Resource Scheduling in Data-Centric Systems

Zujie Ren, Xiaohong Zhang and Weisong Shi

1 Introduction

Effective resource scheduling is a fundamental issue for achieving high performance in various computer systems. The goal of resource scheduling is to arrange the best location of each resource and determine the most appropriate sequence of job execution, while satisfying certain constraints or optimizations. Although the topic of resource scheduling has been widely investigated for several decades, it is still a research hotspot as new paradigms continue to emerge, such as grid computing [1, 2], cloud computing [3, 4], big data analytics [5, 6], and so on.

With the explosive growth of data volumes, more and more organizations are building large-scale data-centric systems (DCS). These systems are hosted by one or more data centers, where they serve as IT infrastructures for data processing, scientific computing, and a variety of other applications involving "big data". Data-centric systems offer new solutions for existing applications and promote warehouse-scale data businesses such as cloud computing, cloud storage services, and so on.

Unfortunately, there is no widely accepted standard definition for data-centric systems. However, in general, if a computing system involves large volumes of data which are hosted by data centers, it can be labeled as "data-centric systems". Examples include large-scale web search engine, data management systems, data mining systems. Particularly, we focus on three kinds of data-centric systems in this chapter:

Z. Ren (✉)
School of Computer Science and Technology,
Hangzhou Dianzi University, Hangzhou, China
e-mail: renzju@gmail.com

X. Zhang
Shenzhen Institutes of Advanced Technology,
Chinese Academy of Science, Shenzhen, China
e-mail: xh.zhang@siat.ac.cn

W. Shi
Department of Computer Science, Wayne State University, Detroit, USA
e-mail: weisong@wayne.edu

S. U. Khan, A. Y. Zomaya (eds.), *Handbook on Data Centers,*
DOI 10.1007/978-1-4939-2092-1_46

- *Cloud computing platforms*. A cloud computing platform is depicted as a large pool of computing and storage resources, which provides various services (IaaS, PaaS and SaaS) and elastic resources [3] to public users via the Internet. Recent years have witnessed a rapid growth in the number of cloud computing platforms, such as Amazon EC2 [7], IBM Blue Cloud [8], Google AppEngine [9], RackSpace [10] and Microsoft Azure [11].
- *Data-Intensive Super Computing (DISC) systems*. DISC systems are new forms of high-performance computing (HPC) systems that concentrate on high-volume data, rather than computation. DISC is responsible for the acquisition, updating, sharing, and archiving of the data. In addition, DISC supports data-intensive computation over high-volume data [12–14].
- *MapReduce-style systems*. MapReduce-style processing systems are designed to deal with big data volume in parallel on large-scale clusters. A traditional and popular example is Hadoop [15], an open-source implementation of the MapReduce framework [16]. Hadoop can easily scale out to thousands of nodes and work with petabyte data.

In the context of DCS, effective resource scheduling is notoriously difficult due to the complexity and diversity of DCS. More specifically, the challenges for scheduling optimization include the following: (1) the software/hardware stack in data-centric systems is composed of many layers [17, 18]. The entities and objectives of scheduling may be completely different across these software/hardware layers. (2) the workload running the data-centric systems is significantly miscellaneous. The workload is usually comprised of long-running applications, Web services, MapReduce jobs, HPC jobs, and so on. Therefore, compared with the traditional distributed systems like distributed file systems and DBMS, data-centric systems pose many more challenges for improving resource efficiency by scheduling due to the system complexity and workload diversity.

To address these challenges, various resource scheduling methods in the context of DCS have been proposed in recent years. For example, motivated by the market behaviors in the field of economics, some literature has focused on regulating the supply and demand of resources in cloud environments, using such as commodity-based [19, 20] or auction-based strategies [21, 22]. These resource scheduling polices are designed for reducing cost for resource consumers and maximizing profits for resource providers. Other literature focuses on optimizing the system throughput by allocating resources based on various heuristics. For example, the scheduler may concentrate on system utilization [23], job completion time [24, 25], load balance [26], energy consumption [27–29], data locality [30, 31], or real-time satisfaction [32, 33].

While the topic of resource scheduling in data-centric systems is broad enough to provide enough content for a book, those existing techniques are scattered and poorly organized. A systematic survey on the existing research advances is necessary and helpful to further improvement and performance optimization. In this chapter, we classify the resource scheduling approaches into three categories according to the scheduling model: resource provision, job scheduling and data scheduling. We give a systematic review of the most significant techniques for these categories, and present

some open problems to be addressed. We believe this systematic and comprehensive analysis can help researchers and engineers to better understand those scheduling techniques and inspire new developments within this field.

The chapter is organized as follows. Section 2 presents the definitions of a list of terminologies used in the chapter. A taxonomy of existing works on resource scheduling is presented in Sect. 3. In Sect. 4, we will look at four case studies, each of which is derived from practical or productional systems. In Sect. 5, we outline interesting future trends and challenges of resource scheduling.

2 Terminology

Due to the diversity of data-centric systems, the terminology used in this field is often inconsistent. To clarify the description in this chapter, we define the following necessary terminology.

Resource. Resource is a collection of components that can be scheduled to perform an operation. Some traditional examples of resources are CPU cores for computing, memory spaces for storage, network links for transferring, electrical power, and so on.

Task. A task is an atomic action from the scheduler's point of view. A Task is defined by a collection of input data and corresponding operations.

Job. A job is a group of tasks that will be executed on a set of resources. The definition of jobs is recursive, which means that jobs are composed of sub-jobs and/or tasks, and sub-jobs can be decomposed further into atomic tasks.

Service. A service is a program to enable access to one or more resources, where the access is provided by a predefined interface. For instance, cloud computing, which is provisioned as services, are broadly divided into three categories: software-as-a-service (SaaS), platform-as-a-service (PaaS), and infrastructure as-a-service (IaaS).

Data-Centric Systems. Although there is no de-facto standard definition for the term "data-centric systems", they are very common in various forms. In most cases, they are characterized (partially or fully) by the following features:

- managing of large volumes of data, in range of petabyte-level and beyond
- hosted by one or more data centers
- involving complex software/hardware stacks
- serving for multiple users and execute diverse workloads

Generally, many computing systems can be labeled as "data-centric systems", such as web search engines, data management systems, and data mining systems. To summarize, this chapter concentrates on three kinds of traditional data-center systems, including cloud computing platforms, data-intensive super computing systems, and MapReduce-style systems.

