J. Parallel Distrib. Comput. 74 (2014) 2561–2573

Contents lists available at ScienceDirect

J. Parallel Distrib. Comput.

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

Trends in big data analytics

Karthik Kambatla [a,*], Giorgos Kollias [b], Vipin Kumar [c], Ananth Grama [a]

[a] Department of Computer Science, Purdue University, United States
[b] IBM T. J. Watson Research Center, United States
[c] Department of Computer Science, University of Minnesota, United States

HIGHLIGHTS

- An overview of the state-of-the-art in big-data analytics.
- Trends in scale and application landscape of big-data analytics.
- Current and future trends in hardware that can help us in addressing the massive datasets.
- Discussion of software techniques currently employed and future trends to address the applications.

ARTICLE INFO

Article history:
Received 25 September 2013
Received in revised form
11 January 2014
Accepted 14 January 2014
Available online 2 February 2014

Keywords:
Big-data
Analytics
Data centers
Distributed systems

ABSTRACT

One of the major applications of future generation parallel and distributed systems is in big-data analytics. Data repositories for such applications currently exceed exabytes and are rapidly increasing in size. Beyond their sheer magnitude, these datasets and associated applications' considerations pose significant challenges for method and software development. Datasets are often distributed and their size and privacy considerations warrant distributed techniques. Data often resides on platforms with widely varying computational and network capabilities. Considerations of fault-tolerance, security, and access control are critical in many applications (Dean and Ghemawat, 2004; Apache hadoop). Analysis tasks often have hard deadlines, and data quality is a major concern in yet other applications. For most emerging applications, data-driven models and methods, capable of operating at scale, are as-yet unknown. Even when known methods can be scaled, validation of results is a major issue. Characteristics of hardware platforms and the software stack fundamentally impact data analytics. In this article, we provide an overview of the state-of-the-art and focus on emerging trends to highlight the hardware, software, and application landscape of big-data analytics.

1. Introduction

With the development of critical Internet technologies, the vision of computing as a utility took shape in the mid 1990s [19]. These early efforts on Grid computing [35] typically viewed hardware as the primary resource. Grid computing technologies focused on sharing, selection, and aggregation of a wide variety of geographically distributed resources. These resources included supercomputers, storage, and other devices for solving large-scale compute-intensive problems in science, engineering, and commerce. A key feature of these frameworks was their support for transparent domain-crossing administration and resource management capability.

The concept of 'data as a resource' was popularized by peer-to-peer systems [65]. Networks such as Napster, Gnutella, and BitTorrent allowed peer nodes to share content – typically multimedia data – directly among one another in a decentralized manner. These frameworks emphasized interoperability and dynamic, ad-hoc communication and collaboration for cost reduction, resource sharing, and aggregation. However, in many of these platforms, considerations of anonymity or privacy issues and scalability were secondary.

More recently, Cloud computing environments [95] target reliable, robust services, ubiquitously accessible (often through browsers) from clients ranging from mass-produced mobile devices to general purpose computers. Cloud computing generalizes prior notions of service from infrastructure-as-a-service (computing resources available in the cloud), and data-as-a-service (data

* Corresponding author.
E-mail addresses: kkambatl@cs.purdue.edu (K. Kambatla), gkollias@us.ibm.com (G. Kollias), kumar@cs.umn.edu (V. Kumar), ayg@cs.purdue.edu (A. Grama).

http://dx.doi.org/10.1016/j.jpdc.2014.01.003
0743-7315/

available in the cloud) to software-as-a-service (access to programs that execute in the cloud). This offers considerable benefits from points-of-view of service providers (cost reductions in hardware and administration), overall resource utilization, and better client interfaces. The compute back-end of cloud environments typically relies on efficient and resilient data center architectures, built on virtualized compute and storage technologies, efficiently abstracting commodity hardware components. Current data centers typically scale to tens of thousands of nodes and computations in the cloud often span multiple data centers.

The emerging landscape of cloud-based environments with distributed data-centers hosting large data repositories, while also providing the processing resources for analytics strongly motivates need for effective parallel/distributed algorithms. The underlying socio-economic benefits of big-data analytics and the diversity of application characteristics pose significant challenges. In the rest of this article, we highlight the scale and scope of data analytics problems. We describe commonly used hardware platforms for executing analytics applications, and associated considerations of storage, processing, networking, and energy. We then focus on the software substrates for applications, namely virtualization technologies, runtime systems/execution environments, and programming models. We conclude with a brief discussion of the diverse applications of data analytics, ranging from health and human welfare to computational modeling and simulation.

1.1. Scale and scope of data analytics

Recent conservative studies estimate that enterprise server systems in the world have processed 9.57×10^{21} bytes of data in 2008 [83]. This number is expected to have doubled every two years from that point. As an example, Walmart servers handle more than one million customer transactions every hour, and this information is inserted into databases that store more than 2.5 petabytes of data—the equivalent of 167 times the number of books in the Library of Congress [94]. The Large Hadron Collider at CERN will produce roughly 15 petabytes of data annually—enough to fill more than 1.7 million dual-layer DVDs per year [60]. Each day, Facebook operates on nearly 500 terabytes of user log data and several hundreds of terabytes of image data. Every minute, 100 h of video are uploaded on to YouTube and upwards of 135,000 h are watched [98]. Over 28,000 multi-media (MMS) messages are sent every second [3]. Roughly 46 million mobile apps were downloaded in 2012, each app collecting more data. Twitter [87] serves more than 550 million active users, who produce 9100 tweets every second. eBay systems process more than 100 petabytes of data every day [64]. In other domains, Boeing jet engines can produce 10 terabytes of operational information for every 30 min of operation. This corresponds to a few hundred terabytes of data for a single Atlantic crossing, which, if multiplied by the 25,000 flights each day, highlights the data footprint of sensor and machine-produced information.

These examples provide a small glimpse into the rapidly expanding ecosystem of diverse sources of massive datasets currently in existence. Data can be structured (e.g., financial, electronic medical records, government statistics), semi-structured (e.g., text, tweets, emails), unstructured (e.g., audio and video), and real-time (e.g., network traces, generic monitoring logs). All of these applications share the potential for providing invaluable insights, if organized and analyzed appropriately.

Applications requiring effective analyses of large datasets are widely recognized today. Such applications include health care analytics (e.g., personalized genomics), business process optimization, and social-network-based recommendations. However, projections suggest that data growth will largely outpace foreseeable improvements in the cost and density of storage technologies, the available computational power for processing it, and the associated energy footprint. For example, between 2002 and 2009 data traffic grew 56-fold, compared to a corresponding 16-fold increase in computing power (largely tracking Moore's law). In comparison, between 1998 and 2005 data centers grew in size by 173% per year [68]. Extrapolating these trends, it will take about 13 years for a 1000-fold increase in computational power (or theoretically 1000× more energy). However, energy efficiency is not expected to increase by a factor of over 25 over the same time period. This generates a severe mismatch of almost a 40-fold increase in the data analytics energy footprint [88,54].

Workload characteristics. A comprehensive study of big-data workloads can help understand their implications on hardware and software design. Inspired by the *seven dwarfs of numerical computation* [11], Mehul Shah et al. [82] attempt to define a set of "data dwarfs" — meaning key data processing kernels — that provide current and future coverage of data-centric workloads. Drawing from an extensive set of workloads, they establish a set of classifying dimensions (response time, access pattern, working set, data type, read vs write, processing complexity) and conclude that five workload models could satisfactorily cover data-centric workloads as of 2010: (i) distributed sort at petabytes scale, (ii) in-memory index search, (iii) recommendation system, featuring high processing load and regular communication patterns, (iv) sequential-access-based data de-duplication and (v) video uploading and streaming server at interactive response rates. While Online Analytic Processing (OLAP) workloads can readily be expressed as a combination of (i), (iii) and (iv), Online Transaction Processing (OLTP) workloads can only be partially captured and might require another category in the future; in-memory index and query support captures some facets of these workloads, but the working sets can become too large to fit in memory.

1.2. Design considerations

The scale, scope and nature (workload characteristics) of big-data analytics applications, individually, provide interesting insights into the design and architecture of future hardware and software systems.

Impact on hardware. The data access patterns and more specifically the frequency of how data is accessed (*cold* versus *hot* data) can drive future memory hierarchy optimizations: data generally starts being *hot*; however as time progresses, it becomes archival, *cold*, most suitable for storage in NVMs. However, there are notable exceptions of periodicity or churn in access patterns (season-related topics, celebrity headlines) and concurrently *hot* massive datasets (comparative genomic calculations) that should be taken into consideration. Furthermore, latent correlations among dimensions can arise with hardware stack projections: a single video, due to multiple formats or language subtitles, results in many versions. These could either be generated offline and stored (thus needing ample storage) or generated on the fly (transcoding and translation on-demand) putting pressure on the computing infrastructure of the data-center, or alternatively on the user's device (client-side computing). Alternatively, one might have to rethink the relative prioritization of advances in processor designs over the performance of the I/O subsystem—a common assumption in current architecture design. At the extreme of such an alternative, an option would be the consideration of a possible "inversion": a hierarchy of compute elements supporting the data store instead of today's designs of memory hierarchies to serve the compute element. Gradually collapsing existing storage hierarchies would smoothen such a transition and further provide savings in energy consumption.

Understanding the workloads could also identify opportunities for implementing special purpose processing elements directly

in hardware. GPUs, field-programmable gate arrays (FPGAs), specialized application-specific integrated circuits (ASICs), and dedicated video encoders/decoders warrant consideration. Such hardware accelerators drastically reduce the energy consumption, compared to their general-purpose processing counterparts. These could be integrated on-chip, leading to families of data-centric asymmetric multiprocessors [66].

Impact on software. Software systems, storage, and computing need to cater to a rather large problem space resulting from the data scale, nature of workloads, and other application requirements like consistency, availability, and partition tolerance.[1] Large data scales demand highly-scalable distributed storage systems that can accommodate and serve large volumes of data, with efficient ingress and egress mechanisms; Apache Flume [34] is one such system that facilitates data movement. Further, these storage systems should support in-memory caching for efficient querying and other OLTP workloads; even HDFS [44] recently added support for caching.

Varied workload characteristics demand a variety of compute engines—batch-processing for pure analytic workloads, stream-processing for more online processing of data, query-processing with transactional support.

