refs. [51,59,49] examine multi-level memory system using NVRAM to assess the performance benefits of this approach.

The storage virtualization techniques discussed in Section 5.1 are primarily directed towards aggregating a large number of storage devices and creating the appearance of a single logical storage subsystem from which storage can be allocated to various applications. Such a view is inadequate for the virtualized data center (VDC) model discussed in Section 3. In particular, each VDC may require its own partition of storage with adequate isolation and protection from other VDCs, and yet it should be possible to move the partition boundaries as needed. Furthermore, it should be possible to manage the storage in each VDC at a fairly low level so that each VDC can configure the storage based on its needs. Providing such a "disaggregation" capability in addition to the usual aggregation capability is currently an open problem that is not addressed by the current cloud storage capabilities.

6. Configuration management in data centers

A comprehensive management of data center assets needs to deal with their entire *life cycle*. The life-cycle extends from the point the asset is initially brought into the data center until it is finally retired from service, as discussed in more detail in Section 6.1. Management usually involves two distinct parts: (a) operations and (b) control. Roughly speaking, operations refer to installation, configuration, patching and other coarse time-granularity activities whereas control refers to finer-grain management of resources. Traditionally, the control part has been handled by the "in-band" side (i.e., by OS and middleware) whereas the operations are handled by the out-of-band (OOB) side, which runs on the baseboard management controller (BMC). However, as we move to management of the more general DVDC model discussed in Section 3, this distinction between operations and control or the separation between OOB and in-band activities becomes less clear. For example, configuring or reconfiguring a virtual machine could be considered a part of control. Similarly, the information obtained from both OOB and in-band sides must be combined for an effective configuration and control.

Management of modern data centers involves a large number of issues which become more challenging as the number of objects to be managed increases. In the following we discuss these issues and the corresponding scalability problems.

6.1. Life-cycle management

It may be tempting to think of data centers and IT facilities as static, where the facilities are installed and then used for a long time before being retired. In reality, most facilities are subject to constant churn: new assets are brought in and installed, and existing ones are patched, reconfigured, repurposed, moved to another geographic location, or replaced. Consequently, it is important to automate these tasks as far as possible so that management activities can be done cost-effectively (e.g., with minimal IT administrator's time), rapidly, safely, and w/o subject to human errors. In the following, we elaborate on the challenges of such an automated life-cycle management.

Consider a situation where a new modular server arrives at a facility and is plugged into some empty slot in a rack or a chassis. In order to logically add this server to the facility, the following tasks are required:

1. *Discovery*: This step involves discovering the HW/SW configuration of each device, and the server as a whole so that it can be deployed correctly. The information produced by the discovery needs to be stored in a standard form so that it can be used for the qualification and provisioning steps discussed next.
2. *Qualification*: A new server could well host malicious HW/SW and cannot be allowed access to the facility w/o a proper assurance procedure. This step initially quarantines the server by firewalling it at the connecting switch so that it is unable to send packets to arbitrary destinations. Assurance checks involves at least 3 aspects: (a) authentication (perhaps based on a certificate stored in tamper proof memory [TPM] module of the server), (b) scanning to detect malware, and (c) compliance checks to ensure that it conforms to the desired IT policies.
3. *Bare metal provisioning*: This step prepares the server for installation and involves a variety of tasks such as HW partitioning, configuration, tuning, and loading basic system software. The partitioning/configuration is likely to depend on the server patch(es) to which the new asset will be added.
4. *Service provisioning*: This step would assign the server partitions (or even the VMs running on them) to the appropriate virtual data centers, and provision them with necessary application software, network/storage access, etc. so that they can start providing intended services.
5. *Monitoring and tuning*: This refers to constant monitoring of vital parameters of the server and taking suitable actions. Monitoring data from various HW and SW elements would typically involve filtering, storage and fusion in order to detect and resolve performance problems, minimize power consumption, determine security attacks, etc.
6. *Remediation*: Refers to activities related to fault detection/diagnosis, security related quarantine, repair, upgrade and replacement. Remediation may be required while the server is in use and thus may interfere with the service.

The first three steps in this list involve BMC, which is the only part of the server that will automatically come up when a new server is plugged into the slot. The provisioning starts with the BMC turning on the main server and communicating with its firmware in order to bootstrap the process of discovery, qualification and bare metal provisioning. Many of the other tasks can be done in OOB or in-band manner or by a combination of the two.

6.2. Management frameworks

There are two fundamental requirements to enable automated discovery and configuration: (a) Availability of configuration information in a standardized format at each device and at higher levels, and (b) A standardized framework for retrieving and processing this information. The Common Information Model (CIM) was developed by distributed management task force (DMTF) for describing computing and business entities and has been adopted widely [62]. CIM is a hierarchical, object-oriented management information language based on UML (unified modeling language) for defining objects and interdependencies between them. Other than structural relationships, CIM can express a variety of dependencies such as those between network connections and underlying network adapters, SW and the HW on which it runs, etc.

A CIM schema defines an object in the entity-relationship style and allows for object-oriented modeling concepts such as nested classes, instantiation, inheritance, and aggregation to allow compact description of complex systems in terms of its components. As an example, a CIM model of an Ethernet NIC will be expected to provide not only the physical structure of the NIC but also the parameters/capabilities that are needed for using and configuring the NIC, e.g. available PHY speeds or HW CRC check capability. The CIM model also provides their settings (e.g., current PHY speed and whether HW CRC is enabled) and methods to change the values.

DMTF has also developed Web-Based Enterprise Management (WBEM) specification that provides mechanisms to exchange CIM information in an interoperable and efficient manner. The components of WBEM include representation of CIM information using XML, CIM operations over HTTP, web services based management (WSMAN), CIM query language, CIM command language interface (CLI), and CIM service location protocol. A popular open-source implementation of WBEM is a called CIMOM (CIM object manager). WSMAN defines a set of operations over SOAP (simple object access protocol) that can be used to query and update CIM repositories. SOAP runs on top of HTTP and because of its plain-text nature, SOAP based operations are easy to debug but can be very heavy duty in terms of overhead.

CIM models represent systems and their parameters mostly at the structural level – much of the semantics of the parameters and the intelligence to properly set them is outside the purview of CIM. For example, CIM is not designed to specify complex relationships between parameter values of various entities or the conditions under which parameters should be set in a particular way. Traditionally, such intelligence is buried in management code. The world-wide-web consortium (W3C) has recently standardized the services modeling language (SML) [www.w3.org/XML/SML/#public_drafts] to fill this gap. SML can describe schemas using XML DTD's (data type definitions). SML documents can refer to elements in other SML documents and can specify complex relationships using schematron (www.schematron.com). Thus SML can allow for resource management based on declared constraints. However, specifying and processing complex constraints using a declarative language such as SML remains quite challenging.

6.3. Storage of management data

The CIM repository of a data center asset can be regarded as a local configuration database that can be queried and updated using WSMAN, CIM-CLI or other means. However, depending exclusively on CIM repository to make provisioning or other decisions becomes impractical even with a small number of servers for two reasons: (a) CIM repositories typically store detailed parameter values of individual devices rather than higher level attributes (e.g., server capacity) that are required for dynamic management and (b) access to CIM repositories is usually very slow because its firmware base and web-services interface. A workable management invariably requires some higher level database that holds not only portions of CIM repository contents but also some derived attributes that can be used more directly in making provisioning decisions. Such a database is often known as configuration management database (CMDB). In fact, a CMDB does not depend entirely on CIM repositories; it may also contain a significant amount of operational data obtained both from OOB and in-band interfaces.

In practice, management SW vendors offer a variety of products targeted towards managing specific aspects. For example, a number of packages are available for bare metal provisioning, performance monitoring, application management, migration, etc. We henceforth call them *External Management Packages* (EMPs). Many EMPs use private data repositories for convenience which may not be compatible with others. The data from some of the EMPs (usually those by the same vendor) may be consolidated into a single CMDB, but this still leaves multiple CMDBs. The net result is a number of repositories with overlapping but incompatible information. The alternative approach of a single comprehensive management system from a single vendor is also undesirable due to inflexibility and lock-in issues.

In the past, configuration databases – whether CIM-DB, package DB or CMDB – tended to be rather static. In fact, CIM-DB's – being firmware based and difficult to modify – still primarily contain information that may be occasionally modified via BIOS, EFI (extended firmware interface) or other pre-boot control program. An example of such an infrequently invoked function is enabling/disabling HW threading. On the other hand, an agile management requires access to a lot of dynamic information such as current power draw, utilization, and available BW. In a virtualized environment, even the configuration parameters such as amount of installed memory and virtual NIC BW become dynamically modifiable parameters. This dynamicity brings in a number of challenges and the solutions become increasingly difficult to scale up for large data centers.

Keeping the asset level information (e.g., current NIC speed) in a local repository (such as CIM-DB or other disk/SSD based repository) is attractive as it allows for a clean, decentralized and easily parallelizable management. In the virtualized environment, the parameters of all VMs running on the machine are also best maintained locally. However, decisions regarding where a new VM should be allocated would require at least part of the information available at a higher level.

The data duplication across multiple repositories immediately brings in the question of consistency maintenance and forces a careful consideration of what information is kept where and in what form. As one extreme, only the static (or rarely changed) data is retained into higher level DB's and all dynamic data is fetched from asset repository as needed. This approach quickly becomes unscalable as the dynamic data increases, particularly with respect to firmware resident information. On the other extreme, maintaining dynamic data primarily in external databases is not only unscalable but also introduces undesirable dependencies. For example, inability to access external database would corrupt asset configuration and cause crashes.

Clearly, an intermediate approach is desired, but there is no theoretical framework to guide what information should go where. The general idea would be to store more directly usable but more abstract information at higher levels; however, it is very challenging to formalize such a notion. As an example, the CMDB might store the *computation capacity* of the server, which, in turn, depends on lower level parameters such as the number of enabled cores or memory speed. In this case, if an update is made to the asset parameters, we need to determine if it affects the

CMDB data and if so, the derived CMDB data needs to be updated. The obvious pitfall here is that if the relationship between the abstracted data and lower level data is not explicit (e.g., buried in code), there is both the consistency (false negative) and unnecessary update (false positive) problem with updates.