3 Classification and State-of-the-Art

In this section, we present a broad view of resource scheduling issues in data-centric systems. As data-centric systems involve multiple software layers, scheduling operations take place on multiple layers. For example, assume that a set of MapReduce jobs are submitted to a data processing application, which is hosted on a cloud platform like Amazon EC2, the scheduling operations will be conducted multiple times. Firstly, when the application for processing MapReduce jobs, such as Hadoop, is loaded on the cloud, the application needs to be provisioned with a certain amount of resources, which is often referred to as *resource provision* (aka. *resource allocation*). Secondly, when the set of job requests are submitted to the application, the scheduler in the application needs to map the set of jobs to multiple servers in a certain manner, which is also known as *job scheduling*. Thirdly, to improve the resource utilization or job execution efficiency, the scheduler within storage systems needs to schedule the data transfer, replication, distribution, either during the job execution or in advance, which is often referred to as *data scheduling*.

3.1 Hierarchy of Resource Scheduling in DCS

In fact, similar as in the context of a data processing system, the resource scheduling issue in DCS also can be generally divided into the problems of resource provision, job scheduling and data scheduling. Although data-centric systems come with variable implementations, we still can abstract a common hierarchical architecture of various data-centric systems from the perspective of scheduling, which is depicted in Fig. 1.

- **Resource provision.** Resource provision is to allocate resources to satisfy multiple applications efficiently. In one aspect, on the top layer of data-centric systems, various applications, VM instances, Web services etc., run on the data-centric systems. They demand a certain type and amount of resources when they are loaded. In the other aspect, the data-centric systems is a unified resource platform, which holds massive computation and storage resources in the data centers. The resources are allocated to users based on a certain policy to satisfy the requirements of resource providers and users. Therefore, the scheduling issue in this layer is often also referred to as *resource allocation* [34–37].
- **Job scheduling.** Within a data-centric system, various jobs, such as HPC and MapReduce-style jobs, will be submitted in parallel by many applications (or users). Simple scheduling algorithms such as FIFO, are hard to satisfy performance requirements in most cases. To improve the job's execution performance, a scheduling algorithm is needed to assign jobs or tasks to appropriate nodes in a certain order. Therefore, the scheduling issue in this layer is also known as *job scheduling*.

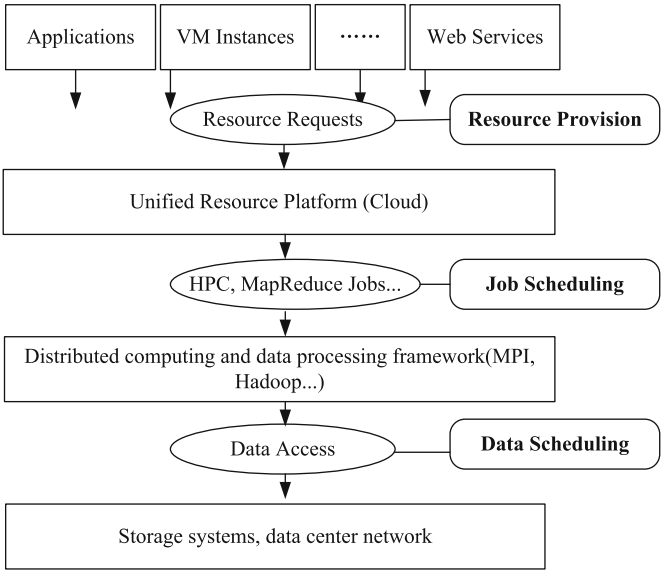

Fig. 1 Resource scheduling hierarchy in DCS

- **Data scheduling.** On the storage layer of data-centric systems, there are large volumes of data stored and managed by distributed nodes. The goal of data scheduling is two-fold: in one aspect, data-centric systems employ various data placement and migration techniques to increase the storage resource utilization, data reliability and availability; in the other aspect, data-centric systems apply online scheduling techniques for data prefetching, data transfer etc., for accelerating the job (request) execution, aiming to reduce to data access/transfer latency Fig. 2.

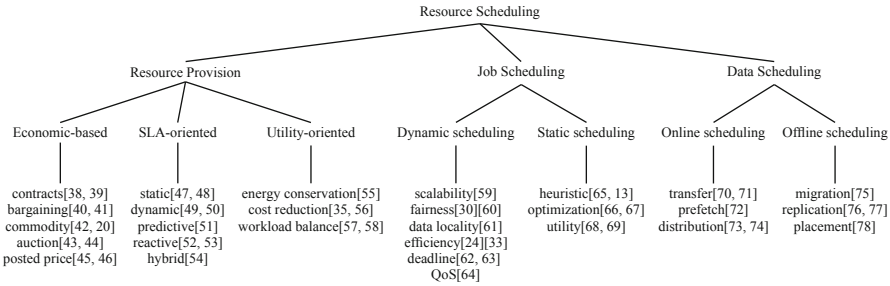

Fig. 2 Resource scheduling taxonomy in DCS

3.2 Resource Provision

During the past few years, cloud computing has become a main trend in delivering IT services, where the computing and storage capabilities are shared among multiplex many users. In a cloud computing platform, the resources are available on-demand, charged on a pay-as-you-go basis. In one aspect, cloud providers hold enormous computing resources in their data centers, while in the other aspect, cloud users lease the resources from cloud providers to run their applications. Usually, the resource requirement imposed by cloud users are heterogeneous [18] and time-varying [79, 80], which makes the scheduling much more complicated.

According to the provision model, we classify the resource provision techniques into three groups: economic-based, SLA-oriented, and utility-oriented. The first and second groups focus on resource provision issues between providers and consumers using economical models or SLA contracts, while the third group concentrates on high-efficiency resource management from the perspective of the data center owner.

3.2.1 Economic-Based Resource Provision

To maximize benefits on cloud platforms, many researchers proposed various economic models to effectively solve the issues of scheduling problems in the grid or cloud environments, such as commodity market [81], posted price [45, 82], tendering/contract [38], bargaining [83, 84], auction [43, 21], and so on. Economics-based methods are very suitable for handling the provision issues in a cloud environment, as they have been effectively utilized in the field of economics to regulate the supply and demand of limited resources.

The concept of a *commodity market* model is similar to commodity trade in real markets in our daily life. Resource providers specify their service prices and charge users according to the amount of resources they use. The users can freely choose a proper service, but the price is unable to change. The prices can be generated based on the resource supply and demand. Generally, the resources are priced in such a way that supply and demand equilibrium is reached.