Big-data analytics generally allows relaxed accuracy constraints on its quantitative output and this can influence aspects of algorithm design. Randomized algorithms project input data into their sketching approximations of reduced size before applying the original, expensive computing kernels and finally project back at the expense of provable bounds for accuracy loss (data size/accuracy tradeoff) [41]. Energy-aware computing kernels can reduce the energy footprint of the analytics calculations while retaining performance with minimal accuracy degradation (energy/accuracy tradeoff) [15].

2. Hardware platforms for data analytics

We now consider current hardware platforms for data analytics, and how these platforms are likely to evolve in the future.

2.1. Memory/storage

In traditional system designs, disks are used for persistent data storage and DRAM memory for faster access/disk caching purposes. However conventional disks have moving parts, largely constraining their lifetime and establishing bounds on their access times. DRAM chips, on the other hand, require static refreshing circuits consuming energy independent of whether there is data read/write activity or not. Non-volatile memory (NVM) technologies address these shortcomings and are expected to play major roles in future designs.

There are several NVM-based proposals for data-centric applications [75]. Flash-based NVMs can be exposed as block devices either through Serial Attached SCSI (SAS) and Serial Advanced Technology Attachment (SATA) or PCI express interfaces or even combined with dis-aggregated memory to provide cost-effective solutions [61]. Other types of NVM such as phase-change memory (PCRAM) and memristors have been proposed to be integrated as byte-addressable memory on the memory bus or stacked directly on the chip (3D-stacking) [73,59,74]. NVM could also be used to implement additional caching levels [96,84], or to replace the persistent store [27], thus facilitating the collapse of memory hierarchies—a single memory type to be used at various levels. The common denominator of these proposals is the steady shift to memory-like interfaces for persistent data storage for reasons of performance and simplification through homogeneity.

Moving computations closer to data is also motivated by the general structure of many of large-scale data management tasks, as subsequently espoused by related frameworks. For example in the frequently-used MapReduce paradigm, data is partitioned across available nodes and tasks are scheduled so that they are collocated with their data operands (to the extent possible). Collocation of tasks and their related data also results in important energy savings. The recent nanostore proposal [74] advocates the collocation of computation with the persistent data store. This is reminiscent of ideas from the past like Active Storage – enhanced disk controllers handling special computation tasks – however of limited application scope at the time of the original proposal [79].

From an evolutionary point of view, disks remain highly cost effective. Consequently, they are unlikely to be totally replaced any time soon. NVM-based technologies should be viewed as attractive components of near-term future designs. In this respect we should also take into account the change NVMs will trigger in the overall software stack. As an example, current file systems are optimized for latencies on the order of milliseconds. NVMs offer latency reduction of approximately three orders of magnitude (microseconds) over this time. There are proposals for using flash-based solid-state-disks (SSDs) to support key-value store abstractions, for workloads that favor it. Yet others propose to organize SSDs as caches for conventional disks (hybrid design). Ideally the persistence of NVMs should be exposed at the instruction-set level (ISA), so that the operating systems can utilize them efficiently (e.g., by redesigning parts that assume memory volatility or provide, to the upper layers, an API for placing archival data on energy-efficient NVM modules). On the other hand, the capability of durable memory writes reduces isolation; this issue could be addressed via durable memory transactions [93,24]. From the perspective of algorithm design and related data structures, non-volatility could push towards alternate, optimized designs and implementations of index structures, [22], key-value stores [91], database and file systems [27,12], all integral components of big-data analytics.

2.2. Processing landscape for data analytics

Chip multiprocessors (CMPs) are expected to be the computational work-horses for big data analytics. It seems however that there is no consensus on the specifics of the core ensemble hosted on a chip. Basically there are two dimensions of differentiation, namely the performance characteristics of each core and the degree of homogeneity of their ensemble.

Assembling chips out of low-power "wimpy" cores, interfacing small amounts of local flash storage, balances energy efficiency requirements with typical computation and I/O profiles in data-centric workloads [7]. The practicality of the approach (FAWN, Fast Array of Wimpy Nodes) was demonstrated by co-designing and building a datastore software system infrastructure aimed at high-rate, key-value lookup queries, achieving 330 queries per joule. This represents two orders of magnitude improvement over corresponding disk-based clusters. This approach is rooted in request-level parallelism of many of the target applications and their data I/O intensive traces best served by NVMs. For example, a dual in-line memory module with two DRAMs although faster than their flash-based counterparts can consume as much energy as a terabyte of disk (energy/performance tradeoff). Flash-based memories, on the other hand, currently cost an order of magnitude lower. However, they are about 100 times slower than DIMMs (performance/cost tradeoff).

[1] The CAP theorem [17] states that any distributed system needs to compromise at least one of consistency, availability and partition tolerance.

Contrary to this, studies [49] have argued that brawny cores still beat wimpy cores most of the time. The cost of additional software optimizations needed when moving to low-performance cores is emphasized, but more importantly, the role of Amdahl's law with the inherently serial part of a user's task putting a hard limit to the overall execution time. This is obviously exacerbated in the case of wimpy nodes. There are also concerns on sharing the cost of non-CPU infrastructure, the degradation of overall response time to parallelized requests with an increasing number of threads, lower utilization due to sub-optimal "fat" task allocation to "thin" cores (bin packing), and performance loss due to the conservative local heuristics employed for termination detection in some parallel runs.

Targeting the consolidation trend, common in virtualization designs, datacenter-on-chip (DoC) architectures have recently been proposed [55]. The authors identify four DoC usage models based on whether consolidated applications are homogeneous and cooperating or not. They identify key challenges related to scalability, addressed mainly by cache hierarchies and lack of performance isolation, since multiple virtual machines may contend for critical shared platform resources, mitigated by the application of QoS techniques.

An interesting line of future research for hardware improvements is presented by Tang et al. [86]. Garbage collection (GC) can take approximately 10% of the total execution time of a Java Virtual Machine (JVM) and can be implemented through different algorithms. After profiling and abstracting their common hotspots, these were implemented in hardware, achieving a 7% improvement on the total JVM execution time. This idea is expected to apply to general data-centric loads: after identifying shared hotspots, one could implement these in hardware, where possible, and thus provide hardware-assisted acceleration of bottleneck virtualization subtasks (in the same spirit as GC).

2.3. Network resources for data analytics

The link, network, and transport layers of standard communication stack were designed with interoperability of components of different technologies and manufacturers as a primary design consideration. In this design the link layer is not reliable, so congestion or unreliable communication channels can cause packet drops. To remedy this, the transport layer is then required to back off of transmission, thus compromising the bandwidth. However, a typical data center network environment is radically different from wide area networks [57]. To begin with, its channels can be considered lossless and this assumption should ideally reflect in the flow mechanisms it implements. It is mostly homogeneous and under a single administrative control, so backward compatibility to already deployed protocols is not an issue. Load balancers and application proxies separate internal traffic from external so there are no fairness concerns with conventional TCP. Round trip times (RTTs) can be less than 250 microseconds, in the absence of queuing. Applications concurrently need extremely high bandwidths and very low latencies. Also, there is little statistical multiplexing, so a single flow can dominate a particular path.

A particular performance bottleneck in data center networks is caused by the extensive use of partition/aggregate design pattern in many large scale web applications: requests from higher layers of the application are broken into pieces and farmed out to workers at lower layers; the responses of these workers are then aggregated to produce a result. It follows that data transmitted back will traverse a bottleneck link in a many-to-one fashion. As the number of concurrent senders increases, the application-level throughput at the receiver collapses to orders of magnitude lower than the link capacity (incast pattern problem [23]). For example, this can happen in MapReduce jobs during the "shuffle" stage, when intermediate key-value pairs from many mappers are transferred to appropriate reducers. A recent variant of TCP, Data Center TCP (DCTCP) [6], addresses this artifact of the fork-join structure of network traffic, by leveraging the ECN protocol extension implemented in most modern commodity switches, allowing end-to-end notification of network congestion without dropping packets. Conversely, future performance problems could drive the customization of switching equipment.

Note that the details of dealing with particular network shortcomings are fundamentally related to the protocol chosen. The basic contenders are Ethernet and Infiniband. Infiniband is an energy-proportional network and this property definitely gives this technology an edge, since energy efficiency is a primary design objective for future data centers. It is expected that both options will be available in the imminent future and selecting between the two will be application-based.

With respect to interconnect technologies, optical and electrical–optical designs offer significant advantages. However, since electrical-to-optical conversions represent energy efficiency bottlenecks, the goal is to use an all-optical switching fabric. This transition will be paved by first eliminating the network interface controller (NIC), so the processor will talk directly to the network; also processor-to-memory paths can be made optical. The recent announcement of the first parallel optical transceiver prototype to transfer one terabit per second by IBM clearly identifies big data analytics as a target market and emphasizes the energy efficiency of the device [52]. Intel also plans to introduce a cost-effective optical interconnect cable in later versions of Thunderbolt interface (already available over copper wire in Apple products). As an example of a hybrid electrical/optical switch, Helios [32] is an architecture promising significant reductions in the number of switching elements, cabling, cost, and power consumption. It is a two-level multi-rooted tree of pod switches and core switches, where the core consists of both traditional electrical packet switches and MEMS-based optical circuit switches. The reason for including electrical packet switches is justified by the need to handle burstiness in aggregated traffic demand between different pairs of pods [39,5]: the number of (optical) circuits required to support this type of traffic would be prohibitive, and electrical packet switching would be desirable.

2.4. Energy considerations in big-data analytics

Energy proportionality is a key objective in the design of cloud systems and components [10]. Most data centers operate at less than half the peak load, although their efficiency is maximized at peak loads. Energy proportionality implies a linear relation between power consumption and load. This can be achieved by explicitly using energy proportional devices, or powering only the absolutely necessary components in the course of a computation. More specifically, work consolidation is usually applied to coarse-grained components like individual server machines. Nevertheless, the applicability of the approach heavily depends on the type of computation: batch processing yields desirable power reductions but online or parallel applications do not permit such an energy saving scheme. Note that energy proportionality is not a nominal feature of optical networking technologies (static laser power consumption).