To enable integrated management of a complex system, it is often necessary to interrelate information from multiple databases. For example, a provisioning package may maintain detailed resource usage information but only abstracted information about faults. In contrast, a package dealing with asset replacement/upgrade may do just the opposite. Combining information from these two databases can be very difficult because of differences in level of detail and the precise semantics of data. Furthermore, the filtering/abstraction employed on incoming data before storing it may be buried in the code rather than specified clearly or formally.

There are two approaches to coordinating multiple CMDBs and both involve a higher level entity which needs to have interfaces to each existing CMDB or EMP databases with which it interacts. One possibility is to let the higher level entity itself be a CMDB that maintains a coherent version of all relevant data abstracted from lower level databases. This is challenging not only in terms of creating a unified view of data but may also be unscalable due to the centralization of all data into one CMDB.

An alternate approach is to make the higher level entity a global reference directory (GRD) similar to the scalable enterprise resource directory (SERD) based architecture defined by Dell and implemented by Altiris (www.altiris.com/upload/dell_cto.pdf). Such an architecture is illustrated in Fig. 4 with a few specific packages and associated databases. It also shows a pool of resources to be managed. The main difference between GRD and top level CMDB is that GRD primarily stores "handles" to data located in other databases, and some *derived attributes* or higher level data that may be more directly useful in management decisions. The methods to relate derived objects to base object parameters are necessarily a part of this description. GRD also maintains two other things: (a) Policies for deciding which EMP to invoke and what parameters to pass to it, and (b) Triggers that define the management functionality

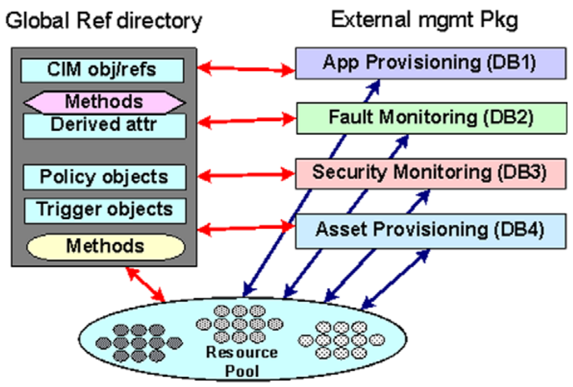

Fig. 4. Illustration of global resource directory based management.

that will be triggered either via alerts from EMPs or due to threshold crossings of data directly monitored in CIM-DBs. GRD can be implemented using LDAP (lightweight data access protocol) based implementations such as Microsoft Active-Directory or Novel's e-directory. Such implementations are highly scalable for read-only data, but are not designed to handle highly dynamic data.

In the context of our 4-layer DVDC model of Section 3, configuration management is required at all four layers. For example, the PIL needs to manage an entire server farm and provide support for creating server patches. The VIL should be able to use these capabilities to create and manage virtual data centers. The VICL then uses the VIL capabilities at multiple locations in order to support the DVDC concept. Finally, the SPL needs to manage applications and their deployment and should have enough visibility into the lower layers in order to make appropriate provisioning and re-provisioning decisions. Providing adequate visibility, abstraction and protection across this hierarchy and doing so in a scalable fashion poses significant management challenges.

6.4. Data collection, filtering and fusion

The management of various life-cycle stages discussed in Section 6.1 require different types of data ranging from very static to highly dynamic. The most dynamic data relates to the Monitoring and Tuning phase (and to a lesser extent to remediation phase). Active monitoring can easily generate so much data that its resource requirements rival or exceed that of the application data. This immediately brings the challenging problem of collecting and intelligently filtering the data so as to simultaneously satisfy conflicting goals of minimizing the amount of stored data and ensuring that no important events or characteristics are missed. In addition, as the number of assets in the data center increase, more intelligent techniques are required to ensure that the complexity and latency of migration/reconfiguration decisions increases significantly slower than linear in the number of nodes.

In large systems, an effective data collection, filtering and fusion must necessarily be opportunistic since statically deciding what data to collect, how to filter it and how to fuse data from different nodes becomes unscalable. In particular, we would normally like to keep the data collection sparse, use aggressive filtering, and avoid large scale data fusion until more detailed information collection is required. In general, it is very difficult to recognize onset of "interesting events" with sparse data, since by definition, an advance recognition of interesting events requires more aggressive data collection. Machine learning approaches can be useful in this context but are traditionally not designed for this environment.

One important aspect in data collection and manipulation is its overhead both in terms of processing as well as communication among nodes. The interference between application and opportunistic monitoring can lead to serious coupling and even instabilities and needs to be avoided. Consider, for example, a situation where the monitoring detects the onset of congestion and goes on to do more intensive monitoring thereby creating more overhead and perhaps more congestion. This dynamics, coupled with the dynamics of the application, could lead to false corrective actions, oscillations or other problems. In theory, these problems can be alleviated by capacity reservation; however, this can be quite difficult or infeasible especially for communication bandwidth. Instead, a more intelligent control over the capacities used by the operations side is required.

One way to isolate operations related communications from regular ones is by keeping the former entirely on the OOB infrastructure. In particular, any communication of data required for fusion is marshaled by the BMC over the management network. While achieving isolation, this approach raises several issues of its own:

1. Limited bandwidth on the management network. Current proposals on sideband network bandwidth allow up to 100 MB/s. While such bandwidth is plenty for routine management alerts, it can be quite inadequate for aggressive usage.
2. Communication limitations between in-band and OOB sides. Typically, this communication is handled via a device interface, i.e., DMA into or out of main memory with BMC acting as a device. Such transfers can be slow.
3. BMC processing limitations. The platform data available to the BMC may need to be filtered and reduced before communication, but the limited processing power of BMC may make this difficult.

It is clear from the above discussion that the data collection, filtering and fusion is a challenging problem that becomes more acute with the data center size. It is also important to realize that as the complexity and latency of these tasks increases, the ultimate control itself becomes more difficult because of the problem of lag, i.e., remedial action based on conditions that no longer exist. These problems become even more complex for DVDCs if applications span across geographic locations.

6.5. Service provisioning challenges

In a large heterogeneous virtualized environment, provisioning of an application or a service can be quite a challenging problem [64,66]. Generally, a new service needs to use several servers and locating appropriate servers requires at least three aspects: (a) residual server capacity, (b) available network and storage bandwidth, and (c) access latencies to the data that the application is intended to work with. For clustered applications, there is also the fourth element that relates to the inter-server communication bandwidth and latency.

There are at least three problems to be solved in accomplishing these tasks: (a) Translation of application workload characteristics into *required capacities*, (b) Estimation of *available capacities* of servers and network, and (c) Design of algorithms to map applications/services based on available capacities and features. Each of these is a very difficult issue and becomes even more so for DVDC's running a wide variety of workloads. Furthermore, the notion of "available" and "required" capacities assumes additivity. In general, workloads may interfere with one another and the additivity property does not hold. The notion of equivalent bandwidth is often used in networking context to allow additivity [63]; a similar notion is needed for computational and storage capacities as well.

For certain environments and applications, the available and required capacities may fluctuate significantly, an accurate initial estimate may not even be very valuable. One extreme method is to select servers based on minimal information on utilization of various resources and known gross workload features (e.g., CPU bound vs. IO bound). In this case, the problem of adequate capacity allocation is handled via performance driven dynamic VM resizing or migration. A more accurate estimation of available capacity can be done by the BMC or higher level controller running "micro-benchmarks" and required capacities via workload modeling. Obviously, there is a tradeoff between accuracy of the estimate, workload stability and migration frequency, but this is not easy to characterize. Machine learning techniques such as those in [67] can be useful in this regard.

Dynamic reprovisioning of a service or application can be triggered by one of three considerations: (a) resource oversubscription at one or more nodes (including communication bandwidth), (b) optimization considerations (e.g., moving the application from a lightly loaded server so that the server can be place in a low power state), or (c) occurrence of specific events such as failures or maintenance activities. Of these, (a) and (b) need to tradeoff several factors including cost of not making the change, monitoring cost, reprovisioning cost, and cost of incorrect choice of servers to which the workload is moved. In most cases, it is difficult to make these tradeoffs because of the complexity of the environment. Ref. [60] discusses the use of machine learning techniques for coordinated management of multiple resources in multiprocessors – similar techniques can be useful in more general dynamic provisioning contexts as well. In case of (c), the most important aspect is to quickly resume service instead of making the optimal choice of a new server. For example, the service may be first moved to another server in the same chassis/rack to minimize VM migration latency and yet another movement contemplated later accordingly considerations (a) and (b).

6.6. Challenges in management processing

The discussion in Section 6.3 focused on management database hierarchy and the issues brought about by multiple databases. This discussion implicitly assumed that the external management packages (EMPs) can transparently handle all assets in the data center. The asset data itself could be contained in a single database or partitioned at the highest levels only (e.g., database per physical site). However, the management functionality itself requires a more decentralized structure [69]. For example, the data center architecture forces a management hierarchy involving server level, chassis/rack-level, and server patch level. In fact, a comprehensive management involves multiple *domains* and a hierarchy within each domain. In a virtualized data center, there are at least four distinct domains of interest, each with a management hierarchy as illustrated in Fig. 5. These domains and potential levels are briefly described below:

1. *Physical assets*: This hierarchy deals with physical groupings of assets and various attributes of each physical group (e.g., current and maximum allowed power consumption, number of underutilized servers, etc.). The top level in this hierarchy is relevant only if the data center spans across multiple physical locations.
2. *Virtual assets*: This hierarchy deals with virtual machines and their groupings in terms of application cluster, virtual data center (defined over a server farm), and the entire DVDC. This hierarchy is required for provisioning resources for applications and virtual data centers.
3. *Network infrastructure*: Network infrastructure management deals with the management plane of switches and routers. This hierarchy reflects the physical network structure. Network management across physical server farms is the domain of ISPs and hence not included.
4. *Software infrastructure*: Software infrastructure is concerned with keeping track of software components and their dependencies.