The *posted price* model is similar to the commodity market model. The only difference is that the posted price model advertises special offers in order to attract consumers. The posted-price offers will have usage conditions, but they might be attractive for some users because the posted prices are generally cheaper compared to regular prices.

Although some economic-based resource allocation are non-price-based [85], most of the economic-based schedulers emphasize the schemes for establishing an appropriate price based upon their users' demands. They in turn determine a proper price that keeps supply and demand in equilibrium. Several market principles are considered in the process of figuring out the price scheme, including equilibrium price [82], Pareto efficiency [86], individual rationality [87], stability [88], and communication efficiency [55].

3.2.2 SLA-Oriented Resource Provision

Although *economic-based methods* achieve impressive performances for allocating resources, there exist some limitations in some cases. The limitations lie in the difficulty for users to determine an quantized resource demand. When a user sends a request for resources to a provider, the provider looks for resources to satisfy the request and assigns the resources to the requesting users, usually as a form of virtual machines with different capabilities. However, for the users of the systems it would be difficult, even unable, to make a decision about the number and types of resources needed, especially when the request is time-varying. The ultimate concern of the user is to meet application-level requirements, instead of determining resource allocation needs.

To address such problems, many researchers proposed dynamic provisioning of resource using virtualization. The amount of provisioned resource can be adjusted with the workload fluctuates over time. Meng et al. [47] proposed a joint-VM provisioning approach in which multiple VMs are consolidated and provisioned together, based on an estimate of their aggregate capacity needs. This approach exploits statistic multiplexing among the workload patterns of multiple VMs to improve the overall resource utilization.

Garg et al. [49] proposed a dynamic resource provision strategy that considers SLAs of different types, particularly transactional and non-interactive applications. Both types of applications have different types of SLA requirements and specifications. For transactional workload, the placement decisions are made dynamically to respond to the workload variation. For non-interactive workload, the resource manager predicts the future resource availability and schedules the jobs by stealing CPU cycles.

Cloud providers such as Amazon EC2, usually offer differentiable QoS guarantees for users, which are essential for ensuring the service quality users received. The QoS guarantees are defined in the form of SLA (Service Level Agreement). Under such circumstances, cloud providers are delegated to make the decisions about the number and types of resources allocated. *SLA-oriented methods* are proposed to allocate resources to each user with the fulfillment of SLA [89]. Besides satisfying the SLA, these methods also concern other system performance metrics, such as improving the resource utilization [35], energy conservation [90], and cost reduction [56].

3.2.3 Utility-Oriented Resource Provision

Besides these two kinds of resource provision, there are some provision techniques that neglect actual levels of services required by different users and assume all requests are of equal importance. These provision techniques focus on the system utilization, rather than the profit and SLA contracts, so they are labeled as utility-oriented resource provision.

Paragon [35] is a heterogeneity and interference-aware data center scheduler, which supports the classification of an unknown application with respect to

heterogeneity and interference. Paragons classification engine utilizes existing data from previously scheduled applications and offline training and requires only a minimal signal about a new workload. It uses singular value decomposition to perform collaborative filtering and identify similarities between incoming and previously scheduled workloads.

Researchers [90] improve the service scheduling by historical workload traces characterization. The long-term workload patterns are derived by workload discretization. The resources are allocated predictively by the predicted base load at hour-level scale and reactively allocated to handle any excess workload at minute-level scale. The combination of predictive and reactive provisioning contributes to meeting SLA requirements, conserve energy, and reduce allocation cost.

Beloglazov et al. [55] proposed resource provisioning and allocation algorithms for energy-efficient management in cloud computing environments. The proposed energy-aware allocation heuristics provision data center resources to client applications in a way that improves energy efficiency of the data center, while delivering the negotiated Quality of Service (QoS).

Birke et al. [91] characterized the evolution and the elasticity of workload demands in several thousands of servers at geographically distributed data centers, to improve the effectiveness of capacity planning and resource provision in data centers.

Xiong et al. [56] proposed a SLA (Service Level Agreement)-based approach for allocating resources to satisfy the quality of service (QoS) constraints while minimizing the total cost of computational power. These QoS metrics include percentile response time, cluster utilization, packet loss rate and cluster availability.

Economic-based methods are very suitable for scheduling resources in cloud environments, for regulating the supply and demand of resources at market equilibrium. With the advent of economic-based methods, SLA-oriented methods are promoted to differentiate QoS guarantees for users. SLA-oriented methods are suitable for the users that are only concerned with application-level requirements, rather than the amounts and types of involved resources. Utility-oriented methods aim to improve the system utilization, regarding all resource requests as having equal importance. Therefore, utility-based methods are applicable in cluster computing systems that do not have to consider customer-driven service managements.

3.3 Job Scheduling

Once the resources are provisioned to applications (or VM instances), each application needs to schedule the allocated resources to perform various computation jobs. In this context, the scheduling problem concerns matching the jobs to the available resources for maximization of system throughput, execution efficiency, and so on. The optimal matching is an optimization problem with NP-complete complexity.

Due to the high diversity of jobs and situations, there is no general job scheduling algorithm that can fit for all jobs. The most widely-used methods are heuristic methods, such as genetic algorithms, tabu search and simulated annealing. These methods

have been successfully applied as approximately optimal algorithms to solve the job scheduling problem.

In this chapter, we classify these job scheduling methods into *static scheduling* and *dynamic scheduling*. Static scheduling techniques are suitable for the environments where the details of all jobs and resources are known prior to the scheduling being performed. On the contrary, dynamic job scheduling is performed on the fly each time a job arrives. Dynamic scheduling techniques are applied in the environments where job information and resource states cannot be available in advance.

3.3.1 Static Job Scheduling

Static scheduling techniques are commonly used in HPC and computing grid environments. In order to minimize the turnaround time, many approximation algorithms have been proposed, such as genetic algorithms [92], simulated annealing algorithms [93], and ant colony algorithms [94]. Some of these approximation methods are inspired by nature's phenomena. They do not guarantee an absolute optimal solution, but they are guaranteed to find an approximate optimal solutions in a timely manner. The quality of these solutions can be tuned by a series of parameters.

Genetic Algorithm is an evolutionary technique for solving job scheduling problem where the solution space is large. Using a genetic algorithm, the scheduling problem is represented as a genome, while a scheduling genome can be defined by the sequence of tasks. Each task and its corresponding start time represents a gene, which is a unit of genome.