Energy optimizations however are correlated with system scale; it is easier to save power in larger deployments, especially in the energy pipeline prior to feeding the electronic devices processing, moving, or storing the data. Cooling and power delivery are, in most cases, amenable to energy savvy design and implementation. André et al. [10] factor a datacenter's efficiency into three terms—a facility term, a server energy conversion term, and the efficiency of the electronic components in performing the

computation itself. Although the third term is most relevant to our discussion, the first two factors account for an unsatisfactory 2.2 Watts of energy per "productive" Watt, on average. In a more holistic approach to optimizing the third term, Baliga et al. [14] consider optimizing energy consumption in global cloud systems by first analyzing the energy consumption profiles for transport, storage, and servers in a variety of scenarios drawn from storage as a service, software as a service, and processing as a service facets of cloud computing.

At a finer-grain, techniques such as dynamic voltage and frequency scaling (DVFS), shutting down functional units selectively, have been proposed to reduce CPU power consumption (which accounts for roughly one third of the energy to the hardware subsystem) [46]. However, static power dissipation and performance requirements impose basic limits. Energy considerations have motivated proposals for asymmetric designs (specialized chip multiprocessors, CMPs); Hardavellas et al. [42] populate the dice with a large, diverse array of application-specific heterogeneous cores that can achieve peak performance and energy efficiency by dynamically disabling all but the most application specific cores (resulting in dark silicon spots).

3. Virtualization technologies

Virtualization is a key concept underlying Cloud deployments reconciling the natural divide between software and hardware. Its basic component is the virtual machine monitor (VMM), a software-abstraction layer enabling the partitioning of the underlying hardware platform into one or more virtual machines [80]. Most importantly (i) existing software should run unmodified within each of the virtual machines, (ii) a "statistically" dominant subset of the instructions must be executing directly on the CPU and (iii) VMM has complete control over the system resources (classical virtualization) [71].

Virtualization is a technique initially developed in the context of mainframes in the late 1960s. With the proliferation of cost-effective machines that followed, the need for resource sharing that dictated virtualization, seized to exist, to the extent that modern hardware does not inherently support this mode. However, the advantages it provides, especially in terms of reliability, security and administration—hardware becomes a pool of resources to run arbitrary services on demand—make its use for data-centric, multi-tenancy environments of big data analytics highly desirable. Specifically, VMMs offer encapsulation of a virtual machine's state, thus facilitating the tasks of load balancing, virtual machine replication, storage and transport, suspend/resume scenarios, hardware and software/configuration failure handling. Additionally, VMMs provide strong isolation between virtual machines, so multiplexing of multiple virtual machines over the same hardware becomes transparent, thus yielding dramatic cost benefits. Security and reliability advantages follow naturally since a malfunctioning program either due to bugs or security compromises is isolated in its virtual machine, problems do not propagate to machines executing under the same VMM.

All hardware subsystems (CPU, memory, I/O, and network) could, in principle, be virtualized. Currently, CPU virtualization is relatively mature, followed by interesting improvements, innovations and research proposals for virtualizing memory management unit (MMU), I/O subsystems, and networks.

A CPU architecture is virtualizable if it supports the basic VMM technique of direct execution: the virtual machine executes on the real machine; however the VMM has the ultimate control of the CPU. This is typically implemented by running the virtual machine's privileged and unprivileged code in CPU's unprivileged mode and reserving its privileged mode for the VMM; when the virtual machine attempts to perform a privileged operation the CPU traps into the VMM, which in turn emulates it by updating the state of the virtual machine. However, the ubiquitous x86 architecture does not provide safe and transparent trap semantics for all its privileged operations. Consequently, various techniques have been proposed. In para-virtualization, the operating system (OS) executing in the virtual machine is patched to replace non-virtualizable operations with suitably engineered, virtualization-friendly equivalents. However, altering the source code of an operating system can be problematic due to licensing issues, and it potentially introduces incompatibilities. In an alternate approach, a binary translator runs the non-virtualizable, privileged parts and dynamically patches the "offending" instructions, also retaining in a trace cache the translated blocks for optimization purposes.

For memory management, VMM maintains a shadow of each virtual machine's memory-management data structure, its shadow page table. VMM updates these structures reflecting operating system's changes and establishes the mapping to actual pages in the hardware memory. Challenges here include enabling the VMM to leverage the operating system's internal state for efficient paging in/out and sharing identical physical pages across multiple virtual machines monitored by a single VMM. This sharing will be particularly important for homogeneous pools (in terms of software configuration) of virtual machines executing, over multicore multiprocessors on-a-chip, the workloads of big data analytics in the future.

I/O virtualization, at least for x86-based architectures, would demand the direct inclusion of code that talks to each of the devices currently in existence into the VMM layer. One solution to this is the hosted architecture: VMM runs as an application atop a host OS and essentially forwards any I/O requests from the guest OS. The hosted architecture might not scale in the server environments with the high-performance network and disk subsystems. Consequently, a VMM executing over bare-metal, reusing certified open source device drivers would be the solution of choice.

The obvious advancement in addressing is to turn to hardware-assisted virtualization [33]. This is already happening—for example, Intel VT-x and AMD-V technologies provide new execution modes for the processor that lets a VMM safely and transparently use direct execution for running virtual machines; the number of related traps and the time to serve them are also reduced. For memory virtualization, extended page tables and virtual-processor identifier tagged (VPID) TLBs are proposed. I/O proposals range between hardwired tables for specifying access control between devices and memory pages to DMA remapping and interrupt virtualizations to supporting virtual functions, each giving an illusion of an entirely separate physical device (a capability already existing in Infiniband); the evolution in this front will also be largely shaped by the rate of transition to channel-like I/O devices like USB and SCSI, simplifying developments.

Virtualization is a broad concept, not limited to system virtualization, the topic of the previous discussion. For big data analytics, high-level language virtualization is a related concept. Recent times have witnessed the broad adoption of virtual machines (like JVM or CLR) as compilation targets for many programming languages. The obvious advantage is the immediate sharing of optimizations in the virtual machine and libraries across an extensive set of languages. Interestingly, these optimizations include the introduction of new (software) ISA opcodes [8], mainly to serve the need of high-productivity languages ported to these runtimes, e.g., dynamically typed scripting languages or the use of just-in-time (JIT) techniques, respectively, reminiscent of hardware-assisted virtualization and binary translation mentioned above. Considering the fact that programmer's productivity in developing future big-data-analytics software is an important cost factor, advances in this direction are important.

4. Software stack for analytics applications

In addition to hardware enhancements, big-data analytics on yotta-scale datasets require a complete re-evaluation of the software stack. Improved software solutions should (i) scale to accommodate large datasets, (ii) efficiently leverage the hardware platforms, and (iii) bridge the increasing gap between the growth of data and computing power [88]. While storage systems must evolve to host these large-scale datasets, data-processing systems need to efficiently process data on these datastores. The use of datacenters with very large clusters of commodity hardware for big-data analytics and the advent of cloud computing have already revolutionized software. Distributed software running in these datacenters must tolerate hardware failures, scale with the amount of data, and be able to elastically leverage additional resources as and when they are provided.

In this section, we discuss the recent advances in both storage and computing fronts, and outline future research directions adopted by several research groups.

4.1. Storage systems

Big-data storage is essential to any form of analytics. The amount of data and the advances in hardware require storage platforms to be distributed, scalable, elastic and fault-tolerant. In addition to these highly-desirable features, applications have their respective demands of the underlying storage. Client-facing applications demand high-availability, even in the presence of node failures or network partitions. Depending on the required amounts of fault-tolerance and availability, current storage systems appropriately replicate data across multiple machines within and across datacenters. Replication involves the overhead of preserving consistency across the replicas; this overhead increases with the number of replicas. Even though archival systems are efficient for storing large amounts of data, random access or the need to preserve meta-data (tags) requires more sophisticated storage models. Big-data is often unstructured and does not fit the strict relational model. This has motivated NoSQL distributed datastores—multi-column key-value stores.

4.1.1. Effects of CAP theorem

An ideal storage platform supports efficient data accesses in the presence of failures (node failures and network partitions), high-availability, and offers a consistent view of data to its clients. However, Brewer, through his famous CAP theorem [17], showed that such an ideal system cannot exist, as it is impossible to guarantee consistency along with high-availability in the presence of partitions. Consequently, distributed storage systems are forced to relax at least one of these constraints.

Applications often drive the design of the underlying storage systems. For instance, if an application does not require strong consistency guarantees, the underlying storage system can provide high-availability in the presence of failures. E.g. Amazon's Dynamo [30] is a classic example here—the *shopping cart* application should be highly-available but can tolerate weaker consistency; Dynamo is a highly-available key-value store that backs this application and offers eventual consistency guarantees.

Future big-data applications might lead to similar innovations in storage. On the other hand, for applications that require all three features, their constituent kernels might be more accommodating; this can result in using multiple storage platforms to serve a single application. Here, we examine a few alternatives and discuss future research directions.

Availability. Most cloud applications that do not directly interface with the client, typically, do not need high-availability; all back-end processes fall under this category of applications. By sacrificing high-availability, the storage platforms that serve these kinds of applications promise strong consistency in the presence of node failures and network partitions. The Google File System (GFS) [36] is a highly-scalable and consistent distributed file system. Google's Bigtable [21] and Megastore [13] are key-value stores that replicate data within and across data-centers respectively. Google's storage systems use Chubby [18] locking service to synchronize accesses to shared resources. The open-source equivalents of these systems, HDFS [44][2] and HBase [43], use Zookeeper [51] for consistency; Zookeeper uses the Zab protocol [78] to propagate the incremental state updates from primary node to backup nodes. These storage systems tolerate node/disk failures through duplication mechanisms—replication, coding (e.g. erasure coding), *etc.*

Consistency. Client-facing applications typically relax consistency guarantees to realize high-availability and performance requirements. While the strong consistency model guarantees that all clients see the same data, weaker consistency models relax this requirement. The weakest model of no consistency, where each replica has a different version of the data without any reconciliation, is less useful. Eventual consistency promises that all replicas would eventually reach a consistent state; note that reconciliation of the different states across replicas can take arbitrarily long depending on the work load, node-failures, and network partitions. Eventual consistency is sufficient for applications dealing with non-sensitive information. For instance Amazon uses Dynamo [30] for its shopping cart, and Facebook stores user information (posts, messages and pictures) in Cassandra [58].