The main purpose of the hierarchical structure is to simplify management. The hierarchy requires two important functions: (a) decomposition of a higher level request into sub-requests that can be delegated to lower levels, and (b) propagation of consolidated results and exceptions to the higher level. For example, when provisioning an application requiring many servers, we can choose the set of racks that will host the application, and leave the task of choosing actual servers within the rack to the rack-manager. This would allow proprietary algorithms to be used within a rack – the only thing that needs to be standardized is the interface between levels. If the racks level is unable to handle the assigned task, it would raise an exception to the higher level. Such a mechanism provides for a whole continuum of inter-level interaction policies: on one extreme, the higher level can select the next level entities almost randomly and depend on the exception mechanism to correct things, and on the other the delegation is carefully managed so that exception are truly rare.

While the challenges of negotiation, delegation and exception feedback arise in any hierarchy, they are further complicated by the presence of multiple incompatible databases. Also, many activities require cooperation between various domains: for example, provisioning a clustered application requires establishing a new group of VMs; however, the mapping of this group on to the physical infrastructure requires interaction between virtual and physical domains. In other words, the controllers of the four hierarchies shown in Fig. 5 do not operate in independently; they need to communicate and coordinate in order to accomplish various life-cycle tasks discussed in Section 6.1. Thus designing an overall architecture of cooperating hierarchies of controllers is itself a challenging task.

The question of in-band vs. OOB control is also important in designing a comprehensive architecture. As stated earlier, the server level OOB processor (or BMC) monitors and controls certain aspects such as server status, fan-speeds or power draw whereas the in-band controller is more concerned with performance issues. In general, it is possible that OOB and in-band functionalities have their own hierarchies, each supplied by a different vendor. Coordination between the two sides in this case is difficult but essential for an effective management.

Because of their modular nature, data center assets can be easily moved around, and usually do for a variety of reasons. In a large data center, it becomes difficult to keep track of these assets. Thus asset management has emerged as an important problem in data centers and some solutions for localizing servers in a data center have appeared [61]. Reference [68] shows how the emerging wireless USB standard can be exploited for accurate asset localization and reference [65] builds on it to provide location based services in the data center.

7. Power and thermal management

The importance of effective power and thermal management in data centers was introduced in Section 2.5. In what follows, we first provide necessary technological background on different aspects of power and thermal management and then identify key opportunities and challenges, existing methodologies, and problems that need to be addressed.

Reducing power consumption in data centers involves a number of facets including: (a) low power hardware design, (b) restructuring of software to reduce power consumption,

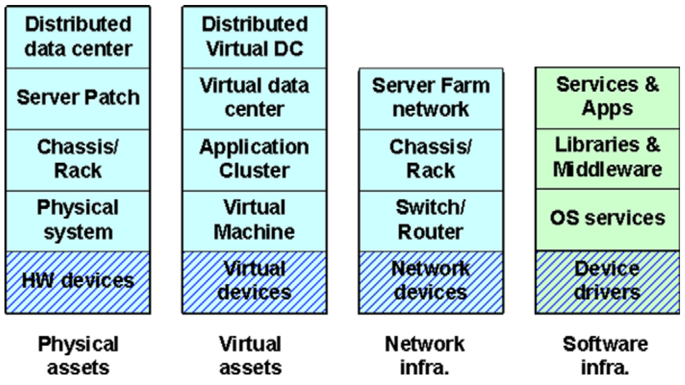

Fig. 5. Illustration of levels of management of IT resources.

(c) exploitation of various power states available in hardware, and (d) proper design, usage and control of data center infrastructure [106]. Of these, (a) and (b) are not specific to data centers, therefore, we address them here only briefly.

As the hardware technology progresses, it will continue to use smaller feature sizes, which automatically reduces device power consumption. However, the increasing wire resistance, clock rates, and transistor counts not only increase overall power consumption but also result in enormous and unsustainable power and current densities. Furthermore, the desire to quickly enter/exit low power state could result in intolerable di/dt (rate of change of current). This further implies that we may be unable to reduce these latencies very much. Similarly, as the feature size decreases, the thickness of the insulator layer will decrease and hence the leakage current will increase. This remains true in spite of recent success of high dielectric insulator layer.

With respect to software design issues, while it is generally true that software that is optimized for performance would also be power efficient, power efficiency is not synonymous with computational efficiency. In particular, batching of computations and data transfers improves power efficiency since it elongates idle periods and allows devices to go into low power states [96]. Also, certain operations or sequence of operations take less energy than others for equivalent effective work. Although low power software design is well studied for embedded software [103], development of general frameworks for low-power software design and characterization of its power-performance tradeoff is a major outstanding challenge.

In the following, we shall elaborate on aspects (c) and (d) in some detail since they are central to much of ongoing work on data center power management.

Almost all major components comprising a modern server offer control knobs in the form of *power states* – a collection of operational modes that trade off power consumption for performance in different ways. Power control techniques can be defined at multiple levels within the hardware/software hierarchy with intricate relationships between knobs across layers. We call a power state for a component *active* if the component remains operational while in that state; otherwise we call the state *inactive*. These active and inactive states offer *temporal power control* for the associated components. Another form of power control is *spatial* in nature, wherein identical copies of a replicated component operate in different active/inactive states, with the overall power/performance trade-off depending on the combination of states. The more general *integrated power control* refers to a cooperative power management of multiple homogeneous or heterogeneous components.

7.1. Active and inactive power states

A computing platform and most devices that are a part of the platform provide active and inactive states. The Advanced Configuration and Power Interface (ACPI) provides a standardized nomenclature for these states and also defines SW interfaces for managing them. ACPI is implemented in all major OSes as the OS-directed Power Management (OSPM).

With respect to the computing part of the platform, ACPI defines five "system" states denoted S0...S5, with the following being the most relevant: S0 (working), S3 (standby – or inactive with state saved into DRAM), and S5 (hibernating – or inactive with state saved into secondary storage). In S0, ACPI defines various abstract device states that can be mapped to the states actually provided by the hardware, as discussed in the following.

7.1.1. CPU power states

The CPU power consumption depends on the number of processing cores, their operational frequencies and voltages, and the workload. CPU offers the richest set of power states. For single-core CPUs, these are the C states (inactive) and P and T states (active) as shown in Fig. 6.

The best known states are the P (performance) states, where P0 refers to the highest frequency state and P1, P2, etc. refer to progressively lower frequency states. Lower frequency allows operation at a lower voltage as well and thus each P state corresponds to a supported (voltage, frequency) pair as illustrated in Fig. 6. The active power consumption is proportional to frequency but goes as square of the voltage – thus a simultaneous voltage and frequency reduction can result in cubic decrease in the power consumption. Furthermore, a lower voltage decreases the leakage current as well and thus results in lower static power consumption also. For this reason, dynamic voltage-frequency switching (DVFS) is among the most explored power management technologies. Many papers have explored compiler and OS assisted use of DVFS to optimize power consumption [70,109,83].

Although transitions are possible from any P state to another, a significant transition latency may be involved depending on the implementation. In particular, the implementation needs to allow for not only the underlying hardware latencies such as voltage settling time and locking time to a new frequency but also software overhead such as ACPI table lookup, user-kernel mode transition, and running the decision algorithm. It is crucial to consider these switching overheads in estimating the power-performance tradeoff due to P state usage.

The successive generations of the semiconductor technology not only reduce feature size, but also operating voltages. The upcoming 22 nm technology would likely operate at 1.1 V, which leaves very little room for further lowering of voltages. Consequently, future P states are likely to be primarily a frequency play and thus not very attractive for reducing power consumption. In fact, it can be argued that with relatively little scope for voltage changes, the so called "race to halt" strategy may be preferred than DVFS. Race to halt refers to the strategy of finishing up work at the highest possible rate and then move to an inactive state.

T (throttling) states assert STPCLK (stop clock) signal every few clock-cycles and thus enforce a duty cycle in CPU activity. State T0 corresponds to 100% duty cycle (i.e., no STPCLK), and states T1, T2, etc. to progressively lower duty cycles as shown in Fig. 6. T states are intended primarily for thermal control (e.g., to ensure that junction temperature does not become too high) and may use long STPCLK periods. The CPU stalls introduced by T states are

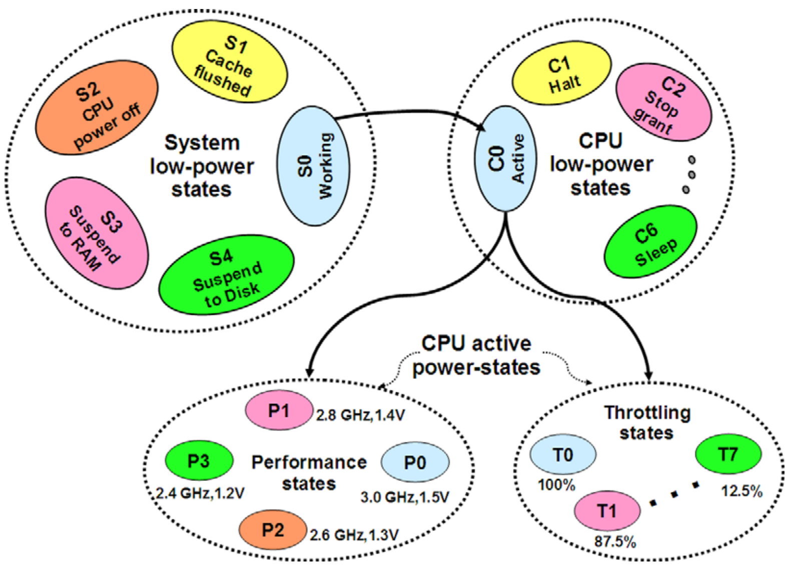

Fig. 6. System vs. CPU power states.

usually a performance killer, but performance is of secondary importance in thermal protection scenarios.

The basic inactive states are denoted as C0, C1, C2,..., where C0 is the active (operational) state and others are inactive states with increasing power savings and entry/exit latencies. When in the C0 state, the CPU can also transition to P and T states. The interpretation of and transitions between various C states are architecture dependent. Generally, C1 and C2 states only turn off clock whereas higher states may also flush core-specific or even shared caches and may lower the voltage. It is expected that future processors will have even deeper inactive states. The realization of various inactive states and hence the power savings they offer is vendor dependent. For example, in some current processors, the interesting C states are C1, C3, and C6, with power consumption (as a percentage of C0 power consumption) in the range of 50%, 25%, and 10% respectively, and transition latencies of the order of 10 ns, a few µs, and about 10 µs, respectively.