The Simulated Annealing (SA) is a well-known greedy method where the search process is simulated by the thermal procedure of obtaining low-energy crystalline states of a solid. To avoid falling local optimum, SA results in a worse solution in some cases, however in most cases it results in a better solution. Analogous to the thermal procedure of metal smelting, the probability is based on the temperature that decreases for each iteration. This means, as the search progresses, a worse solution is increasingly difficult to be generated.

So far, static scheduling has been widely applied in the field of grid computing. Braun et al. [95] evaluated and compared the efficiency of 11 heuristics, including GA, SA, Tabu, Minimum Execution Time (MET), Minimum Completion Time (MCT), and so on. This study gives valuable guidelines for choosing a technique which outperforms another under a specific circumstance. More details on static scheduling can also be found in [96].

3.3.2 Dynamic Job Scheduling

Dynamic scheduling are applicable to the situation when the jobs arrive one after another, rather than being fixed. During the jobs execution, available resources can be scheduled on the fly to handle the new coming jobs. The goals of various dynamic job scheduling methods differ greatly. Besides system throughput, many job scheduling

methods are designed to emphasize other metrics in certain environments, including fairness, load balance, QoS guarantee, energy consumption, and so on.

Schedulers in Hadoop are a representation of the implementation of dynamic job scheduling. The original default scheduler in Hadoop uses FIFO policy to schedule jobs. Later significant research efforts have been devoted to developing more effective and efficient schedulers. Now, the default scheduler in Hadoop is replaced by FAIR scheduler [60]. Moreover, a variety of alternative job schedulers, i.e. Delay Scheduler, Dynamic Proportional Scheduler, Capacity Scheduler etc., have been proposed.

Zaharia et al. [60] proposed FAIR Scheduler, with a rational of allocating every job a fair share of the slots over time. In fair scheduler, jobs are assigned to pools, which are assigned a guaranteed minimum quota of logic units of resources, aka. *slots*. Slots are first allocated into pools and then allocated to individual jobs within each pool. Each pool is given a minimum share and the sum of minimum quota of all pools does not exceed the system capacity. Idle slots are shared among jobs and assigned to the job with the highest slot deficit. Due to its simplicity and high performance, FAIR scheduler has gained a high popularity in Hadoop community. However, some recent work [97] has shown that the FAIR scheduler is not very well-suited for scheduling diverse workloads with considerably small jobs.

Similar as FAIR scheduler, Capacity Scheduler was also developed to ensure a fair allocation of computing resources among large number of users. The jobs from these users are submitted to different queues. Each queue is configured with a fraction of resource capacity, and free resources can be shared among the queues. Within each queue, the share of resources allocated to a user is limited, this is to guarantee that no user occupies or controls the resources exclusively. In addition, jobs can be configured with priorities. Jobs with high priorities can be allocated resources preferentially.

Delay scheduling method proposed by Zaharia et al. [31] preferentially schedule jobs to nodes where these jobs have good data locality. The method would schedule the job of which the input data is available on a node with free slots, rather than the job with the highest priority. Delay scheduling performs well in typical Hadoop workloads because there are multiple locations where a task can run to access each data block.

YARN [59], known as the next generation of Hadoop compute platform, separates resource management functions from the programming model. This separation makes various alternative programming models besides MapReduce applicable on YARN, such as Dryad [98], Spark [99], and so on.

In YARN, the functionalities of the *JobTracker* node in traditional Hadoop is split and performed by two components: a global *ResourceManager* and per-application *ApplicationMasters*. The *ResourceManager* allocates resources among all the applications in the system. The *ResourceManager* cooperates with per-node slaves, and form the data-computation framework. The *ApplicationMaster* is responsible for negotiating resources from the *ResourceManager* and working with the computing slaves to execute and monitor the tasks.

Chang et al. [24] proposed a theoretical framework for optimal scheduling in MapReduce. The authors formulate a linear program which minimizes the job completion times to solve the problem. Given the hardness at solving the linear program, approximate algorithms are designed to achieve feasible schedules within a small constant factor of the optimal value of the objective function.

Sandholm et al. [100] developed a dynamic priority (DP) scheduler, which allows users to bid for task slots or quality of service levels dynamically. For a given user, the budget of slots is proportional to the spending rate at which a user has previously bid for a slot and inversely proportional to the aggregate spending rate of all existing users. When a group of slots have been allocated to one user, that same spending rate is deducted from the users budget. Using this mechanism, the scheduler allows users to optimize and customize their slots allocation according to job requirements and system overhead.

Sparrow [33] provides low response times for parallel sub-second jobs that are executed on a large-scale cluster. The authors focus on short task workload scheduling for low-latency and high throughput. The schedulers run on a set of machines that operate autonomously and without shared state. Such a decentralized design offers attractive properties of high scalability and availability.

Energy-aware methods aim to optimize energy consumption by job dispatching. The method proposed by Wang et al. [29] and the one proposed by Kliazovich et al. [26] belong to this group of methods. Wang et al. [101] presents a thermal aware scheduling algorithm for data centers to reduce the temperatures inside of the data center. An analytical model, which describes data center resources with heat transfer properties and workloads with thermal features are used to guide the scheduler to find suitable resources for workload execution.

Nguyen et al. proposed a reputation-based resource selection scheme to reduce the energy waste caused by failures. They introduced a reputation model, called Opera, combined with a vector representation of the reputation and the just-in-time feature that represents the real-time system status. Opera enables the scheduler in Hadoop to select appropriate nodes which helped to reduce not only the number of re-executed tasks, but also improve the energy efficiency of the whole system.

Job scheduling focused on matching multiple jobs to multiple nodes using various heuristics. Scheduling techniques for MapReduce jobs usually use dynamic heuristics, such as fairness, data locality, and execution efficiency. While for HPC jobs, the scheduling techniques use static heuristics, such as OLB (Opportunistic Load Balancing), MET, MCT, GA, SA, and so on.

3.4 Data Scheduling

In the early stage of distributed computing systems, such as data grids, the data scheduling was coupled with job scheduling. In this mechanism, the cost for data access and movement are taken into considerations when deciding job scheduling. However, due to the increased growth of data size, data scheduling was gradually

decoupled from job scheduling [102], and became an important issue in large-scale distributed systems.