Not all applications requiring high-availability can afford eventual consistency; popular examples include storage of electronic medical records and bank accounts. Inconsistencies in such sensitive information can lead to severe problems. Recent efforts have proposed alternate consistency models. PNUTS [28] proposes per-record consistency, where all replicas apply updates to a row in the same order. Though this is a stronger guarantee than eventual consistency, it does not fully support serializability. COPS [62], scalable causal consistency model, enforces causal dependencies between keys across the entire cluster; all operations on the datastore are ordered and hence inconsistencies due to out-of-order execution are avoided. Microsoft's Windows Azure Storage (WAS) [20] promises a highly-available storage service with strong consistency within a storage stamp; a storage stamp can host only a limited amount of data (100 TB). WAS achieves all three characteristics within a storage stamp by decoupling the nodes that promise high availability in the presence of partitions from those that guarantee strong consistency. This decoupling is achieved by adopting a two-layered model where the lower layer addresses network partitioning and the upper layer ensures consistency. Decoupling of nodes and the use of two-layers, however, entail performance overheads associated with a higher number of replicas and inter-layer communication. Birman et al. [16] propose a consistency model for soft-state replication that does not require durability. They achieve order-based consistency through in-memory logging of updates, resulting in strongly consistent model albeit with weak durability. They argue that the unavoidable delays the CAP theorem prescribes stem from the durability aspect of C in CAP; hence, by relaxing the durability guarantees, they achieve strong consistency.

[2] Recently, HDFS added redundancy in the master through a pair of Active–Standby masters. If the Active master fails, the Standby takes over the role of master. However, there is a failover period during which the system is unusable.

4.1.2. Resource utilization

The performance and resource-usage of above-mentioned distributed storage platforms are as important as the features of high-availability, consistency and tolerance to partitions. The increasing gap between data size and compute power makes a case for improved resource-usage. While users tend to ignore slow- or ill-performing applications, increased resource-usage can lead to prohibitive energy cost. For instance, high-availability translates to more replicas, entailing storage overhead for extra replicas, per-update network overhead, and the energy overhead associated with storage and transfer of data.

Recent research efforts have attempted to improve the performance and resource-usage of these storage platforms. Techniques such as erasure-coding help reduce the size of the data transferred to replicas and stored. Even though erasure-coding adds a coding/decoding cost, the space reduction reduces network latency and disk space. In fact, Google's next generation file system Colossus uses Reed–Solomon encoding to reduce its storage footprint. The advent of SSDs also greatly reduces the disk access latencies, but comes with a different failure model—the wear-leveling in flash storage devices requires checking for storage errors through checksums. Erasure-coding techniques would automatically correct disk errors as well. Network usage within a datacenter can be improved using techniques like TCP Nice [90] and Dr. Multicast [92]. TCP Nice improves network usage by using spare bandwidth for asynchronous, background communication to avoid interfering regular demand traffic. Dr. Multicast enables IP-multicast in datacenters to reduce the sender/receiver latencies. By mapping traditional IP-multicast operations to either use UDP-multisend or send to traditional IP-multicast address, Dr. Multicast makes multicast in datacenters manageable by limiting the number of IP-multicast groups. With user-clouds spanning multiple datacenters, there is a need for similar or better techniques for inter-datacenter communication.

4.2. Data processing considerations

Big-data analytics applications differ in the kind of input, data access patterns and the kind of parallelism they exhibit. Applications with online (streaming) input process each input/request individually incurring significant latency costs, while those with large datasets as inputs can batch I/O and avoid these latencies. Client-facing applications (e.g., querying) randomly access the underlying storage, while back-end processes that run on entire datasets have a more sequential access pattern. While most web-applications exhibit data-parallelism, scientific applications often exhibit task parallelism. Even among data-parallel applications, some use iterative algorithms with each iteration operating on the same data. Different applications require different data-processing techniques and optimizations. However, all the models used for big-data analytics in datacenters need to be fault-tolerant, scale with data, and elastically utilize additional resources.

4.2.1. Data-parallel models

Typical analytics applications are data-parallel, involving computations on independent data-items. This data-parallelism can be extracted using the simple SPMD technique; the single operation is applied to each individual data-item potentially in parallel. The data can be distributed across different compute nodes to be operated on concurrently. Here, we consider three different models, depending on the kind of input.

Batch processing. Batch-processing applies to processing large datasets, where (I/O) operations on multiple data-items can be batched for efficiency. In the context of big-data analytics, Google's MapReduce [29] is the first major data-processing paradigm. Dean and Ghemawat proposed MapReduce [29] to facilitate development of highly-scalable, fault-tolerant, large-scale distributed applications. The MapReduce runtime system divests programmers of low-level details of scheduling, load balancing, and fault tolerance. The map phase of a MapReduce job takes as input a list of key-value pairs, $\langle key, value\rangle{:}list$, and applies a programmer-specified (map) function, independently, on each pair in the list. The output of the map phase is a list of keys and their associated value lists – $\langle key, value{:}list\rangle{:}list$, referred to as intermediate data. The reduce phase of the MapReduce job takes this intermediate data as input and applies another programmer-specified (reduce) function on each pair of key and value list. The MapReduce runtime supports fault-tolerance through a deterministic replay mechanism, where a failed map/reduce task is simply re-executed. The MapReduce programming model and its open-source version Hadoop [40] have been widely adopted in the big-data analytics community for their simplicity and ease-of-programming. Hadoop is expected to touch half of the world's data by 2015 [50].

A number of research efforts have targeted improving both the systems and applications aspects of MapReduce. The MapReduce programming model has been validated on diverse application domains like data-analytics and data-mining. Pig [67] offers a high-level SQL like language for easier analysis of large-scale data, while HadoopDB [2] builds a distributed database by using Hadoop for (storage and analysis) distribution over single-node databases. While MapReduce restricts the data-flow to map and reduce phases in favor of simplicity, Dryad [53] supports a more general data-flow model expressed as directed acyclic graphs for more advanced developers. To extend the applicability of MapReduce, MapReduce Online [25] supports online aggregation and continuous queries, while Kambatla et al. [56] extend MapReduce to support asynchronous algorithms efficiently through relaxed synchronization semantics. Other performance optimizations target performance in different environments [47,76].

Bulk synchronous parallel processing. Even though MapReduce applies to broad classes of batch-processing applications, it may not be the optimal paradigm in every case. For instance, iterative algorithms (e.g., graph algorithms) operate on the same input (e.g., graph) in each iteration. Having to reload data in every iteration from distributed storage is expensive and unnecessary. In these cases, bulk-synchronous parallel processing (BSP) [89] of the data works well. In the BSP model, computation proceeds in supersteps (iterations); in each iteration, concurrent computations are executed in parallel on participating nodes (cluster-nodes), followed by a global synchronization step where tasks communicate if necessary. Each computation operates on the data local to that node, the input can be cached at various levels and does not need reloading.

Adopting this BSP model, Google proposed Pregel [63] for iterative graph algorithms. Pregel holds the entire graph in memory distributed across nodes, thus avoiding disk-access latencies. The primary overhead is the communication at the end of each superstep, which is essential to application semantics. The open source implementations of Pregel—Java-based implementations GoldenOrb [38], Apache Giraph [37], and the Erlang-based implementation Phoebus [69]—are being increasingly used in analyzing social networks (LinkedIn, Facebook, etc.) and other scientific applications.

Pregel and related models store entire graphs in memory for improved performance. This is likely not feasible in the future, given the anticipated growth in data. Consequently, techniques for efficiently caching and processing graphs need to be developed. One approach loads smaller partitions into memory and performs local communications in memory itself. This requires effective partitioning techniques; most existing partitioning techniques

cannot be readily used for large-scale data in a distributed setting on graphs with specific characteristics (for e.g., power law).

Event processing. Event-driven applications require continuous and timely processing of events; the tight latency constraints disallow batching events together for efficiency. Event-driven applications typically involve processing, monitoring, and notification of events. These applications, involving complex event processing, read as input a stream of events and either (i) independently process individual data items (stream-processing) or (ii) detect complex event patterns (e.g., credit-card fraud detection). While operations on data are independent in stream-processing applications, detecting complex events requires maintaining a state corresponding to past events; i.e., stream-processing is context-free whereas complex event detection is context-sensitive. Prior research (Aurora [1], Borealis [4], Publish/Subscribe systems [31]) addresses both stream-processing and complex event detection in depth, albeit for different input rates and hardware.

Stream-processing systems operate on continuous data streams: e.g., click streams on web pages, user request/query streams, monitoring events, notifications, etc. The importance of stream-processing systems increases as more modern applications impose tighter time constraints on a particular event's propagation along the pipeline. For instance, Google, Facebook and other advertising companies analyze user-data (e.g., wall post or status change) to display specific ads corresponding to the user's search query. As they aim reduce the time it takes for ads related to the particular event to be displayed, more data would go through the stream-processing pipelines.

The progress of stream-processing systems, in the cloud (datacenters), has been relatively slow. Efforts [26] have gone into supporting streaming-inputs in batch-oriented systems like MapReduce, but as expected, suffer from high latencies. Though several concepts used in previous stream-processing systems like Borealis [4] and IBM's System S [9] still hold, they lack support for elasticity and hence do not readily lend themselves to the utility computing model of the cloud. Furthermore, the cost model of the cloud is very different from the fixed-size distributed clusters these systems were proposed for. Recently, systems like Apache S4 [81] and Twitter's Storm [85] have been designed particularly for use in datacenters to operate on big-data. However, these systems have limited support for elasticity and dynamic load-balancing. In addition to this, the fault-tolerance mechanisms used by all the above-mentioned systems require dedicated backup nodes leading to significant resource costs in the cloud.

In order to elastically use additional resources, computing nodes need to dynamically off-load some of their work to the newly added nodes. Such load-balancing requires the migration of PEs (processing elements—compute elements of a stream-processing engine) across compute nodes. PE migration can also be used to tolerate node-failures, by migrating the PEs running on the failed node to other nodes. Indeed, existing stream-processing systems [4] tolerate failures by migrating PEs from the failed node to a dedicated stand-by node. The migration is typically achieved through active or passive stand-by. The active stand-by approach duplicates work on the dedicated backup node, thus using twice the amount of resources. The passive stand-by approach employs checkpointing the internal state and downstream messages to reconstruct state in case of failures. Recovery involves reading the checkpoint and replaying the messages upstream, thus leading to relatively long recovery times.