In case of a multi-core CPU, most of the above states apply to individual cores as well. However, there are some important differences, primarily relating to the architectural considerations. For example, if all cores lie on the same voltage plane, they can only allow independent frequency control and thus limit the "P" states for cores. We refer to core states using "C" as a prefix, e.g., core C states are called CC states, core P states CP states, etc. The overall or package-level CPU states are still meaningful, and indicate how the non-core, package level logic should be handled. This logic includes core-interconnect, integrated memory controller, shared cache, etc. Clearly, the package state can be no lower power than the highest power core state. For example, if some cores are in CC6 state while others are in CC3, the package state will be generally set as C3 and suitable low power states to use for non-core components can be decided accordingly. A CPU package state could imply more global actions as well, which we discuss under integrated control.

7.1.2. Memory power states

Memory is composed of a number of DIMMs, and the memory power consumption is a function of number of DIMMs, DIMM size, channel speed, and channel utilization. Here we focus on DIMMs based on the popular double-data rate (DDR) technology which has evolved from DDR1 to DDR2 to the now proliferating DDR3. Although DDR technology continues to drive down per GB power consumption, the ever increasing appetite for more memory is already making memory power consumption rival CPU consumption. Thus aggressive management of memory power is essential.

Each DDR DIMM is divided into "ranks", with 1, 2, or 4 ranks per DIMM. A "rank" is a set of memory devices that can independently provide the entire 64 bits (8 bytes) needed to transfer a chunk over the memory *channel* that the DIMM is connected to. The most common server DIMMs are dual-rank with ECC (error correcting code) enabled and involve nine x8 devices per rank. Each DDR3 device is internally organized as a set of 8 "banks" that can be accessed in parallel. Memory controllers often support multiple channels, each allowing one or more DIMMs and capable of independent data transfer. Since the data from all ranks of all DIMMs on a channel must flow over that channel, the ranks can be lightly utilized even if the channel is quite busy.

As its name implies, a DDR DRAM transfers 8 bytes (64 bits) of data on both edges of the clock. It thus takes four cycles to transfer a typical 64 byte cacheline. DDR3 can support clock frequencies of up to 1 GHz, which translates into 16 GB/s per channel, which can be quite plentiful. Unfortunately, DRAM latencies have stayed almost constant even as the clock rates have increased substantially. This trend is expected to continue in the foreseeable future and is often referred to as the *memory wall* phenomenon.

DDR technology includes several mechanisms to reduce power consumption. The most basic is the use of lower voltages in newer versions (e.g., 1.5 V in DDR3 vs. 1.8 V

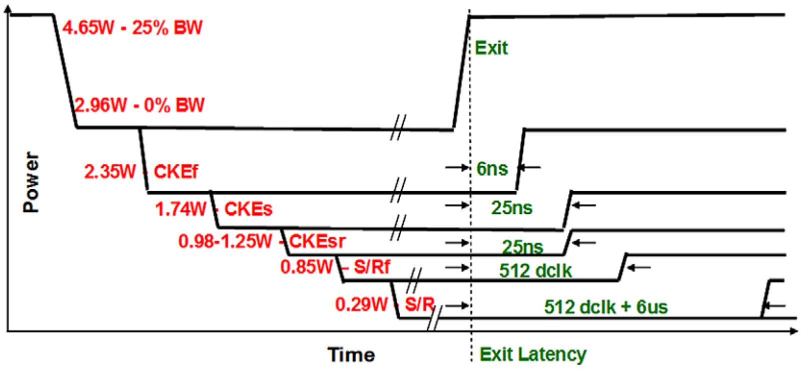

Fig. 7. DDR memory power states.

for DDR2, and 1.2 V for future DDR4). Through dynamic management of electrical terminations such as input buffer termination (IBT) and on-die termination (ODT), the technology minimizes the standby power consumption and improves signal integrity at the cost of some additional latency. In particular, with 2 GB DIMMs the standby power consumption is approximately 3 W whereas the 100% loaded power consumption is approximately 9.8 W. Several low-power idle states have also been identified and are briefly explained below:

- *Fast CKE:* In this state (denoted CKEf) the clock enable (CKE) signal for a rank is de-asserted and I/O buffers, sense amplifiers, and row/column decoders are all deactivated. However, the DLL (digital locked loop) is left running.
- *Slow CKE:* In this state (denoted CKEs), the DLL is also turned off, resulting in lower power consumption but higher exit latency. If all ranks of a DIMM are in slow CKE mode, the DIMM register can also be turned off thereby leading to even lower power consumption w/o any additional latency. Further circuitry can be turned off if all DIMMs on a channel are in CKEs mode. We denote these modes as CKEsr.
- *Self-refresh:* In this mode much of the DRAM circuitry is placed in an inactive low power mode and the DRAM refreshes itself rather than under the control of the memory controller. There are two such modes: S/Rf (self-refresh fast) where the phase locked loop (PLL) required for synchronizing DRAM clock with external clock signal remains on, and S/Rs (self-refresh slow) where it is turned off.

Fig. 7 shows power consumption and exit latencies in various power modes of DRAM using a "waterfall diagram".[2] It is seen that CKEf can save 1.5 W per DIMM with only four DRAM clocks (dclks) of latency. On the other hand, CKEs saves only 0.6 W more with considerably larger latency. It would appear that a right strategy would be to use CKEf frequently and during longer idle periods *promote* the state from CKEf to CKEs. Unfortunately, transitions between the two modes are expensive, adding some 12 dclks of latency.

[2] The stated numbers are for a 2-rank, 2 GB DDR3/1333 DIMM with 2x refresh rate, and could vary a lot from vendor to vendor.

It is seen that the self-refresh latencies are extremely long, even without PLL off. Thus, self-refresh is not useful in normal (C0) operating mode; self-refresh is typically employed when the entire socket is in low-power mode such as C6.

In the above, we consider inactive power states of memory. By running DRAM at lower clock rates, the active power can be reduced significantly at the cost of lower memory BW. The clock rate reduction often does *not* increase access latencies significantly since the RAS, CAS and page close operations can be performed in fewer clocks at lower clock rates. This situation is a result of the "memory wall" phenomenon discussed above.

7.1.3. Interconnection network and links

Modern computer systems use a variety of networking media both "inside-the-box" and outside. The out-of-box interconnects such as Ethernet, Fiber-Channel, and InfiniBand, are well known and their ports can consume a substantial percentage of the IT power in a large data center. However, internal interconnects can also collectively consume a significant percentage of platform power due to their number, high switching rates and increasing silicon wire resistance [98]. A modern platform sports a number of inter-connects including PCI-Express (PCI-E), link between "south-bridge" and "north-bridge", processor-memory interconnect (such as QuickPath™ or HyperTransport™), and inter-core interconnects. An intelligent power management of such interconnects also becomes crucial for platform power reduction.

As with other devices, the active power modes in links result from the provision of multiple speeds. Depending on the type of link, the speed change may be either a matter of simply changing the clock rate or a switch-over to a different PHY. An example of the latter is the Ethernet operating at standard rates such as 100 MB/s or 1 GB/s. Such a PHY switch can be extremely slow. Even the clock rate changes can be quite slow since they invariably require a handshake to ensure that both directions are operating at the same clock rate. The energy efficient Ethernet [81,72] is considering a rapid PHY switching scheme, but even this can be extremely slow for a fine granularity control.

Most current links support at least two low power states, called L0s and L1, respectively [86]. For L0s, the power consumption ranges between 20% and 50% of the idle (or L0 state) power consumption and the entry/exit latencies are in the tens to hundreds ns range. L1 power consumption is generally much smaller at the cost of exit latencies in the range of several microseconds. These very high exit latencies make L1 unsuitable for internal links in the C0 state. Whereas most existing link control algorithms deployed are reactive in nature (e.g., exit from low power state when the traffic does arrive), it is possible to consider *proactive algorithms* that attempt to be ready for arriving traffic. The efficacy of combining reactive and predictive techniques for link state control has been investigated in [86]. More sophisticated schemes, appropriate for out-of-box links have also been explored [79].

7.1.4. Storage media

The "data tsunami" problem discussed earlier means that the power consumption in storage media will remain a significant percentage of data center power consumption. Yet, unlike CPU and memory, the traditional magnetic disks do not allow aggressive power management. With magnetic disks, much of the power consumption is related to simply spinning the disk. Dynamic changes to the RPM of hard disks (DRPM) has been proposed as one method of disk power control [47]. However, at low rational speeds, the applications will experience substantially higher latencies and potentially unsatisfactory performance, even if the disk utilization is low. Furthermore, the implications of dynamically varying rotational speeds on the reliability of disks are still unclear.

7.2. Spatial control

7.2.1. CPU cores

The package states introduced in Section 7.1 can be considered as a special form of spatial control that relates to all cores. However, several other forms of spatial control can be used to in interesting ways:

1. If nearby cores are inactive, it may be possible to run at a frequency above that of the CP0 state (potentially by raising voltage above CP0 level). This is referred to as *turbo mode* and is useful for workloads that do not scale well with the number of cores.
2. Ideally, running a core constantly in CP0 should not result in any thermal events (e.g., PROCHOT signal being asserted). However, ensuring this requires substantial margins in chip, package and heat-sink design. Instead, it may be easier to keep a few unused cores and rotate workload among them based on some thermal threshold. This is normally referred to as *core hopping*.
3. With many cores available in future CPUs, it is expected that the cores will not be well utilized. In this case it helps to consolidate workload on a few cores that are evenly distributed on the chip and run them in CP0 state and put others in deep sleep mode. This is the traditional *width control* applied to CPU cores.

In practice, it is possible to combine these and other spatial mechanisms in order to match CPU power and performance profile to the applications. Several papers have considered spatial control, but primarily using active states. For example, Ref. [107] considers a closed loop control for multi-core processors that keeps temperature below a certain limit. Reference [85] considers DVFS control of cores to meet given power budgets.

7.2.2. Interconnection links

Most of the link types described earlier are now based on bit-serial technologies with differential signaling where the link bandwidth is scaled by running multiple "lanes." Such links allow dynamic width changes wherein certain lanes can be put in low power modes to trade-off power consumption against bandwidth. A highly desirable feature of width control is that so long as some lanes are active, the non-zero communication bandwidth significantly reduces the impact of high latencies associated with the state change.