There have been several recent studies investigating new approaches for data management and data transfer in distributed systems. These approaches can be classified into two categories: *online* data scheduling and *offline* data scheduling. The former focuses on scheduling data for serving the job (request) execution. The main goal is to reduce to data access latency and improve the job (request) execution efficiency. The latter handles the data scheduling for improving the storage resource utilization or improving the data reliability. These data scheduling approaches are offline because they are not directly performed for the online job execution.

3.4.1 Online Data Scheduling

Balman et al. [70] developed data scheduling methodologies and the key attributes for reliability, adaptability and performance optimization of distributed data placement tasks. An adaptive scheduling of data placement tasks is proposed for improving end-to-end performance. The adaptive scheduling approach includes dynamically tuning data transfer parameters over wide area networks for efficient utilization of available network capacity and optimized end-to-end data transfer performance.

To optimize the performance of data transfer, Chowdhury et al. [71] proposed a global data transfer management architecture and a set of network resource scheduling algorithms. Guo et al. [103] decrease the network traffic via inter-flow data aggregation with an efficient incast tree.

Al-Fares et al. [104] proposed a dynamic flow scheduling system, called Hedera, for multi-stage switch topologies found in data centers. Hedera collects flow information from constituent switches, computes non-conflicting paths for flows, and instructs switches to re-route traffic accordingly. The design goal of Hedera is to maximize aggregate network utilization-bisection bandwidth and to do so with minimal scheduler overhead or impact on active flows.

Seo et al. [72] proposed prefetching and pre-shuffling optimization to improve the MapReduce performance. The prefetching scheme involves the intra-block prefetching and the inter-block prefetching. The prefetching scheme exploits data locality, while the pre-shuffling scheme significantly reduces the network overhead required to shuffle key-value pairs.

3.4.2 Offline Data Scheduling

To improve data locality, Abad et al. [76] observed the correlation between benefits of data locality and data access patterns. They propose a distributed adaptive data replication algorithm, called DARE, that aids the scheduler to achieve better data locality. DARE addresses two problems, how many replicas for each file and where to place them. DARE makes use of probabilistic sampling and a competitive aging algorithm independently at each node. It takes advantage of existing remote data accesses in the system and incurs no extra network usage.

To save the energy consumption caused by communication fabric, DENS [26] combines energy efficiency and network awareness to achieve the balance between job performance, QoS requirement, traffic demands and energy consumed by the data center. DENS is designed to avoid hotspots with a data center while minimizing the number of computing servers required for job execution. DENS is particulary relevant in data centers running data-intensive jobs which produce heavy data transfer.

Ranganathan et al. [102] developed a data scheduling framework to satisfy various and general metrics and constraints, including resource utilization response times. The data movement operations may be either tightly bound to a job, or performed by a decoupled, asynchronous process on the basis of historical data access patterns.

In the context of traditional data storage systems, such as data grids, various offline data scheduling have been proposed and implemented. Offline data scheduling focuses on data storage, transfer, copy and replication management, aiming to improve the utilization ratio of storage resources and data access QoS, instead of directly serving the process of task execution. Online data scheduling focuses on job execution acceleration, and explores the strategies of data prefetch, parallel transfer and distribution for task execution procedure on a massive data processing framework. Compared with offline data scheduling, online data scheduling overcomes the limitation of lack-responsivity to job execution, and limits data I/O latency during the job execution.

4 Case Studies

Section 3 reviewed recourse scheduling techniques from three aspects, resource provision, job scheduling and data scheduling. In this section, we present how these techniques work in practical production systems. Particularly, we have chosen Amazon EC2, Dawning Nebulae, Taobao Yunti, and Microsoft SCOPE as the cases for study.

4.1 Amazon EC2

Amazon EC2 is one of the most popular IaaS cloud platforms which allow users to rent computing and storage resources to run applications, typically in forms of virtual machines. EC2 enables users to create virtual machines, each of which is called an instance. EC2 defines several type of instances,and configures each type with different computing power, memory and storage capacity.

EC2 applies *commodity market* and *posted pricing* models for provisioning the resource to users. More specifically, EC2 creates separate resource pools and has separate capacities for each type of VM. The market price for each VM type can fluctuate periodically to reflect the balance between demand and supply. Using the *commodity market* model, EC2 announces its service price according to the resource capacity and configuration. Customers can choose an appropriate service that meet

their objective. The pricing policy can be derived from the resource supply and demand. In general, services are priced in such a way that achieves a supply and demand equilibrium. Using the *posted price* model, EC2 announces the special offers as a supplement of regular prices. The scheduling compares whether special offers can meet the requirement of users, and match the supply and demand if they are matched. If not, the scheduling apply commodity strategy as usual.

In addition, EC2 offers three purchasing models to facilitate the cost optimization for users. The models provide different guarantees regarding when instances can be launched and terminated.

1. On-Demand instances, which allow users to pay an hourly fee with no guarantee that launching will be possible at any given time.
2. Reserved instances, which allow users to pay a low, one-time fee and in turn receive a significant discount on the hourly usage charge for that instance.Paying a yearly fee buys clients the ability to launch one reserved instance whenever they wish.
3. Spot instances, which enable users to bid for unused Amazon EC2 capacity. The Spot Price changes periodically based on supply and demand, and customers whose bids meet or exceed it gain access to the available Spot Instances.

4.2 Dawning Nebulae

Supercomputers are regarded as the important infrastructure to carry out high performance computing. They are expected to run not only computation-intensive applications but also data-intensive applications, which challenges the job scheduling softwares on these supercomputers. To satisfy the requirements of different users, the scheduling softwares must exploit various policies, and assign different kinds of jobs flexibly. Here, we use Dawning Nebulae as a case of the job scheduling techniques applied to supercomputers.

Dawning Nebulae is a supercomputer developed by Chinese Academy of Sciences. It includes more than 9200 multi-core CPUs, and more than 4600 NVIDIA GPUs. It achieves a performance of more than 1270 trillion operations per second or 1.27 petaflops [105]. It ranked second in the TOP 500 list of the world's most powerful supercomputers released in June 2010 [106]. Dawning Nebulae has been set up in NSCC-Shenzhen[http://www.nsccsz.gov.cn]. It provides about 200 user groups and research entities with application services such as weather forecast, ocean data simulation, gene research, universe evolution, and so on.