Detecting complex events, on the other hand, is a different problem; it involves pattern matching on event content. Traditionally, content-based publish/subscribe systems were used for this. Publish/subscribe systems for emerging datacenter architectures continue to be actively investigated. The main challenge here is to integrate the pub/sub system with other storage and computation platforms, particularly their fault-tolerance mechanisms. Apache HedWig [45], the state-of-the-art in this space, offers guaranteed-delivery, high-availability, incremental scalability and supports topic-based matching on a limited number of topics (10^6) and subscriptions (10 per topic). Clearly, this is not enough to serve future anticipated workloads.

4.2.2. Data-dependent parallelism

The application scope of data-centric (data-parallel) programming models can be significantly improved by allowing communication/ data-sharing across concurrent computations. Data-sharing through shared address space (e.g., a shared disk-resident key-value store) enables speculation and task-parallelism. Speculative-parallelism (or amorphous data-parallelism) [70] is where parallel execution of tasks can lead to potentially conflicting concurrent computations. Though non-conflicting computations can be executed in parallel, these conflicts can only be detected at runtime. Exploiting this form of parallelism requires communication across computations to detect and resolve potential conflicts.

Traditionally, message-passing/shared-memory models have been used to solve these problems as they allow explicit communication. Running elastic, scalable and fault-tolerant equivalent of MPI in the cloud would suffice; however, it is not trivial to build such equivalents and active work is going on in the area. Apache Yarn and other resource-scheduling efforts (Mesos [48]) might help in defining resource abstractions which can be used to realize this objective. Meanwhile, researchers have considered communicating over distributed key-value stores as shared address-spaces. Piccolo [72] allows distributed computations to store mutable state through the abstraction of a distributed in-memory key-value table. TransMR [77] proposes transactional execution of computations over key-value stores; map and reduce tasks are transactionally executed over Bigtable. The conflicts are detected when conflicting transactions are validated and re-executed. While Piccolo suffers from memory overheads, TransMR involves not-so-scalable distributed transactions.

4.3. Integration of different models

As application requirements and underlying platforms change, modern applications must often use multiple programming models in their workflow. Stream-processing (counting) of click streams followed by batch-processing (analytics: e.g., correlation of clicks) is one such example. There is a need for tighter integration of the programming models not only with each other, but also with the underlying storage. For instance, an efficient pub/sub system can be built by co-locating a stream-processing system over a distributed key-value store; the subscriptions can be stored in a table (partitioned on topic) distributed across the cluster-nodes, and the published events can be sent to those nodes which have the related subscriptions where topic- and content-based matching can be performed.

Apache YARN (Yet Another Resource Negotiator) [97] is one step closer to achieving the first goal; it enables running multiple programming models on the same cluster by scheduling resources across models. Much work needs to be done to support an efficient and usable composite programming model in support of emerging applications.

5. Application scope of emerging big data analytics

The push towards collecting and analyzing large amounts of data in diverse application domains is motivated by a variety of factors:

- In complex systems that do not lend themselves to intuitive models, data-driven modeling and hypothesis generation is key to understanding system behavior and interactions. Such applications arise in natural environments (the sciences), social interactions, and engineered systems, among others.
- In commercial and business enterprises, the role of big-data analytics is well recognized in enhancing efficiencies and guiding decision processes.
- In health and human welfare, big-data analytics offer tremendous potential for prognostic interventions, novel therapies, and in shaping lifestyle and behavior. Big-data analytics is also key to cost efficiencies and sustainability of the healthcare infrastructure.
- In interactive or client-oriented environments, big-data analytics guide the systems interface with the clients. Examples of businesses shaping their operational environment to optimize client experience (and business outcomes) are well understood. Ubiquitous environments such as smart homes are now emerging, simultaneously optimizing for living spaces as well as energy footprint and cost.
- In complex systems that lend themselves to shaping and control, analytics enable controlled evolution and design.

The aforementioned application areas are, by no means, exhaustive. Rather, they provide a sense of the vast scope and impact of big-data analytics. In the rest of this section, we highlight some of the important application domains in greater detail, highlighting the tremendous socio-economic impact. In particular, we discuss how the applications are likely to evolve in the future and how they will fundamentally shape our environment.

5.1. Health and human welfare

One of the most compelling applications of big-data analytics is in the domain of health and human welfare. In addition to the self-evident benefits in terms of enhancing well-being, this area also presents some of the largest and fastest growing datasets. While it is difficult to estimate current size and growth rates, by some estimates, the global size of clinical data stands at roughly 150 Exabytes in 2011, increasing at a rate between 1.2 and 2.4 Exabytes per year.[3] Clinical data primarily corresponds to electronic medical records (EMRs) and imaging data. This data is rapidly increasing both in terms of size of records and coverage in population. EMRs will contain personal genomic data, data from other high-throughput screens, be linked to personalized drug–response profiles.

In addition to clinical data, healthcare data also includes pharmaceutical data (drug molecules and structures, drug targets, other biomolecular data, high-throughput screening data (microarrays, mass spec, sequencers), and clinical trials), data on personal practices and preferences (including dietary habits, exercise patterns, environmental factors), and financial/activity records. Effectively integrating all of this data holds the key to significant improvements in interventions, delivery, and well-being.

The cost-benefits of big-data analytics in the healthcare domain are also well-acknowledged. A recent study by the McKinsey Global Institute[4] estimates that healthcare analytics could create more than $300 billion in value every year. Similar cost efficiencies could also be realized in Europe (estimated at $149 billion).

Data in healthcare poses some of the most challenging problems to large-scale integration and analysis. Beyond its sheer volume and heterogeneity, virtually all data is collected at point-of-care, and is stored in widely distributed repositories with varying access capacities. Data consumers (care providers, analytics with expert-in-the-loop) are often not collocated with either point-of-care or data repositories. For example, imaging data (MRI, fMRI) is often accessed overseas by trained radiologists to render expert opinion and diagnoses. Even if data interchange formats are fully standardized, the multimodal nature of data poses challenges for integration. While data volumes in healthcare are currently dominated by imaging data, it is conceivable that in the foreseeable future, personalized genomics and high-throughput screens would influence data analysis. Several analysis tasks are also time-critical. For example, patient diagnoses, progression of outbreaks, etc., all have tight performance requirements.

Issues of data quality, privacy and security, and effectiveness of analysis are critical in healthcare informatics. Clinical data handling must be constrained not just be established laws and practices, but also by subjects' expectations of privacy. Pharmaceutical data represents valuable intellectual property, and its use (even in a collaborative environment) is highly guarded.

Given its significance and technical complexity, there have been significant investments in healthcare IT. With literally hundreds of millions of computing devices from implanted devices, patient sensors, in-home care devices, mobile devices, and coarser compute elements higher in the abstraction, healthcare IT represents one of the largest integrated distributed environments, and its impact is likely to increase sharply in the imminent future.

5.2. Nature and natural processes

One of the pressing questions of our times relates to the changes to our habitat and its long-term impact on our environment. A wealth of data is being collected relating to our environmental footprint and its observable impact. On one hand, we can now see, at high spatio-temporal resolution, events such as deforestation and urban encroachment. At the same time, we can also monitor retreating polar ice caps, glaciers, and extreme weather events. This data is typically collected from satellite imagery, weather radar, and terrestrial monitoring and sensing devices. Recent efforts have also targeted collection of data on the carbon footprint of key sources. These datasets typically reside in larger data centers, offering relatively high-throughput access.

In contrast to data in healthcare IT, analytics on these datasets pose a different set of challenges. Considerations of security and privacy are not as critical, although several datasets are considered proprietary or sensitive. The main challenges here derive from the spatio-temporal scales, the magnitude of data, and the difficulties associated with validation of long-term predictions based on models. The scale of data makes it difficult to migrate; consequently, novel distributed analyses are being deployed for such applications. Going forward, one may expect a deeper integration of various datasets (land use, deforestation, carbon emissions, terrestrial and satellite-based imaging, and computational modeling), to comprehensively study global climate change and to assign specific causality.

Other related problems include natural resource management, including land- and water-resources management, sustainable development, and environmental impact assessment. Data for these analyses come from diverse sources—including sensors monitoring environmental state, human activity (manufacturing, economic output), and extraneous factors. While such analyses is in relative infancy, it is likely that such models will be critical in sustainability.

[3] http://blogs.sas.com/content/hls/2011/10/21/how-big-is-big-data-in-healthcare/.

[4] http://www.mckinsey.com/Insights/MGI/Research/Technology_and_Innovation/Big_data_The_next_frontier_for_innovation.

5.3. Government and the public sector

Government and the public sector are primary motivators for big-data analytics. The Cloud First initiative, with the goal of closing approximately 1,000 government data centers by 2015 and move 79 services to the cloud, as well as the more recent (May 23, 2012) directive entitled "Building a 21st Century Digital Government" issued by the President of USA clearly demonstrate decisive steps towards addressing critical data handling and analytics needs.[5] A number of US Federal agencies, including the Department of Treasury, have moved their public websites to Amazon's Elastic Cloud (EC2). Commercial services have been created, such as AWS GovCloud, which specifically target moving intensive workloads to the cloud.[6] USA.gov, US government's official Web portal – the go-to destination for anything related to the government, run by the General Services Administration (GSA) – is another notable example of this trend.[7] These projects have drastically reduced operational costs, upgrade time, and downtime.[8]

Significant new opportunities are on the horizon. In a world where mobile broadband subscriptions are expected to grow from nearly 1 billion in 2011 to over 5 billion globally in 2016, and with the projection that by 2015, more Americans will access the Internet via mobile devices than desktop machines, new models of governance, such as mobile electronic voting will become viable. The convenience of such models of governance motivates stronger involvement of the public in government decisions and processes. Fraud detection is another significant analytics application. According to the Federal Bureau of Investigation (FBI), just health-care fraud costs U.S. taxpayers over $60B each year. Automated means of identifying such fraudulent transactions and tax offenses hold the potential for significant financial efficiencies. Accurate projections on the work–force dynamics, based on the inferred, detailed profiles of the professional activity of the population, can drive appropriate policy decisions and measures. The use and management of public resources such as transportation networks and fixed infrastructure could be optimized by exploiting continually sensed data relating to usage patterns. This is particularly relevant in times of emergency planning and preparedness.