A dynamic width control algorithm has to operate within constraints of feasible widths associated with the underlying link hardware. For example for a x10 link, the supported widths may be only x10, x4, x2, and x1. As with temporal control, both reactive and proactive techniques can be used for link width control. In [87], we discuss a complete algorithm called DWCA (dynamic width control algorithm) and show that it easily outperforms the link power state control algorithm in terms of latency impact and power savings. Note that when there is no traffic to transmit, the width can and should be reduced down to zero. Thus, DWCA does include link power state control as a special case.

7.2.3. Memory and secondary storage

In terms of spatial control of memory, it is possible to put a subset of ranks in low power mode in a synchronized manner in order to have only a fraction ranks active at a time and thereby reduce the power consumption. Ref. [89] discusses a closed loop control scheme that attempts to limit performance degradation for such a synchronized control. More intrusive controls are also possible: for example, one could copy "hot" pages from an entire DIMM to another DIMM and then put the DIMM in deep sleep mode (e.g., S/Rs); however, the effectiveness of such controls needs to be evaluated carefully because of significant latency, BW usage, and power consumption associated with data movement.

The most basic spatial power control for hard disk drives is to spin-down idle disks. Some recent research has also considered shutting down a subset of the disks within RAID arrays to reduce power consumption without hurting performance by using smarter data placement and dynamic rearrangement [56,58].

The substantial storage related power consumption in data centers can be alleviated by SSDs. As shown in Table 1, all NVRAM technologies consume an order of magnitude lower power than hard disk drives, and virtually no idle power. The drawbacks inherent in most solid-state technologies with respect to their lifetimes and higher costs are likely to limit the extent of benefits they bring. As technological breakthroughs in solid-state technologies improve their life-time and performance, they should play an increasing role in reducing storage subsystem power. Meanwhile, complementary efforts are likely to continue on various ways to make disk-based storage more power-efficient. Novel data layout techniques along with in-memory caching/buffering schemes can be used reduce seek related power consumption. An example of this is the use of fractal trees [42] instead of the traditional B-trees in databases. The data layout could also be altered dynamically or data could be migrated in order to facilitate shutting down subsets of disks while attempting to minimize the impact on performance [58]. As an example, data that is less performance critical could be stored on slower disks that consume less power.

7.3. Integrated power management

Integrated control is concerned with coordinated management of heterogeneous components or devices building upon the individual temporal and spatial control knobs discussed above. Integrated power management can be viewed along two dimensions: (i) horizontal, which co-ordinates the control knobs within one level of the hardware/software stack and (ii) vertical, which co-ordinates the operation across different layers.

The CPU package state introduced earlier can be thought of as a composite state that involves integrated power management of not only the cores but other non-core entities such as core-interconnect, shared cache and integrated memory controller. In particular, if all the cores are in a state where their core caches are flushed, the corresponding package state may specify partly flushing of the shared cache as well. A low-power package state could even trigger power state transitions for components outside the package. For example, if all two CPU packages are in, say, C6, the interconnect between them could be transitioned to L1 state, perhaps after some suitable delay. However, much of this is currently handled in an ad hoc manner.

Modern processors are beginning to design functionality to exercise the power control options for various components in a coordinated fashion. For example, the power management unit (PMU) is a dedicated micro-controller on the newer Intel cores whose sole purpose is to manage the power envelope of the processor (See communities.intel.com/community/openportit/server/blog/2009/04/27/). This design allows for increased flexibility by allowing functionality to be moved to firmware rather than being hardware-based. The control mechanisms implemented by the power control unit are driven by real-time measurements from sensors built into the main cores that monitor temperature, power, and current. There could be a hierarchy of PMUs, with those at the higher levels operating at coarser temporal and spatial granularities and interacting with lower level PMUs. An effective control may require higher level PMU(s) to coordinate with the OOB side that collects a variety of board and system level power/thermal data.

Horizontal integrated control of heterogeneous devices has become an interesting and viable option in secondary storage with the presence of NVRAM technologies. As discussed in Section 5.2, the NAND Flash-based drives consume significantly lower power compared to magnetic hard disk drives. They could be used for building more power-efficient storage systems in data centers where they are used for selective storage of performance-critical and popular content thereby providing increased opportunities for turning off the power-hungry hard disk drives. As discussed earlier, such design needs to carefully consider the higher costs of SSDs as well as the reliability problems inherent in them.

Vertical co-ordination must deal with appropriate mappings between the control options defined across different layers. In particular, the power management done by PMU could benefit from "hints" provided by the OS or the application. The OS can specify to the CPU the maximum tolerable latency via the "MWAIT" instruction so that the PMU can do a better job of choosing the C states. Similar arguments hold at other levels, e.g., application or middleware providing hints to the OS and for the control of other types of states.

7.4. Power conversion and distribution

As mentioned in Section 2.4, a significant amount of power is wasted in the power conversion and distribution infrastructure in the data center. Several technologies are currently being considered in order make this infrastructure more efficient. First, if the distribution voltage to racks can be raised from the current 110–220 V to 400–440 V, it will automatically reduce losses. However, there are safety and insulation issues with "high voltage data centers" that need to be worked out. Second, power conversion losses can be trimmed by reducing the number of times AC–DC conversion takes place in the data center. In particular, after an initial conversion to DC, all further conversion and distribution can stay DC. Both of these changes are rather radical and feasible only in new data centers.

A different approach is to make server and client power supplies more energy efficient and smarter. By some estimates there are currently more than 10 billion power supplies in use in the world [91]. Fig. 8 shows the efficiency of high efficiency and normal (low-efficiency) power supply units (PSUs) as a function of load. It is seen that at low loads, the PSU efficiency can be quite poor. Most servers in data centers run at rather low utilization and dual redundant power supplies are often used for reliability, thereby resulting in a rather low sustained load and efficiency. Furthermore, PSU inefficiency applies to the entire input power drawn by the server, which means wasted watts could be significant at all load levels. A secondary effect of PSU inefficiency is the heat generation that PSU fans need to remove.

It is clear that technologies that can maintain high PSU efficiency at all load levels are highly desirable. Phase shedding smart PSUs are one such solution. Such a power supply uses some number $N > 2$ of phases, of which only $n \leqslant N$ phases are active simultaneously. The change can be initiated either by the PSU itself based on the power drawn or via a change signal provided by the control software running either on the BMC or main CPU. On each change, the phases need to be rebalanced. For example, a 450 W power draw on a 6 phase power supply should have 60° separation between phases, with each phase delivering 75 W RMS power. Changing the number of phases is necessarily very slow since the rebalancing will require at least one cycle (16.7 ms at 60 Hz). Furthermore, the digital interface to PSU and power/temperature sensors tends to be a rather slow polling based interface. This brings in challenges in accurate control due to significant lag.

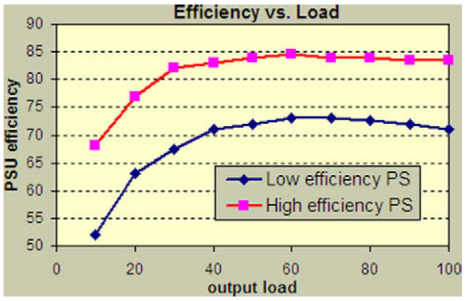

Fig. 8. Load vs. efficiency for power supplies.

In addition to PSU adjustments, server-side adjustments may also be needed. If the PSU detects a loss of input power, a quick notification to server/BMC can start a shutdown of inessential operations and devices to conserve power. In the extreme case where there is no UPS (but backup generators are available), the backup power can take 10–15 s to come on and it may be necessary to start shutdown by committing information to non-volatile storage. Even with UPS, it helps to reduce the required UPS capacity and in this case, servers need to take a variety of actions (e.g., shutting down most cores in a CPU or putting them in the lowest P state) in order to quickly match the drawn power to the installed UPS capacity. As sustainability and data center cost issues grow, good solutions to adapting drawn power to available power will become more important.

Next, we consider voltage regulators (VRs). Traditionally, voltage regulators are located on the motherboard and help cope with the proliferation of different voltages required by various components. VRs do DC to DC voltage conversion by modulating the input voltage with a duty cycle and passing it through a low pass filter. The process is often only about 85–90% efficient with efficiency dropping with the load as in power supplies. Phase shedding VRs have been proposed as a solution, but suffer from issues similar to those for power supplies (besides being expensive). A coordinated control of multiple phase shedding VRs can be quite challenging since different VRs have different time constants.

7.5. Cooling infrastructure

Traditional data center cooling infrastructure can be very expensive and consumes 25% or more of the total data center power. It also often does not work very efficiently. Evolving cooling technologies emphasize more localized cooling or try to simplify cooling infrastructure.

Chillerless, ambient or "free" cooling solutions do away with chiller plants which can be expensive, take up a lot of space, consume energy, and waste a lot of water in form of evaporation. These solutions are "open-loop" in that cooler ambient air is taken in and hot air is expelled from the building. Depending on the local weather, ambient cooling may result in higher temperature operation. Some large-scale studies indicate that ambient cooling can reduce cooling costs significantly without degrading reliability. For example, in a proof-of-concept data center operated by Intel, ambient cooling with 100% air exchange at up to 90 F was used without any humidity control and minimal air filtration. The result was 67% power savings using ambient cooling 91% of the time [84].

In order to compensate for the higher temperature of the air drawn from outside, ambient cooling generally needs to circulate a larger volume of air through the data center. This requires the design of larger and/or more fans which, in turn, consume more energy. This increase in energy expended towards circulating the larger volume of air must be carefully traded-off against the reduction due to getting rid of the CRAC unit. In order to ease movement of large volume of air (and thereby reduce energy consumption), rack occupancies can be reduced, but this increases the real-estate cost of the data center.

More localized cooling solutions may use cooling directly built into a rack. Such solutions can be much more efficient in that they place cooling next to the heat source as a vertical unit attachable to the side of a rack. Modular cooling may augment or replace the building or room level cooling. Modular solutions may also be assisted with temperature and humidity sensors along with variable-speed fans and humidity control. Modular solutions pose interesting load distribution problems – in particular, the optimal strategy is to concentrate all load on certain number of racks operating at some optimal cooling level whereas servers in other racks are placed in low-power mode or shut down and left without any cooling.