Dawning Nebulae includes huge computing resource and storage resource, and has to depend on a special and powerful software platform to manage these resource. Platform LSF (Load Sharing Facility) [107] is such a platform. It contains multiple distributed resource management softwares, and it can connect computers into a cluster, monitor loads of systems, schedule and balance workload and so on. Here, we only focus on the scheduling software of Platform LSF, and take it as the scheduler of Dawning Nebulae.

The scheduler provides several scheduling policies like first-come-first-service (FCFS), preemption, fair share, and so on. It supports multiple policies co-existing in the same cluster. For convenience of description, we introduce these policies one by one.The first policy is FCFS. According to this policy, the scheduler attempts to assign jobs in the order submitted. However, the shorter jobs with higher priorities will be pending for a long time if a long job with low priority was submitted earlier.

The second policy is the preemption policy. Preemption is not enabled until all the job slots in a cluster are occupied. After receiving the job with high priority, the scheduler suspends one job with low priority to free the slots occupied by the job. And then, it assigns the job with high priority to these slots. It resumes the suspended job if free job slots are available.

The third policy is the fair share policy. According to this policy, the scheduler divides cluster resources into shares, and assign shares to users. The policy can avoid the cluster resources monopolized by one user. The forth policy is exclusive policy. With this policy, the scheduler allows a job exclusive use of specified server hosts, and does not preempt the exclusive jobs. The last policy is the backfill policy. Under the policy, the scheduler allows small jobs to use the slots reserved for other jobs. However, it will kill those small jobs if they cannot be finished within their run limit.

4.3 Taobao Yunti

With the rapid growth of data volume in many enterprises, effective and efficient analytics on large-scale data becomes a challenging issue. Large-scale distributed computing systems, such Hadoop, have been applied by more and more organizations. Here, we take a Hadoop production cluster in Taobao [108] as another example to illustrate job scheduling techniques.

Taobao is the biggest online e-commerce enterprise in Asia, ranked 10th in the world as reported by Alexa. The Yunti cluster is an internal data platform in Taobao for processing petabyte-level business data mostly derived from the e-commerce web site of "www.taobao.com". The total volume of data stored in the Yunti has exceeded 25 PB, and the data volume grows with the speed of 30 TB per day.[1] The goal of the Yunti cluster is to provide multi-user businesses with large-scale data analysis service for some online applications. Yunti is built on Hadoop 0.19, with some slight modifications.

In the early stage, the Yunti cluster directly employed FAIR [60] to allocate the slots because FAIR achieves high performance and supports multi-user clusters. However, after several months of system running, it is observed that FAIR is not optimal for scheduling small jobs within a miscellaneous workload. The goal of FAIR is to assure the fairness among all jobs. FAIR always reassigns idle slots to the pool with the highest slot deficits. However, small jobs usually apply fewer slots, thus the slot deficits of small jobs are often smaller than the ones of normal jobs. Therefore, small jobs are more likely to suffer from long waits than the other jobs.

[1] These statistics were released on the year of 2012.

The users of Yunti submitting small jobs, including application developers, data analysts and project managers from different departments in Taobao, will complain about the long-waits.

As new workloads which feature short and interactive jobs are emerging, small jobs are becoming pervasive. Many small jobs are initiated by interactive and online analysis, which requires instant and interactive response. Ren et al. [97] proposed and implemented a job scheduler called Fair4S, to optimize the completion time of small jobs. Fair4S introduces pool weights and extends job priorities to guarantee the rapid response for small jobs. It is verified that Fair4S accelerates the average waiting times by a factor of 7 compared with FAIR scheduler for small jobs.

4.4 Microsoft SCOPE

SCOPE [109] is a distributed computation platform in Microsoft for processing large-scale data analysis jobs and serving a variety of online services. Tens of thousands of jobs are executed on SCOPE everyday. Scope integrates parallel databases with MapReduce systems, achieving both good performance and scalability.

SCOPE relies on a distributed data platform, named COSMOS, for storing large volumes of data sets. COSMOS is designed to run on tens of thousands of servers and has similar goals to other distributed storage systems, like Google File System [110] and Hadoop Distributed File System [111]. COSMOS is an append-only file system optimized for large sequential I/O. All writes are append-only, and concurrent writers are serialized by the system. Data are distributed and replicated for fault tolerance and compressed to save storage and increase I/O throughput.

In SCOPE, the executions of jobs are scheduled by a centralized job manager. The job manager constructs the job graph (directed acyclic graph) and schedules the tasks across the available servers in the cluster. The job manager simplifies job management by classifying distinct types of vertices into separate stages. Like JobTracker in Hadoop, the job manager maintains the job graph and monitors the status of each vertex (task) in the graph.

As SCOPE is deployed on globally distributed data centers, an automated mechanism to place application data across these datacenters is quite necessary. SCOPE employs a data placement algorithm, called Volley [112], to minimize the bandwidth cost and data access latency. Volley analyzes the logs using an iterative optimization algorithm based on data access patterns and client locations, and outputs migration recommendations back to the cloud service.

Volley periodically analyzes COSMOS to determine whether the migration should be executed. To perform the analysis, Volley relies on the SCOPE to accelerate the analysis efficiency. The analysis procedure is composed of three phases. In Phase 1, a reasonable initial placement of data items based on client IP addresses is computed. In Phase 2, the placement of data items by moving them freely over the surface of the earth is improved iteratively, which consumes the dominant computational time. Phase 3 iteratively collapses data with the satisfaction of capacity constraints of data centers.

5 Future Trends and Challenges

The topic of resource scheduling has been investigated in a great deal of literature, however, this is still an emerging field and there are many open problems in the area of data-centric systems. In this section, we enumerate a few such challenges that may help to inspire new developments in the field.

Increasing System Heterogeneity. With the progress of IT technologies, new software and hardware products emerge increasingly. In order to improve system performance and satisfy users' requirements, data centers have to adopt timely new products such as SSD and SDN [113], and hence they always include different types of equipment, even multiple generation equipments of the same type. Data centers are heterogeneous inevitably, and their heterogeneity grows with the adoption of new equipments. The ever-growing heterogeneity challenges resource provision especially when considering the different requirements from users.

It's very common that some tasks are designed to run on some machines for special purposes, i.e. the machines with special accelerators for an expected performance goal. Users define the constraints or preference of the machines to run their tasks by task specifications, which provide detailed requirements of users, meanwhile this makes resource provision more difficult and complicated. In addition, such resource affinity and constraints also complicate task migration.