5.4. Commerce, business, and economic systems

Perhaps, the most visible application of big-data analytics has been in business enterprises. It is estimated that a retailer fully utilizing the power of analytics can increase its operating margin by 60%. Utilizing new opportunities (for e.g., location-aware and location-based services) leads to significant potential for new revenues. A comprehensive analytics framework would require integration of supply chain management, customer management, after-sales support, advertising, etc. Business enterprises collect vast amounts of multi-modal data, including customer transactions, inventory management, store-based video feeds, advertising and customer relations, customer preferences and sentiments, sales management infrastructure, and financial data, among others. The aggregate of all data for large retailers is easily estimated to be in the exabytes, currently. With comprehensive deployment of RFIDs to track inventory, links to suppliers databases, integration with customer preferences and profiles (through store loyalty programs), and fully integrated financial systems, the potential for improved efficiencies is tremendous and is only just starting to be realized.

Datasets in such applications are relatively well structured and integrated. Since these analyses typically operate in closed systems (i.e., much of the data, infrastructure, and analyses is performed within the same security domain), issues of privacy and security in analyses are easier to handle. Data quality is not a major concern and resources are relatively easily available in state-of-the-art data centers. The major bottleneck in this domain is the development of novel analytics methods that scale to vast amounts of multimodal data.

5.5. Social networking and the internet

"When a 5.9 Richter earthquake hit near Richmond, VA, on August 23rd, 2011, residents in New York City read about the quake on Twitter feeds 30 s before they experienced the quake themselves". This extract from the recent US presidential directive for digital government vividly describes the speed and impact of information flow in social networks. Drawing parallels to the evolution of search engines – one of the most successful big-data analytics applications to date – one may expect a significant increase in the personalization and real-time delivery of content to social network users. This calls for effective, event-based algorithmic constructs, featuring asynchronous approximations to information diffusion in smaller (sparsified), versions of dynamically evolving link structures. Analyzing interactions as functions of time and other attributes helps understand the emergence of patterns of collective behaviors, enabling shaping of information flows, resource management, and prediction. Summarization of link representations for marking substructures or nodes of interest present targets for online marketing strategies.

The Internet itself serves as the infrastructure of a semantic web. Since most of its information is currently unstructured, there is a significant motivation for organizing it into structured forms to infer related concepts and relations, in order to automate reasoning. Text search and indexing technologies are relatively mature; however novel graph mining techniques are in relative infancy. Robust, offline, automatic image and video indexing and searching capabilities provide new dimensions in the search for entity correlations, putting additional strain on big-data analytics infrastructure.

5.6. Computational and experimental processes

As computing platforms push the envelope to the exascale, simulations can be performed at scales and resolutions unimaginable in even the recent past. From quantum-mechanical modeling to astro-physical simulations, these model span spatial scales from atoms to galaxies, and timescales from femtoseconds to eons. Other experiments, such as the Large Hadron Collider, generate large amounts of data from detectors sensing emissions from high speed collision of sub-atomic particles. In each case, the exascale datasets generated must be analyzed for generating and validating scientific insights.

On emerging platforms, there is a need to inline analysis with simulation/data acquisition. This is motivated by the fact that often, data storage rates cannot match data generation. In yet other applications/platforms, analyses must scale to large computing platforms. Data is often available at one location, emphasizing parallel analytics, as opposed to distributed techniques. Based on the impending advances in processing, networking, and I/O technologies, it is likely that datasets will change qualitatively and quantitatively in the near future.

[5] http://www.whitehouse.gov/sites/default/files/omb/egov/digital-government/digital-government.html.

[6] http://aws.amazon.com/govcloud-us/.

[7] http://www.wired.com/insights/2012/08/5-coolest-gov-cloud-projects/.

[8] http://www.frost.com/prod/servlet/cio/232651119.

6. Conclusion

The volume of data operated upon by modern applications is growing at a tremendous rate, posing intriguing challenges for parallel and distributed computing platforms. These challenges range from building storage systems that can accommodate these large datasets to collecting data from vastly geographically distributed sources into storage systems to running a diverse set of computations on data. Resource and semantic constraints, like Brewer's CAP theorem, require handling these problems on a per-application basis, exploiting application-specific characteristics and heuristics. Recent efforts towards addressing these challenges have resulted in scalable distributed storage systems (file systems, key-value stores, etc.) and execution engines that can handle a variety of computing paradigms. In the future, as the data sizes continue to grow and the domains of these applications diverge, these systems will need to adapt to leverage application-specific optimizations. To tackle the highly distributed nature of data sources, future systems might offload some of the computation to the sources itself to avoid the expensive data movement costs.

Recent hardware advances have played a major role in realizing the distributed software platforms needed for big-data analytics. Future hardware innovations — in processor technology, newer kinds of memory/storage or hierarchies, network architecture (software-defined networks) — will continue to drive software innovations. Strong emphasis in design of these systems will be on minimizing the time spent in moving the data from storage to the processor or between storage/compute nodes in a distributed setting.

References

[1] Daniel J. Abadi, Don Carney, Ugur Çetintemel, Mitch Cherniack, Christian Convey, Sangdon Lee, Michael Stonebraker, Nesime Tatbul, Stan Zdonik, Aurora: a new model and architecture for data stream management, VLDB J. 12 (2) (2003) 120–139.

[2] Azza Abouzeid, Kamil Bajda-Pawlikowski, Daniel Abadi, Alexander Rasin, Avi Silberschatz, HadoopDB: an architectural hybrid of MapReduce and DBMS technologies for analytical workloads, in: VLDB, 2009.

[3] http://agbeat.com/tech-news/how-carriers-gather-track-and-sell-your-private-data/.

[4] Yanif Ahmad, Bradley Berg, Uğur Cetintemel, Mark Humphrey, Jeong-Hyon Hwang, Anjali Jhingran, Anurag Maskey, Olga Papaemmanouil, Alexander Rasin, Nesime Tatbul, Wenjuan Xing, Ying Xing, Stan Zdonik, Distributed operation in the borealis stream processing engine, in: Proceedings of the 2005 ACM SIGMOD International Conference on Management of Data, SIGMOD'05, ACM, New York, NY, USA, 2005, pp. 882–884.

[5] Mohammad Al-Fares, Alexander Loukissas, Amin Vahdat, A scalable, commodity data center network architecture, in: Victor Bahl, David Wetherall, Stefan Savage, Ion Stoica (Eds.), SIGCOMM, ACM, 2008, pp. 63–74.

[6] Mohammad Alizadeh, Albert G. Greenberg, David A. Maltz, Jitendra Padhye, Parveen Patel, Balaji Prabhakar, Sudipta Sengupta, Murari Sridharan, Data center TCP (DCTCP), in: Shivkumar Kalyanaraman, Venkata N. Padmanabhan, K.K. Ramakrishnan, Rajeev Shorey, Geoffrey M. Voelker (Eds.), SIGCOMM, ACM, 2010, pp. 63–74.

[7] David G. Andersen, Jason Franklin, Michael Kaminsky, Amar Phanishayee, Lawrence Tan, Vijay Vasudevan, FAWN: a fast array of wimpy nodes, Commun. ACM 54 (7) (2011) 101–109.

[8] Rasmus Andersen, Brian Vinter, The scientific byte code virtual machine, in: GCA, 2008, pp. 175–181.

[9] H. Andrade, B. Gedik, K.L. Wu, P.S. Yu, Processing high data rate streams in system S, J. Parallel Distrib. Comput. 71 (2) (2011) 145–156.

[10] Luiz André Barroso, Urs Hölzle, The Datacenter as a Computer: An Introduction to the Design of Warehouse-Scale Machines, in: Synthesis Lectures on Computer Architecture, Morgan & Claypool Publishers, 2009.

[11] Krste Asanovic, Ras Bodik, Bryan Christopher Catanzaro, Joseph James Gebis, Parry Husbands, Kurt Keutzer, David A. Patterson, William Lester Plishker, John Shalf, Samuel Webb Williams, Katherine A. Yelick, The Landscape of Parallel Computing Research: A View from Berkeley. Technical Report UCB/EECS-2006-183, EECS Department, University of California, Berkeley, 2006.

[12] M. Athanassoulis, A. Ailamaki, S. Chen, P. Gibbons, R. Stoica, Flash in a DBMS: where and how? Bull. IEEE Comput. Soc. Tech. Committee Data Eng. (2010).

[13] Jason Baker, Chris Bond, James C. Corbett, J.J. Furman, Andrey Khorlin, James Larson, Jean-Michel Leon, Yawei Li, Alexander Lloyd, Vadim Yushprakh, Megastore: Providing Scalable, Highly Available Storage for Interactive Services, in: CIDR'11, 2011.

[14] J. Baliga, R.W.A. Ayre, K. Hinton, R.S. Tucker, Green cloud computing: balancing energy in processing, storage, and transport, Proc. IEEE 99 (1) (2011) 149–167.

[15] Costas Bekas, Alessandro Curioni, A new energy aware performance metric, Comput. Sci.-Res. Dev. 25 (3–4) (2010) 187–195.

[16] K. Birman, D. Freedman, Qi Huang, P. Dowell, Overcoming cap with consistent soft-state replication, Computer 45 (2) (2012) 50–58.

[17] E.A. Brewer, Towards robust distributed systems, in: Proc. 19th Annual ACM Symposium on Priniciples of Distributed Computing, PODC, 2000, pp. 7–10.

[18] Mike Burrows, The chubby lock service for loosely-coupled distributed systems, in: Proceedings of the 7th Symposium on Operating Systems Design and Implementation, OSDI'06, USENIX Association, Berkeley, CA, USA, 2006, pp. 335–350.

[19] Rajkumar Buyya, Chee Shin Yeo, Srikumar Venugopal, James Broberg, Ivona Brandic, Cloud computing and emerging IT platforms: vision, hype, and reality for delivering computing as the 5th utility, Future Gener. Comput. Syst. 25 (6) (2009) 599–616.