Moving down the spatial hierarchy, smarter cooling solutions have emerged at the chassis, enclosures, and server-level in the form of variable-speed fans modulated by temperature measurements. Server level fans may be augmented or replaced by shared chassis and/or rack-level fans. With decreasing size of servers and multi-core CPUs, the challenge of moving a sufficient amount of air through the server requires multiple small, high-velocity fans, which bring in noise issues. Also, modeling thermal and cooling behavior of tight enclosures with multiple heat generating elements and shared fans/air-flow becomes extremely difficult. Much of the current design methods depend upon empirical models of thermal coupling between various components based on typical air-flow patterns; more analytic design methods are required but can be very difficult.

7.6. Power/thermal management challenges

7.6.1. Measurement and metrics

Power measurement capabilities exist both within most modern servers and power distribution units (PDUs). To enable provisioning and dynamic control solutions that can modulate power consumption within different parts of the data centers in desirable ways, it is important to understand the dependency of power usage upon utilization of various resources within a data center. Another important issue is to predict the power needs of an aggregate such as a rack of servers. Some preliminary work on this relies on deriving probability distributions of the power and resource consumption of individual workloads using offline profiling and then using standard statistical techniques to aggregate them is contained in [74].

Temperature measurement facilities exist at various levels (starting from chip level) and rely on sensors of various degrees of sophistication. It is often possible to exploit the measurement time series to predict temperature events into the future and these can be used for control purposes. A number of recent efforts have devised models and simulation tools for capturing thermal effects ranging from chip-level [101], server, disks [80], rack [75], to room-level [99]. It would be desirable to develop integrated tools for modeling and prediction of thermal phenomena at these multiple spatial levels and their interactions. Additionally, models providing different tradeoffs between accuracy and computational overheads are also important.

The notion of *energy-proportionality* has been recognized as a desirable objective, which requires that some useful notion of performance scale linearly with the power consumed [73]. A true energy proportionality is clearly unachievable since most devices have significant idle mode power consumption and any transition to lower power mode incurs both the performance overhead (entry/exit latency) and power overhead (power consumption during entry and exit and). However, the concept can be useful in that it provides for idealized behavior to target. Server virtualization coupled with efficient migration techniques provides a way to consolidate workload on fewest number of servers so that the rest of them can be shut down. This provides one way of approximating the energy proportionality ideal across a sets of servers in the data center [105].

In order to better focus efforts on data center energy efficiency, it is important to develop meaningful metrics relating to cost, energy, cooling and performance. Currently the only widely used metric is PUE (power use efficiency) which measures the total power consumed by the data center divided by the IT power. Although useful, PUE is a rather crude metric. It is easy to define a PUE like metric at almost any level however, since efficiencies generally improve with load, the metric would be load dependent. Moreover, power efficiency without accounting for performance impact may not be very useful. Another popular metric, namely performance per watt attempts to capture performance-power tradeoff, but it too is fraught with problems since the meaning of "performance" is application dependent. A related problem is that characterizing the impact of power on performance is often very challenging and there is little in the way of formal models to address this gap [88]. Finally, in a virtualized environment, it should be possible to estimate power and thermal effects of individual VM's, but this can be very challenging since the VMs can interact in complex ways and the power consumptions don't simply add up. For example, a poorly behaved VM can increase the power consumption of other VMs.

7.6.2. Challenges in integrated control

Although the presence of myriad power knobs offered by various components offers opportunities for power savings and control, it also brings about significant management challenges. How should these knobs be manipulated in a coordinated fashion to achieve desirable aggregate behavior? Answering this question requires ways to model their inter-dependencies and reason about integrated control. A dual question to address is how many control knobs are desirable? From the last section, it is clear that, especially at the hardware component-level, the options for coordinated control are numerous in existing machines. What are the spatial and temporal granularities at which these knobs should operate and what are appropriate ways to partition functionality between hardware/firmware and various layers of software?

A variety of power and thermal control loops exist in data centers at different levels within the spatial hierarchy ranging from chip-level, component, server, rack, room-level to data center. The granularity of decision making ranges from less than a μs at the chip-level, seconds at server level, and minutes or hours at room-level. Often different control loops work on different objectives such as minimizing average power, peak power or temperature variations. These control loops may be designed independently and may conflict in their control strategies. The required coordination among the loops may need to abide by some specific rules as well. For example, the peak power control loop trying to enforce fuse limits should be given priority over the average power control loop trying to minimize energy consumption. These challenges have started to receive some attention [97,108], but significant additional challenges remain. In particular, different control loops may be designed by different vendors (HW/FW embedded loops vs. those operated by BMC vs. those in OS/middleware). An effective control requires well-defined interfaces through which the control loops can cooperate (instead of simply competing).

Since power/thermal effects are a property of a physical resource, power/thermal management is often based on the behavior of the physical resource directly. However, in a virtualized environment, a more intelligent management may be possible by considering the behavior of individual VM's sharing that physical resource. For example, the VM that consumes the most power or causes other VMs to consume more power because of its poor cache behavior may be subject to a tighter control than others. This can be quite challenging since in general it is very difficult to accurately attribute power and thermal effects to individual VMs. Reference [95] defines virtual power states on a per-VM basis which can be manipulated independently and the net effect mapped to the physical device. However, a good mapping scheme can be quite challenging in general.

7.6.3. Provisioning of power and cooling equipment

In order to cover the worst case situations, it is normal to over-provision systems at all levels of the power hierarchy, ranging from the power supplies within servers [90], Power Distribution Units (PDUs), Uninterrupted Power Supply (UPS) units, etc. Reference [76] estimates that over-provisioning in Google data centers to be about 40%. This over-estimation is driven by the use of "nameplate" power/thermal requirements of servers, which often assume that the server is not only provisioned with maximum possible physical resources such as DRAM, IO adapters and disks, but all these devices simultaneously operate close to their capacity. In practice, it is extremely rare to find workloads that can simultaneously stress more than two resources. For example, CPU bound workloads typically do very little IO and vice versa. Also, most servers typically run at much lower utilization than 100%.

Such over-provisioning of power increases data center setup and maintenance costs for power conversion/distribution infrastructure and cooling infrastructure at all levels. Since up to 3/4th of the power is simply wasted, it also increases the utility costs. Data centers are already beginning to discount name-plate power of servers by a certain percentage (often 25% or higher) in estimating the power distribution and cooling needs. However, a more accurate estimation of the "discounting" is very useful. Refs. [76,77] examine attempt to do this for hosted workloads in order to cost-effectively provision the power supply hierarchy. In order to address the drop in efficiency with load, Ref. [92] considers replacing high capacity PDUs with a larger number of smaller capacity PDUs.

As the provisioned power capacity moves closer to the average required capacity, there is a non-negligible probability that the provisioned capacity will occasionally prove to be inadequate. There are two ways to address such potential deficit: (a) local protection mechanisms, and (b) load migration. Local protection mechanisms attempt to cap power consumption in order to stay within the allowed budget at various levels (e.g., servers, racks, data center). This can be done by exploiting any of the active or inactive power modes – the main difference being that the adaptation is not workload driven, but rather driven by the limits. This requires a completely different set of algorithms and brings in unique problems of its own. For example, normally, the P-state switchover is triggered by the processor utilization (as in Intel's demand based switching or AMD's PowerNow schemes). In the power deficient situation, we may want to force CPU into P1 state just when the utilization shoots up to 100%. This can result in some undesirable behavior that needs to be managed properly [100]. Load migration refers to simply moving the workload (e.g., VM) to a different server and must account for migration power and latencies.

Uneven heat generation and cooling in a data center can be quite inefficient. Rack based cooling solutions discussed above help, but can be expensive and still suffer from imbalances within a rack. An alternate approach is to allow for smart cooling control by sensing hot-spots and adjusting air-flow appropriately [71]. A complementary technique is to balance the heat generation by properly balancing the load among active racks and servers. In this context, temperature aware workload placement [93], cooling aware scheduling [104] and dynamic load migration techniques [102] have been investigated in the literature. The location based services discussed in Section 6.6 can also be exploited for better power/thermal balancing as illustrated in [65]. However, significantly more work remains to be done on the issues of intelligent power distribution, power capping, thermal balancing and energy efficient cooling.

7.6.4. Power/thermal issues in DVDCs

The use of server, storage, and network virtualization raises several challenges in power and thermal management of data centers. In particular, if multiple virtualized data centers share the same underlying server infrastructure, it becomes difficult to clearly isolate power consumption and cooling costs associated with each virtualized data center. At lower levels, virtualization means that the power consumption associated with individual applications or services cannot be accurately measured or predicted thereby making it difficult to carefully track the energy efficiency of various applications or to charge for services based on energy consumption. While estimating direct energy consumption of executing a piece of code is hard enough, what is really required is an applications share of energy costs associated with all aspects of operating a data center including storage access, networking and even cooling.

New challenges related to coordination of IT and cooling control loops are arising due to the agility and flexibility offered by server virtualization. For example, exploiting virtualization to dynamically modulate the number of active servers based on utilization, may create thermal imbalances. This, in turn, may require more cooling and thereby actually increase the overall electricity consumption [94]. Therefore, the algorithm that decides the physical placement of virtual machines should also incorporate projections of the impact of its decision-making on power draw and temperature.

The VICL, as envisioned in Section 1 would allow creation of DVDC from resources spanning multiple physical data centers. Some significant challenges, in addition to the issues of accounting, would arise in such settings. Given that the price and availability of power could vary across the locations where the physical server patches are located, it would be interesting to examine the problem of constructing DVDCs so as to exploit this aspect in lowering energy costs. There is, of course, a temporal aspect to the problem as well since the energy prices and availability may vary according to overall demand and supply issues that are beginning to be exposed to customers by the utilities and may be forced by greater use of renewable energy. Similarly, the exposure of carbon footprint of various types of energy supply and carbon based pricing opens up prospects for more sophisticated considerations in designing and operating DVDCs.