Scalable Decentralized Scheduling. In a system with a centralized architecture, scheduling decision are made by a master node. The node maintains all information about tasks and keeps track of all available resources in the system. A centralized scheduler can be deployed easily, while its performance is limited by the master node. However, in a decentralized system, a master node and multiple slave nodes cooperate to schedule tasks. Hence, the scheduler in such a system can assign tasks with higher performance and scalability.

Decentralized schedulers have begun to attract more and more attentions as the scales of data centers grow. In decentralized schedulers, the nodes involved in co-scheduling are assumed to be autonomous, and responsible for their own scheduling decisions. However, if these nodes make these decisions independently, they can only optimize their performance rather than the performance of the whole system. New techniques and models need to be designed to schedule jobs, and hence optimize the performance of the whole system.

Enhancing Information Sharing. In data-centric systems of which the resources belong to multiple providers, users request resources to run their applications, while providers respond to these requests, and allocate resource for the users. If providers and users can share detailed information about resources and applications, schedulers can make efficient decision, and optimize system performance. However, providers and users only reveal limited information about resources and applications due to security concerns as well as other reasons. Some works were carried out to capture characters of workloads by analyzing historical trace, which makes it feasible to optimize job schedulers according to workloads. For periodic jobs, if we can derive their

characters, we can optimize the scheduling of these kind of jobs by pre-scheduling. Unfortunately, there exist few examples of such work.

Schedulability Analysis. When processing real-time jobs like interactive queries, periodic jobs and so on, a data-centric system must satisfy the time constraints of them. However, it is challenging to satisfy the time constraints because the system has to respond to the requirements from multiple users with relative QoS, especially when the job scales increase dramatically. And hence, an efficient and smart scheduler is needed to handle these kind of real-time jobs. Unfortunately, not all data-centric systems are suitable for real-time jobs. So it is very important to analyze whether a system can process real-time jobs with the specified time constraints before submitting real-time jobs to the system. There exist some research works which carry out scheduability analysis, however they only apply to multiprocessors [114] and virtualized platforms [115]. Besides, the models in these works are simple and only suitable for computing resources. Therefore, these works cannot be exploited to do scheduability analysis in data-centric systems, and new scheduability analysis techniques should be investigated as soon as possible with the consideration of computing resources, storage resources, network bandwidth, job scale, data distribution, resource competition, dynamic load, and so on.

Predictive Resource Allocation. Resource demand prediction [116] plays an essential role in dynamic resource allocation and job scheduling. For example, if a user has a job that needs to be finished within a certain deadline, an adequate amount of computing resources must be allocated. To determine whether or not a certain amount of resources are "adequate", the user needs to predict the completion time of the job with the resources. However, due to the heterogeneity and dynamism of the workload, the prediction of future resource demands would be hardly accurate. Reiss et al. [18] analyzed Google trace data [117] to reveal several insights which are helpful for improving the resource scheduling in a cloud infrastructure. The most notable characteristics of workload are heterogeneity and dynamism, which make the resource demand prediction very difficult.

6 Conclusions

In this chapter, we gave a survey of the scheduling techniques used in the three kinds of data-centric systems, including cloud computing platforms, data-intensive super computing systems, and MapReduce-style systems. According to the scheduling model, we categorized these techniques into three groups, including resource provision, job scheduling and data scheduling. We reviewed the new techniques systematically and outlined the open problems in each level. Further more, four practical systems selected from the industrial field are discussed to further understand the scheduling techniques and their applications. Finally, we concluded with some open problems in resource scheduling, aiming to inspire new developments within this field.

Acknowledgement We thank Raymond Darnell Lemon for his valuable comments on the early version of this chapter. This research is supported by NSF of Zhejiang (LQ12F02002), NSF of China (No. 61202094), Science and Technology Planning Project of Zhejiang Province (No.2010C13022). Xiaohong Zhang is supported by Ph.D. foundation of Henan Polytechnic University (No. B2012-099). Weisong Shi is in part supported by the Introduction of Innovative R&D team program of Guangdong Province (NO. 201001D0104726115), Hangzhou Dianzi University, and the NSF Career Award CCF-0643521.

References

1. Schwiegelshohn, U., Badia, R.M., Bubak, M., Danelutto, M., Dustdar, S., Gagliardi, F., Geiger, A., Hluchy, L., Kranzlmüller, D., Laure, E., et al.: Perspectives on grid computing. Future Generation Computer Systems **26**(8) (2010) 1104–1115
2. Xhafa, F., Abraham, A.: Computational models and heuristic methods for grid scheduling problems. Future generation computer systems **26**(4) (2010) 608–621
3. Armbrust, M., Fox, A., Griffith, R., Joseph, A.D., Katz, R., Konwinski, A., Lee, G., Patterson, D., Rabkin, A., Stoica, I., et al.: A view of cloud computing. Communications of the ACM **53**(4) (2010) 50–58
4. Foster, I., Zhao, Y., Raicu, I., Lu, S.: Cloud computing and grid computing 360-degree compared. In: Grid Computing Environments Workshop, 2008. GCE'08, Ieee (2008) 1–10
5. Dittrich, J., Quiané-Ruiz, J.A.: Efficient big data processing in hadoop mapreduce. Proceedings of the VLDB Endowment **5**(12) (2012) 2014–2015
6. Madden, S.: From databases to big data. Internet Computing, IEEE **16**(3) (2012) 4–6
7. Amazon Elastic Compute Cloud: http://aws.amazon.com/ec2/
8. Irwin, D., Chase, J., Grit, L., Yumerefendi, A., Becker, D., Yocum, K.G.: Sharing networked resources with brokered leases. resource **6** (2006) 6
9. Ciurana, E.: Developing with Google App Engine. Apress (2009)
10. Rackspace: http://www.rackspace.com
11. Windows Azure: http://www.windowsazure.com/
12. Bryant, R.E.: Data-intensive supercomputing: The case for disc. (2007)
13. Garg, S.K., Yeo, C.S., Anandasivam, A., Buyya, R.: Environment-conscious scheduling of hpc applications on distributed cloud-oriented data centers. Journal of Parallel and Distributed Computing **71**(6) (2011) 732–749
14. Gorton, I., Gracio, D.K.: Data-intensive computing: A challenge for the 21st century. Data-Intensive Computing: Architectures, Algorithms, and Applications (2012) 3
15. White, T.: Hadoop - The Definitive Guide. O'Reilly (2009)
16. Dean, J., Ghemawat, S.: Mapreduce: Simplified data processing on large clusters. In: OSDI. (2004) 137–150
17. Chen, Y.: Workload-driven design and evaluation of large- scale data-centric systems (May, 09 2012)
18. Reiss, C., Tumanov, A., Ganger, G.R., Katz, R.H., Kozuch, M.A.: Heterogeneity and dynamicity of clouds at scale: Google trace analysis. In: SoCC. (2012) 7
19. Macías, M., Guitart, J.: A genetic model for pricing in cloud computing markets. In: SAC, ACM (2011) 113–118
20. Niyato, D., Vasilakos, A.V., Zhu, K.: Resource and revenue sharing with coalition formation of cloud providers: Game theoretic approach. In: CCGRID, IEEE (2011) 215–224
21. Lin, W.Y., Lin, G.Y., Wei, H.Y.: Dynamic auction mechanism for cloud resource allocation. In: CCGRID, IEEE (2010) 591–592
22. Lucas-Simarro, J.L., Moreno-Vozmediano, R., Montero, R.S., Llorente, I.M.: Dynamic placement of virtual machines for cost optimization in multi-cloud environments. In: HPCS, IEEE (2011) 1–7