[20] Brad Calder, Ju Wang, Aaron Ogus, Niranjan Nilakantan, Arild Skjolsvold, Sam McKelvie, Yikang Xu, Shashwat Srivastav, Jiesheng Wu, Huseyin Simitci, Jaidev Haridas, Chakravarthy Uddaraju, Hemal Khatri, Andrew Edwards, Vaman Bedekar, Shane Mainali, Rafay Abbasi, Arpit Agarwal, Mian Fahim ul Haq, Muhammad Ikram ul Haq, Deepali Bhardwaj, Sowmya Dayanand, Anitha Adusumilli, Marvin McNett, Sriram Sankaran, Kavitha Manivannan, Leonidas Rigas, Windows azure storage: a highly available cloud storage service with strong consistency, in: Proceedings of the Twenty-Third ACM Symposium on Operating Systems Principles, SOSP'11, ACM, New York, NY, USA, 2011, pp. 143–157.

[21] Fay Chang, Jeffrey Dean, Sanjay Ghemawat, Wilson C. Hsieh, Deborah A. Wallach, Mike Burrows, Tushar Chandra, Andrew Fikes, Robert E. Gruber, Bigtable: a distributed storage system for structured data, in: OSDI, 2006.

[22] Shimin Chen, Phillip B. Gibbons, Suman Nath, Rethinking database algorithms for phase change memory, in: CIDR, 2011, pp. 21–31. www.crdrdb.org.

[23] Yanpei Chen, Rean Griffith, Junda Liu, Randy H. Katz, Anthony D. Joseph, Understanding TCP incast throughput collapse in datacenter networks, in: Jeffrey C. Mogul (Ed.), WREN, ACM, 2009, pp. 73–82.

[24] Joel Coburn, Adrian M. Caulfield, Ameen Akel, Laura M. Grupp, Rajesh K. Gupta, Ranjit Jhala, Steven Swanson, NV-heaps: making persistent objects fast and safe with next-generation, non-volatile memories, in: Proceedings of the 16th International Conference on Architectural Support for Programming Languages and Operating Systems, 16th ASPLOS'11, ACM Press, Newport Beach, CA, USA, 2011, pp. 105–118.

[25] Tyson Condie, Neil Conway, Peter Alvaro, Joseph Hellerstein, Khaled Elmeleegy, Russell Sears, MapReduce online, in: NSDI, 2009.

[26] Tyson Condie, Neil Conway, Peter Alvaro, Joseph M. Hellerstein, Khaled Elmeleegy, Russell Sears, MapReduce online, in: NSDI, 2010.

[27] Jeremy Condit, Edmund B. Nightingale, Christopher Frost, Engin Ipek, Benjamin C. Lee, Doug Burger, Derrick Coetzee, Better I/O through byte-addressable, persistent memory, in: Proceedings of the 22nd Symposium on Operating Systems Principles (22nd SOSP'09), Operating Systems Review (OSR), ACM SIGOPS, Big Sky, MT, 2009, pp. 133–146.

[28] Brian F. Cooper, Raghu Ramakrishnan, Utkarsh Srivastava, Adam Silberstein, Philip Bohannon, Hans-Arno Jacobsen, Nick Puz, Daniel Weaver, Ramana Yerneni, Pnuts: Yahoo!'s hosted data serving platform, Proc. VLDB Endow. 1 (2) (2008) 1277–1288.

[29] Jeffrey Dean, Sanjay Ghemawat, MapReduce: simplified data processing on large clusters, in: OSDI, 2004.

[30] Giuseppe DeCandia, Deniz Hastorun, Madan Jampani, Gunavardhan Kakulapati, Avinash Lakshman, Alex Pilchin, Swaminathan Sivasubramanian, Peter Vosshall, Werner Vogels, Dynamo: Amazon's highly available key-value store, in: Proceedings of Twenty-First ACM SIGOPS Symposium on Operating Systems Principles, SOSP'07, ACM, New York, NY, USA, 2007, pp. 205–220.

[31] Patrick Th. Eugster, Pascal A. Felber, Rachid Guerraoui, Anne-Marie Kermarrec, The many faces of publish/subscribe, ACM Comput. Surv. 35 (2) (2003) 114–131.

[32] Nathan Farrington, George Porter, Sivasankar Radhakrishnan, Hamid Hajabdolali Bazzaz, Vikram Subramanya, Yeshaiahu Fainman, George Papen, Amin Vahdat, Helios: a hybrid electrical/optical switch architecture for modular data centers, in: Shivkumar Kalyanaraman, Venkata N. Padmanabhan, K.K. Ramakrishnan, Rajeev Shorey, Geoffrey M. Voelker (Eds.), SIGCOMM, ACM, 2010, pp. 339–350.

[33] J. Fisher-Ogden, Hardware support for efficient virtualization.

[34] Apache flume, http://flume.apache.org.

[35] I. Foster, C. Kesselman (Eds.), The Grid: Blueprint for a New Computing Infrastructure, Morgan-Kaufmann, 1999.

[36] Sanjay Ghemawat, Howard Gobioff, Shun-Tak Leung, The google file system, in: Proceedings of the Nineteenth ACM Symposium on Operating Systems Principles, SOSP'03, ACM, New York, NY, USA, 2003, pp. 29–43.

[37] Apache giraph, http://incubator.apache.org/giraph.

[38] Goldenorb, http://www.raveldata.com/goldenorb.

[39] Chuanxiong Guo, Guohan Lu, Dan Li, Haitao Wu, Xuan Zhang, Yunfeng Shi, Chen Tian, Yongguang Zhang, Songwu Lu, BCube: a high performance, server-centric network architecture for modular data centers, in: Pablo Rodriguez, Ernst W. Biersack, Konstantina Papagiannaki, Luigi Rizzo (Eds.), SIGCOMM, ACM, 2009, pp. 63–74.

[40] Apache hadoop, http://hadoop.apache.org.

[41] Nathan Halko, Per-Gunnar Martinsson, Joel A. Tropp, Finding structure with randomness: probabilistic algorithms for constructing approximate matrix decompositions, SIAM Rev. 53 (2) (2011) 217–288.

[42] Nikos Hardavellas, Michael Ferdman, Babak Falsafi, Anastasia Ailamaki, Toward dark silicon in servers, IEEE Micro 31 (4) (2011) 6–15.

[43] Apache hbase, http://hadoop.apache.org/hbase.

[44] Apache hadoop hdfs, http://hadoop.apache.org/hdfs.