8. Conclusions

In this paper we provided a detailed discussion of a variety of issues faced by data centers, the current state of affairs and a vision for the future. We also discussed a variety of challenges that need to be solved in the areas of data center storage, networking, management and power/thermal issues. It is hoped that the article will provide researchers many interesting avenues to explore in realizing the vision of highly scalable, well managed and energy efficient, distributed virtualized data centers of the future.

Acknowledgements

The author is grateful to Dr. Bhuvan Urgaonkar of Pennsylvania State University for his substantial contributions to the Storage and Power/Thermal sections of this paper. He also provided significant comments on other sections of the paper and helped with polishing them. Comments from anonymous reviewers were very helpful in improving the organization and presentation of the paper.

References

[1] C. Belady, In the Data Center, Power and Cooling Costs More Than The IT Equipment it Supports, ElectronicsCooling 13 (1) (2007).
[2] J. Zhen, Five Key Challenges of Enterprise Cloud Computing, <cloudcomputing.sys-con.com/node/659288>.

[3] F.J. Alfaro, J.L. Sanchez, J. Duato, QoS in InfiniBand Subnetworks, IEEE trans. on parallel & distributed systems 15 (9) (2004).
[4] D. Awduche, L. Berger, et al., RSVP-TE: Extensions to RSVP for LSP Tunnels, <http://www.ietf.org/rfc/rfc3209.txt>.
[5] P. Balaji, P. Shivam, P. Wyckoff, D. Panda, High performance user level sockets over Gigabit Ethernet, in: Proc. of IEEE Int. Conf. on Cluster Computing, September 2002 179–186.
[6] P. Balaji, H.V. Shah, D.K. Panda, Sockets vs RDMA interface over 10-Gigabit networks: an in-depth analysis of the memory traffic bottleneck", Proc. of RAIT Workshop, September 2004.
[7] D. Bergamasco, Ethernet Congestion Manager (ECM), See au-bergamasco-ethernet-congestion-manager-070313.pdf at <www.ieee802.org/1/files/public/docs2007>.
[8] A.L. Caro, J.R. Iyengar, et al., SCTP : A proposed standard for robust Internet data transport, IEEE Computer 36 (11) (2203) 20–27.
[9] M. Casado, M. Freedman, J. Pettit, et al., Ethane: Taking control of the enterprise, in: Proc. of SIGCOMM, August, 2007.
[10] S. Cho, R. Bettati, Aggregated aggressiveness control on groups of TCP flows, in: Proc. of Networking, 2005.
[11] P.J. Crowcroft, P. Oechslin, Differentiated end to end internet services using a weighted proportional fair sharing, Proc. of 1998 ACM SIGCOMM.
[12] P. Desnoyers, Empirical evaluation of NAND flash memory performance, in: Proc. of HotStorage, Big Sky, Montana, 2009.
[13] D. Dunning, G. Regnier, et al., The virtual interface architecture, IEEE Micro 18 (2) (1998) 66–76.
[14] H. Dwivedi, *Securing Storage: A Practical Guide to SAN and NAS Security*, Addison-Wesley, 2005.
[15] M. Al-Fares, A. Loukissas, A. Vahdat, A scalable, commodity data center network architecture, in: Proc. of 2008 ACM SIGCOMM.
[16] W. Feng, P. Balaji, et al., Performance characterization of a 10-gigabit ethernet TOE, Proc. of HPCA (2005) 58–63.
[17] F. Le Faucheur (Ed.), Multiprotocol label Switching (MPLS) Support of Differentiated Services, http://www.ietf.org/rfc/rfc3270.txt.
[18] R. Greenwald, R. Stakowiak, J. Stern, *Oracle Essentials*, fourth ed., O'Reilly, 2007.
[19] W. Huang, Q. Gao, J. Liu, D.K. Panda, High performance virtual machine migration with RDMA over modern interconnects, in: IEEE Intl. Conf. on Cluster Computing (Cluster'07), Austin, TX, September, 2007.
[20] G. Ibanex, A. Garcia, A. Azcorra, Alternative multiple spanning tree protocol (AMSTP) for optical ethernet backbones, in: Proc. of 29th IEEE Conference on Local Networks, 2004.
[21] H.-W. Jin, P. Balaji, et al., Exploiting NIC architectural support for enhancing IP based protocols on high performance networks, J. Parallel Distributed Comput., in press.
[22] K. Kant, TCP offload performance for front-end servers, in: Proc. of Globecom, 2003, San Francisco, CA.
[23] K. Kant, Towards a virtualized data center transport protocol, in: Proceedings of 2008 INFOCOM Workshop on High Speed Networks, Phoenix, AZ, April, 2008.
[24] K. Kant, Application centric autonomic BW control in utility computing, in: Sixth IEEE/ACM Workshop on Grid Computing, Seattle, WA, November, 2005.
[25] K. Kant, N. Jani, SCTP performance in data center environments, in: Proceedings of SPECTS, July 2005, Philadelphia, PA.
[26] K. Kant, Virtual link: an enabler of enterprise utility computing, in: Proceedings of International Symposium on Parallel Processing & Applications (ISPA), Sorrento, Italy, December, 2006.
[27] C.E. Leiserson, Fat-trees: universal networks for hardware-efficient supercomputing, IEEE Transactions on Computers 34 (10) (1985) 892–901.
[28] J. Liu, J. Wu, D.K. Panda, High performance RDMA-based MPI implementation over infiniband, International Journal of Parallel Programming 32 (3) (2004).
[29] J. Martin, A. Nilsson, I. Rhee, Delay based congestion avoidance in TCP, IEEE/ACM Transactions on Networking 11 (3) (2003) 356–369.
[30] QoS Support in MPLS networks, MPLS/Frame Relay alliance whitepaper, May, 2003.
[31] J. McDonough, Moving Standards to 100 GbE and beyond, IEEE applications and practice, November, 2007.
[32] R.N. Mysore, A. Pamboris, N. Farrington, et al., PortLand: a scalable fault-tolerant layer 2 data center network fabric, in: Proc. of 2009 ACM SIGCOMM.
[33] R. Noronha, X. Ouyang, D.K. Panda, Designing a high-performance clustered NAS: a case study with pNFS over RDMA on InfiniBand, in: Intl. Conf. on High Performance Computing (HiPC 08), December, 2008.
[34] C.B. Reardon, A.D. George, C.T. Cole, I. Ammasso, Comparative performance analysis of RDMA-enhanced Ethernet, in: Workshop on High-Performance Interconnects for Distributed Computing, 2005.
[35] G. Regnier, S. Makineni, et al., TCP onloading for data center servers, in: IEEE Computer, Special Issue on Internet Data Centers, November, 2004.
[36] S-A. Reinemo, T. Skeie, et al., An overview of QoS capabilities in InfiniBand, advanced switching interconnect, and ethernet, in: IEEE Communications Magazine, July, 2006, pp. 32–38.
[37] T. Shanley, *InfiniBand Network Architecture*, Mindshare Inc., 2002.
[38] Y. Tien, K. Xu, N. Ansari, TCP in wireless environments: problems and solutions, in: IEEE Radio Communications, March, 2005, pp. 27–32.
[39] J.P.G. Sterbenz, G.M. Parulkar, Axon: a high speed communication architecture for distributed applications, in: Proc. of IEEE INFOCOM, June, 1990, pp. 415–425.
[40] M. Wadekar, G. Hegde, et al., Proposal for Traffic Differentiation in Ethernet Networks, See new-wadekar-virtual%20-links-0305.pdf at <www.ieee802.org/1/files/public/docs2005>.
[41] J. Wang, K. Wright, K. Gopalan, XenLoop: a transparent high performance inter-vm network loopback, in: Proc. of 17th International Symposium on High Performance Distributed Computing, Boston, MA, June, 2008.
[42] M.A. Bender, M.F. Colton, B.C. Kuszmaul, Cache-oblivious string B-trees, in: Proceedings of PODS, 2006, pp. 233–224.
[43] P. Cappelletti, An overview of Flash Memories, <www.mdm.infm.it/Versatile/Essderc2007/9-00.pdf>.
[44] F. Chen, D. Koufaty, X. Zhang, Understanding intrinsic characteristics and system implications of flash memory based solid state drives, in: Proceedings of ACM Sigmetrics 2009, Seattle, June, 2009, pp. 181–192.
[45] R. Desikan, S. Keckler, D. Burger, Assessment of MRAM Technology Characteristics and Architectures, TR CS Dept., University of Texas at Austin, 2002.
[46] E. Gal, S. Toledo, Algorithms and data structures for flash memories, ACM Computing Survey 37 (2) (2005) 138–163.
[47] S. Gurumurthi, A. Sivasubramaniam, M. Kandemir, H. Franke, DRPM: dynamic speed control for power management in server class disks, in: Proceedings of ISCA 2003, pp. 169–179.
[48] Intel X25-E SATA Solid State Drive Product Manual, <http://download.intel.com/design/flash/nand/extreme/319984.pdf>.
[49] K. Kant, Role of Compression in Multi-level Memory Systems, <www.kkant.net/download.html>.
[50] H. Kohlstedt, Y. Mustafa, et al., Current status and challenges of ferroelectric memory devices, Microelectronic Engineering 80 (2005) 296–304.
[51] B.C. Lee, E. Ipek, O. Mutlu, D. Burger, Architecting phase change memory as a scalable DRAM alternative. Proc. of ISCA-2009.
[52] S. Lee and B. Moon, Design of Flash-based DBMS: An In-Page Logging Approach, Proc. of ACM SIGMOD, August, 2007, pp 55–66.
[53] A. Leventhal, Flash Storage Memory, Communications of the ACM (2008) 47–51.
[54] G. Muller, T. Happ, et al., Status and outlook of emerging nonvolatile memory technologies, IEEE Intl. Electron Devices Meeting (2004) 567–570.
[55] D. Narayanan, E. Thereska, A. Donnelly, et al., Migrating server storage to SSDs: analysis of tradeoffs, in: Proc. of 4th European conference on Computer systems (EuroSys 2009), Nuremberg, Germany, pp. 145–158.
[56] E. Pinheiro, R. Bianchini, C. Dubnicki, Exploiting redundancy to conserve energy in storage systems, in: Proc. of ACM SIGMETRICS, 2006, pp. 15–26.
[57] C.H. Wu, T.W. Kuo, L.P. Chang, An efficient B-tree layer implementation for flash-memory storage systems, ACM Trans. Embedded Computing Syst. 6 (3) (2007).
[58] Q. Zhu, Z. Chen, L. Tan, et al., Hibernator: helping disk arrays sleep through the winter, in: Proc. of SOSP, 2005, pp. 177–190.
[59] P. Zhou, Bo Zhao, et al., A durable and energy efficient main memory using phase change memory technology, in: Proceedings of ISCA-2009.
[60] R. Bitirgen, E. Ipek, J.F. Martinez, Coordinated management of multiple interacting resources in chip multiprocessors: a machine learning approach, in: Proceedings of 41st IEEE/ACM International Symposium on Microarchitecture, 2008, pp. 318–329.
[61] C. Brignone et al., Real time asset tracking in the data center, Distributed Parallel Databases 21 (2007) 145–165.

[62] Common Information Model. <www.wbemsolutions.com/tutorials/CIM/cim-specification.html>.
[63] E. Gelene, M. Xiaowen, R. Onvural, Bandwidth allocation and call admission control in high speed networks, IEEE Commun. Mag. (1997) 122–129.
[64] M. Kallahalla, M. Uysal, R. Swaminathan, et al., SoftUDC: a software-based data center for utility computing, IEEE Computer 37 (11) (2004) 38–46.
[65] K. Kant, N. Udar, R. Viswanathan, Enabling location based services in data centers, IEEE Network Mag. 22 (6) (2008) 20–25.
[66] P. Padala, K. Shin, X. Zhu, et al., Adaptive control of virtualized resources in utility computing environments, in: Proceedings of 2007 ACM SIGOPS/EuroSys Conference Lisbon, Portugal, pp. 289–302.
[67] J. Rao, C.-Z. Xu, CoSL: a coordinated statistical learning approach to measuring the capacity of multi-tier websites, in: Proceedings of International Conference on Parallel and Distributed Processing, April 2008, pp 1–12.
[68] N. Udar, K. Kant, R. Viswanathan, Asset localization in data center using WUSB radios, in: Proceedings of IFIP Networking, May 2008, pp. 756–767.
[69] KJ. Xu, M. Zhao, J. Fortes, et al., On the use of fuzzy modeling in virtualized data center management, in: Proceedings of International Conference on Autonomic Computing, June 2007, pp. 25–35.
[70] N. AbouGhazleh, D. Mosse, B.R. Childers, R. Melhem, Collaborative operating system and compiler power management for real-time applications, ACM Trans. Embedded Systems 5(1) 82–115.
[71] C.E. Bash, C.D. Patel, R.K. Sharma, Dynamic thermal management of air cooled data centers, in: Proceedings of 10th Conference on Thermal and Thermomechanical Phenomenon in Electronic Systems, June 2006, pp 452–459.
[72] F. Blanquicet, K. Christensen, An initial performance evaluation of rapid PHY Selection (RPS) for energy efficient ethernet, in: IEEE Conference on Local Computer Networks October 2007, pp. 223–225.
[73] L.A. Barroso, U. Holzle, The case for energy-proportional computing, IEEE Computer 40 (12) (2007) 33–37.
[74] J. Choi, S. Govindan, B. Urgaonkar, et al., Profiling, prediction, and capping of power consumption in consolidated environments, in: Proceedings of MASCOTS, 2008, pp. 3–12.
[75] J. Choi, Y. Kim, A. Sivasubramanium, et al., Modeling and managing thermal profiles of rack-mounted servers with thermostat, in: Proceedings of HPCA, 2007, pp. 205–215.
[76] X. Fan, W.-D. Weber, L.A. Barroso, Power provisioning for a warehouse-sized computer, in: Proceedings of 34th International Symposium on Computer Architecture (ISCA), 2007.
[77] S. Govindan, J. Choi, B. Urgaonkar, et al, Statistical profiling-based techniques for effective power provisioning in data centers, in: Proceedings of 4th European Conference on Computer systems (EuroSys 2009), Nuremberg, Germany, pp. 317–330.
[78] S. Graupner, V. Kotov, H. Trinks, Resource-sharing and service deployment in virtual data centers, in: Proceedings of ICDCS Workshop, July 2002, pp. 666–671.
[79] M. Gupta, S. Singh, Dynamic link shutdown for power conservation on ethernet links, in: Proceedings of IEEE International Conference on Communications, June 2007.
[80] S. Gurumurthi, Y. Kim, A. Sivasubramanium, Using STEAM for thermal simulation of storage systems, IEEE Micro 26 (4) (2006) 43–51.
[81] IEEE task group 802.3.az, Energy Efficienct Ethernet. <www.ieee802.org/3/az/public/nov07/hays_1_1107.pdf>.
[82] S. Harizopoulos, M.A. Shah, J. Meza, P. Ranganathan, Energy efficiency: the new holy grail of data management systems research, in: Conference on Innovative Data Systems Research, January 4–7, 2009.
[83] T. Heath, E. Pinheiro, et al., Application Transformations for Energy and Performance-Aware Device Management, in: Proceedings of 11th International Conference on Parallel Architectures and Compilation, 2002.
[84] <http://www.itbusinessedge.com/cm/community/news/inf/blog/intel-tests-free-cooling-in-the-data-center/?cs=20341>.
[85] C. Isci, A. Buyuktosunoglu, C.-Y. Cher, et. al., An analysis of efficient multi-core global power management policies: maximizing performance for a given power budget, in: Proceedings of IEEE Micro Conference, 2006.
[86] K. Kant, J. Alexander, Proactive vs. reactive idle power control, August 2008, <www.kkant.net/download.html>.
[87] K. Kant, Power control of high speed network interconnects in data centers, in: Proceedings of High Speed Networks Symposium at INFOCOM, 2009.
[88] K. Kant, Towards a science of power management, IEEE Computer 42 (9) (2009) 99–101.
[89] K. Kant, A control scheme for batching DRAM requests to improve power efficiency. <www.kkant.net/download.html>.
[90] C. Lefurgy, X. Wang, M. Ware, Server-level power control, in: Proceedings of International Conference on Autonomic Computing, 2007.
[91] Bob Mammano, Improving Power Supply Efficiency The Global Perspective, Texas Instruments report available at focus.ti.com/download/trng/docs/seminar/Topic1BM.pdf.
[92] D. Meisner, B.T. Gold, T.F. Wenisch, PowerNap: eliminating server idle power, in: Proc. of ASPLOS, 2009.
[93] J. Moore, J. Chase, P. Ranganathan, R. Sharma, Making scheduling cool: temperature-aware workload placement in data centers, in: Proc. of Usenix Annual Technical Conf., April, 2005.
[94] R. Nathuji, A. Somani, K. Schwan, Y. Joshi, CoolIT: Coordinating Facility and IT Management for Efficient Datacenters, HotPower, 2008.
[95] R. Nathuji, K. Schwan, VirtualPower: coordinated power management in virtualized enterprise systems, in: Proc. of SOSP 2007.
[96] A. Papathanasiou, M. Scott, Energy efficiency through burstiness, in: Proc of the 5th IEEE Workshop on Mobile Computing Systems and Applications (WMCSA'03), October, 2003, pp. 44–53.
[97] R. Raghavendra, P. Ranganathan, et al., No power struggles: coordinated multi-level power management for the data center, in: Proc. of 13th ASPLOS, March, 2008.
[98] V. Raghunathan, M.B. Srivastava, R.K. Gupta, A survey of techniques for energy efficient on-chip communication, in: Proc. the 40th Conference on Design Automation, 2003.
[99] L. Ramos, R. Bianchini, Predictive thermal management for data centers, in: Proc. of HPCA, 2008.
[100] P. Ranganathan, P. Leech, D. Irwin, J. Chase, Ensemble-level power management for dense blade servers, in: Proc. of ISCA 2006, pp. 66–77.
[101] T.D. Richardson, Y. Xie, Evaluation of thermal-aware design techniques for microprocessors, in: Proc. of Int. Conf. on ASICs, 2005, pp. 62–65.
[102] R.K. Sharma, C.E. Bash, C.D. Patel, et al., Balance of power: dynamic thermal management for internet data centers, IEEE Internet Computing 9 (1) (2005) 42–49.
[103] T.K. Tan, A. Raghunathan, N.K. Jha, Software architectural transformations: a new approach to low energy embedded software, in: A. Jerraya et al. (Eds.), Book Chapter in Embedded Software for SOC, Kluwer Academic Publishers, 2003, pp. 467–484.
[104] Q. Tang, S.K. Gupta, D. Stanzione, P. Cayton, Thermal-aware task scheduling to minimize energy usage of blade server based datacenters, in: Proc. of Second IEEE Symp. on Autonomic and Secure Computing, October, 2006, pp. 195–202.
[105] N. Tolia, Z. Wang, M. Marwah, et al., Delivering energy proportionality with non energy-proportional systems – optimizing the ensemble, in: Proc. of HotPower, 2008.
[106] V. Venkatachalam, M. Franz, Power reduction techniques for microprocessors, ACM computing surveys 37 (3) (2005) 195–237. <http://www.ics.uci.edu/vvenkata/finalpaper.pdf>.
[107] Y. Wang, K. Ma, X. Wang, Temperature-constrained power control for chip multiprocessors with online model estimation, in: Proc. of ISCA, 2009.
[108] X. Wang, and Y. Wang, Co-Con: Coordinated control of power and application performance for virtualized server clusters, Proc. of 17th IEEE Intl. workshop on QoS, Charleston, SC, July 2009.
[109] Q. Wu, M. Martonosi, et al., A dynamic compilation framework for controlling microprocessor energy and performance, in: Proc. of Int. Symp. on Microarhcitecture, Barcelona, 2005, pp. 271–282.

Krishna Kant has been with Intel Corp since 1997 where he has worked in a variety of research areas including traffic characterization, security/robustness in the Internet, data center networking, and power/thermal management of computer systems. From 1991 to 1997 he was with Telcordia Technologies (formerly Bellcore) where he worked on SS7 signaling and congestion control. Earlier, he spent 10 years in academia. He received his Ph.D. degree in Computer Science from University of Texas at Dallas in 1981.