[45] Apache hedwig, https://cwiki.apache.org/BOOKKEEPER/hedwig.html.
[46] Andrew Herdrich, Ramesh Illikkal, Ravi R. Iyer, Donald Newell, Vineet Chadha, Jaideep Moses, Rate-based QoS techniques for cache/memory in CMP platforms, in: Proceedings of the 23rd International Conference on Supercomputing, 23rd ICS'09, ACM Press. Intel, Yorktown Heights, NY, USA, 2009, pp. 479–488.
[47] Improving MapReduce Performance in Heterogeneous Environments, USENIX Association, San Diego, CA, 2008, 12/2008.
[48] Benjamin Hindman, Andy Konwinski, Matei Zaharia, Ali Ghodsi, Anthony D. Joseph, Randy Katz, Scott Shenker, Ion Stoica, Mesos: a platform for fine-grained resource sharing in the data center, in: Proceedings of the 8th USENIX Conference on Networked Systems Design and Implementation, NSDI'11, USENIX Association, Berkeley, CA, USA, 2011, p. 22.
[49] U. Hölzle, Brawny cores still beat wimpy cores, most of the time, IEEE Micro 30 (4) (2010).
[50] Hortonworks blog, http://hortonworks.com/blog/executive-video-series-the-hortonworks-vision-for-apache-hadoop.
[51] Patrick Hunt, Mahadev Konar, Flavio P. Junqueira, Benjamin Reed, Zookeeper: wait-free coordination for internet-scale systems, in: Proceedings of the 2010 USENIX Conference on USENIX Annual Technical Conference, USENIXATC'10, USENIX Association, Berkeley, CA, USA, 2010, p. 11.
[52] Made in IBM labs: holey optochip first to transfer one trillion bits of information per second using the power of light, 2012. http://www-03.ibm.com/press/us/en/pressrelease/37095.wss.
[53] Michael Isard, Mihai Budiu, Yuan Yu, Andrew Birrell, Dennis Fetterly, Dryad: distributed data-parallel programs from sequential building blocks, in: EuroSys, 2007.
[54] The International Technology Roadmap for Semiconductors, 2010, http://www.itrs.net/.
[55] R. Iyer, R. Illikkal, L. Zhao, S. Makineni, D. Newell, J. Moses, P. Apparao, Datacenter-on-chip architectures: tera-scale opportunities and challenges, Intel Tech. J. 11 (3) (2007) 227–238.
[56] Karthik Kambatla, Naresh Rapolu, Suresh Jagannathan, Ananth Grama, Asynchronous algorithms in MapReduce, in: IEEE International Conference on Cluster Computing, CLUSTER, 2010.
[57] S. Kandula, S. Sengupta, A. Greenberg, P. Patel, R. Chaiken, The nature of data center traffic: measurements & analysis, in: Proceedings of the 9th ACM SIGCOMM Conference on Internet Measurement Conference, ACM, 2009, pp. 202–208.
[58] Avinash Lakshman, Prashant Malik, Cassandra: a structured storage system on a p2p network, in: SPAA, 2009.
[59] Benjamin C. Lee, Ping Zhou, Jun Yang, Youtao Zhang, Bo Zhao, Engin Ipek, Onur Mutlu, Doug Burger, Phase-change technology and the future of main memory, IEEE Micro 30 (1) (2010) 143.
[60] http://public.web.cern.ch/public/en/LHC/Computing-en.html.
[61] Kevin Lim, Jichuan Chang, Trevor Mudge, Parthasarathy Ranganathan, Steven K. Reinhardt, Thomas F. Wenisch, Disaggregated memory for expansion and sharing in blade servers, in: Proc. 36th International Symposium on Computer Architecture, 36th ISCA'09, ACM SIGARCH, Austin, TX, 2009.
[62] Wyatt Lloyd, Michael J. Freedman, Michael Kaminsky, David G. Andersen, Don't settle for eventual: scalable causal consistency for wide-area storage with cops, in: Proceedings of the Twenty-Third ACM Symposium on Operating Systems Principles, PSOSP'11, ACM, New York, NY, USA, 2011, pp. 401–416.
[63] Grzegorz Malewicz, Matthew H. Austern, Aart J.C Bik, James C. Dehnert, Ilan Horn, Naty Leiser, Grzegorz Czajkowski, Pregel: a system for large-scale graph processing, in: SIGMOD, 2010.
[64] http://www.information-management.com/issues/21_5/big-data-is-scaling-bi-and-analytics-10021093-1.html.
[65] Dejan S. Milojicic, Vana Kalogeraki, Rajan Lukose, Kiran Nagaraja, Jim Pruyne, Bruno Richard, Sami Rollins, Zhichen Xu, Peer-to-peer Computing. Technical Report HPL-2002-57R1, Hewlett Packard Laboratories, May 2, 1900.
[66] Tomer Y. Morad, Uri C. Weiser, A. Kolodnyt, Mateo Valero, Eduard Ayguadé, Performance, power efficiency and scalability of asymmetric cluster chip multiprocessors, Comput. Archit. Lett. 5 (1) (2006) 14–17.
[67] Christopher Olston, Benjamin Reed, Utkarsh Srivastava, Ravi Kumar, Andrew Tomkins, Pig Latin: a Not-So-Foreign language for data processing, in: SIGMOD, ACM, 2008. ID: 1376726.
[68] Innovation at Google: The Physics of Data, 2009, http://www.parc.com/event/936/innovation-at-google.html.
[69] Phoebus, https://github.com/xslogic/phoebus.
[70] Keshav Pingali, Donald Nguyen, Milind Kulkarni, Martin Burtscher, Amber Hassaan, Rashid Kaleem, Tsung-Hsien Lee, Andrew Lenharth, Roman Manevich, Mario Méndez-Lojo, Dimitrios Prountzos, Xin Sui, The tao of parallelism in algorithms, in: PLDI, 2011.
[71] Gerald J. Popek, R.P. Goldberg, Formal requirements for virtualizable third generation architectures, Commun. ACM 17 (7) (1974).
[72] Russell Power, Jinyang Li, Piccolo: building fast, distributed programs with partitioned tables, in: OSDI, 2010.
[73] Moinuddin K. Qureshi, Vijayalakshmi Srinivasan, Jude A. Rivers, Scalable high performance main memory system using phase-change memory technology, in: Proc. 36th International Symposium on Computer Architecture, 36th ISCA'09, ACM SIGARCH, Austin, TX, June 2009, IBM T.J. Watson RC.
[74] Parthasarathy Ranganathan, From microprocessors to nanostores: rethinking data-centric systems, IEEE Comput. 44 (1) (2011) 39–48.
[75] Parthasarathy Ranganathan, Jichuan Chang, (Re)designing data-centric data centers, IEEE Micro 32 (1) (2012) 66–70.
[76] Colby Ranger, Ramanan Raghuraman, Arun Penmetsa, Gary Bradski, Christos Kozyrakis, Evaluating mapreduce for multi-core and multiprocessor system, in: Proceedings of the 13th Intl. Symposium on High-Performance Computer Architecture (HPCA), Phoenix, AZ, 2007.
[77] Naresh Rapolu, Karthik Kambatla, Suresh Jagannathan, Ananth Grama, Transmr: data-centric programming beyond data parallelism, in: Proceedings of the 3rd USENIX Conference on Hot Topics in Cloud Computing, HotCloud'11, USENIX Association, Berkeley, CA, USA, 2011, pp. 19–19.
[78] Benjamin Reed, Flavio P. Junqueira, A simple totally ordered broadcast protocol, in: Proceedings of the 2nd Workshop on Large-Scale Distributed Systems and Middleware, LADIS'08, ACM, New York, NY, USA, 2008, pp. 2:1–2:6.
[79] Erik Riedel, Garth Gibson, Christos Faloutsos, Active storage for large-scale data mining and multimedia, in: Proceedings of the 24th International Conference on Very Large Data Bases, VLDB'98, Morgan Kaufmann, East Sussex, San Francisco, 1998, pp. 62–73.
[80] Mendel Rosenblum, Tal Garfinkel, Virtual machine monitors: current technology and future trends, IEEE Comput. 38 (5) (2005) 39–47.
[81] Apache s4, http://incubator.apache.org/s4.
[82] Mehul Shah, Parthasarathy Ranganathan, Jichuan Chang, Niraj Tolia, David Roberts, Trevor Mudge, Data dwarfs: Motivating a Coverage Set for Future Large Data Center Workloads, Technical Report HPL-2010-115, Hewlett Packard Laboratories, November 8, 2010.
[83] J.E. Short, R.E. Bohn, C. Baru, How much information? 2010–report on enterprise server information. UCSD Global Information Industry Center, 2011.
[84] Clinton Wills Smullen IV, Vidyabhushan Mohan, Anurag Nigam, Sudhanva Gurumurthi, Mircea R. Stan, Relaxing non-volatility for fast and energy-efficient STT-RAM caches, in: HPCA, IEEE Computer Society, 2011, pp. 50–61.
[85] Twitter storm, https://github.com/nathanmarz/storm.
[86] Jie Tang, Shaoshan Liu, Zhimin Gu, Xiao-Feng Li, Jean-Luc Gaudiot, Achieving middleware execution efficiency: hardware-assisted garbage collection operations, J. Supercomput. 59 (3) (2012) 1101–1119.
[87] http://www.statisticbrain.com/twitter-statistics/.
[88] Scalable, Energy-Efficient Data Centers and Clouds, 2012, http://iee.ucsb.edu/Data_Center_Report.
[89] Leslie G. Valiant, A bridging model for parallel computation, Commun. ACM 33 (8) (1990) 103–111.
[90] Arun Venkataramani, Ravi Kokku, Mike Dahlin, Tcp nice: a mechanism for background transfers, in: Proceedings of the 5th symposium on Operating Systems Design and Implementation, OSDI'02, ACM, New York, NY, USA, 2002, pp. 329–343.
[91] Shivaram Venkataraman, Niraj Tolia, Parthasarathy Ranganathan, Roy H. Campbell, Consistent and durable data structures for non-volatile byte-addressable memory, in: Gregory R. Ganger, John Wilkes (Eds.), FAST, USENIX, 2011, pp. 61–75.
[92] Ymir Vigfusson, Hussam Abu-Libdeh, Mahesh Balakrishnan, Ken Birman, Robert Burgess, Gregory Chockler, Haoyuan Li, Yoav Tock, Dr. multicast: Rx for data center communication scalability, in: Proceedings of the 5th European conference on Computer systems, EuroSys'10, ACM, New York, NY, USA, 2010, pp. 349–362.
[93] Haris Volos, Andres Jaan Tack, Michael M. Swift, Mnemosyne: lightweight persistent memory, in: Proceedings of the 16th International Conference on Architectural Support for Programming Languages and Operating Systems, 16th ASPLOS'11, ACM Press, Newport Beach, CA, USA, 2011, pp. 91–104.
[94] March 2012. http://www.wallstreetdaily.com/2012/03/21/forever-growth-trends-and-five-stocks-set-to-profit/.
[95] A. Weiss, Computing in the clouds, Computing 16 (2007).
[96] Xiaoxia Wu, Jian Li, Lixin Zhang, Evan Speight, Ram Rajamony, Yuan Xie, Hybrid cache architecture with disparate memory technologies, in: Proc. 36th International Symposium on Computer Architecture, 36th ISCA'09, ACM SIGARCH, Austin, TX, 2009.
[97] Apache yarn, http://hadoop.apache.org/common/docs/r0.23.0/hadoop-yarn/hadoop-yarn-site/YARN.html.
[98] http://www.youtube.com/yt/press/statistics.html.

Karthik Kambatla is a Software Engineer in the Scheduling team at Cloudera Inc and is currently pursuing his PhD in the Department of Computer Science at Purdue University. Karthik's interests lie in distributed execution engines and his research focuses on improving their performance and applicability. He is a committer on the Apache Hadoop project.

Giorgos Kollias received the BSc in Physics in 2000 and the MSc in Computational Science in 2002 from the University of Athens, Greece, and the PhD in Computer Science from the University of Patras, Greece, in 2009. He then moved to Purdue University, USA and worked as a Postdoctoral Researcher in the Computer Science Department and the Center for Science of Information till 2013. Currently he works in IBM T.J. Watson Research Center, USA. His research interests include dynamical systems, Problem Solving Environments, graph analysis, randomized linear algebra and parallel computing.

Vipin Kumar is currently William Norris Professor and Head of the Computer Science and Engineering Department at the University of Minnesota. Kumar's current research interests include data mining, high-performance computing, and their applications in Climate/Ecosystems and Biomedical domains. He has authored over 300 research articles and has coedited or coauthored 11 books including widely used text books "Introduction to Parallel Computing" and "Introduction to Data Mining". His research has resulted in the development of the concept of isoefficiency metric for evaluating the scalability of parallel algorithms, as well as highly efficient parallel algorithms and software for sparse matrix factorization (PSPASES) and graph partitioning (METIS, ParMetis, hMetis). Kumar is a Fellow of the ACM, IEEE and AAAS. He received the Distinguished Alumnus Award from the Indian Institute of Technology (IIT) Roorkee (2013), the Distinguished Alumnus Award from the Computer Science Department, University of Maryland College Park (2009), and IEEE Computer Society's Technical Achievement Award (2005). Kumar's foundational research in data mining and its applications to scientific data was honored by the ACM SIGKDD 2012 Innovation Award, which is the highest award for technical excellence in the field of Knowledge Discovery and Data Mining (KDD).

Ananth Grama is the Director of the Computational Science and Engineering program and Professor of Computer Science at Purdue University. He also serves as the Associate Director of the Center for Science of Information. Ananth received his B. Engg from Indian Institute of Technology, Roorkee (1989), his M.S. from Wayne State University (1990), and Ph.D. from the University of Minnesota (1996). His research interests lie in parallel and distributed systems, numerical methods, large-scale data analysis, and their applications. Ananth is a recipient of the National Science Foundation CAREER award (1998), University Faculty Scholar Award (2002–07), and is a Fellow of the American Association for the Advancement of Sciences (2013).