23. Wolf, J., Balmin, A., Rajan, D., Hildrum, K., Khandekar, R., Parekh, S., Wu, K.L., Vernica, R.: On the optimization of schedules for mapreduce workloads in the presence of shared scans. The VLDB Journal **21**(5) (2012) 589–609
24. Chang, H., Kodialam, M.S., Kompella, R.R., Lakshman, T.V., Lee, M., Mukherjee, S.: Scheduling in mapreduce-like systems for fast completion time. In: INFOCOM, IEEE (2011) 3074–3082
25. Wolf, J.L., Rajan, D., Hildrum, K., Khandekar, R., Kumar, V., Parekh, S., Wu, K.L., Balmin, A.: Flex: A slot allocation scheduling optimizer for mapreduce workloads. In: Middleware. (2010) 1–20
26. Kliazovich, D., Bouvry, P., Khan, S.U.: DENS: data center energy-efficient network-aware scheduling. Cluster Computing **16**(1) (2013) 65–75
27. Chen, Y., Alspaugh, S., Borthakur, D., Katz, R.H.: Energy efficiency for large-scale mapreduce workloads with significant interactive analysis. In: EuroSys, ACM (2012) 43–56
28. Wang, L., Khan, S.U.: Review of performance metrics for green data centers: a taxonomy study. The Journal of Supercomputing **63**(3) (2013) 639–656
29. Wang, L., Khan, S.U., Chen, D., Kolodziej, J., Ranjan, R., Xu, C.Z., Zomaya, A.Y.: Energy-aware parallel task scheduling in a cluster. Future Generation Comp. Syst **29**(7) (2013) 1661–1670
30. Isard, M., Prabhakaran, V., Currey, J., Wieder, U., Talwar, K., Goldberg, A.: Quincy: fair scheduling for distributed computing clusters. In: SOSP, ACM (2009) 261–276
31. Zaharia, M., Borthakur, D., Sarma, J.S., Elmeleegy, K., Shenker, S., Stoica, I.: Delay scheduling: a simple technique for achieving locality and fairness in cluster scheduling. In: EuroSys. (2010) 265–278
32. Borthakur, D., Gray, J., Sarma, J.S., Muthukkaruppan, K., Spiegelberg, N., Kuang, H., Ranganathan, K., Molkov, D., Menon, A., Rash, S., Schmidt, R., Aiyer, A.S.: Apache hadoop goes realtime at facebook. In: SIGMOD Conference. (2011) 1071–1080
33. Ousterhout, K., Wendell, P., Zaharia, M., Stoica, I.: Sparrow: Scalable scheduling for sub-second parallel jobs. Technical Report UCB/EECS-2013-29, EECS Department, University of California, Berkeley (April 2013)
34. Buyya, R., Yeo, C.S., Venugopal, S., Broberg, J., Brandic, I.: Cloud computing and emerging IT platforms: Vision, hype, and reality for delivering computing as the 5th utility. Future Generation Comp. Syst **25**(6) (2009) 599–616
35. Delimitrou, C., Kozyrakis, C.: Paragon: QoS-aware scheduling for heterogeneous datacenters. In: ASPLOS. (2013) 77–88
36. Vasic, N., Novakovic, D.M., Miucin, S., Kostic, D., Bianchini, R.: Dejavu: Accelerating resource allocation in virtualized environments architectural support for programming languages and operating systems, (17th ASPLOS'12). In: Proceedings of the 17th International Conference on, ACM Press (2012) 423–436
37. Zhu, X., Young, D., Watson, B.J., Wang, Z., Rolia, J., Singhal, S., McKee, B., Hyser, C., Gmach, D., Gardner, R., Christian, T., Cherkasova, L.: 1000 islands: an integrated approach to resource management for virtualized data centers. Cluster Computing **12**(1) (2009) 45–57
38. Kale, L.V., Kumar, S., Potnuru, M., DeSouza, J., Bandhakavi, S.: Faucets: Efficient resource allocation on the computational grid. In: Proceedings of the 2004 International Conference on Parallel Processing (33th ICPP'04), Montreal, Quebec, Canada, IEEE Computer Society (August 2004) 396–405
39. Rodero-Merino, L., Caron, E., Muresan, A., Desprez, F.: Using clouds to scale grid resources: An economic model. Future Generation Computer Systems **28**(4) (2012) 633 – 646
40. Kang, Z., Wang, H.: A novel approach to allocate cloud resource with different performance traits. In: Proceedings of the 2013 IEEE International Conference on Services Computing. SCC '13, Washington, DC, USA, IEEE Computer Society (2013) 128–135
41. Sim, K.M.: Towards complex negotiation for cloud economy. In: Advances in Grid and Pervasive Computing. Springer (2010) 395–406
42. Garg, S.K., Vecchiola, C., Buyya, R.: Mandi: a market exchange for trading utility and cloud computing services. The Journal of Supercomputing **64**(3) (2013) 1153–1174

43. Izakian, H., Abraham, A., Ladani, B.T.: An auction method for resource allocation in computational grids. Future Generation Comp. Syst **26**(2) (2010) 228–235
44. Zaman, S., Grosu, D.: Combinatorial auction-based allocation of virtual machine instances in clouds. In: CloudCom, IEEE (2010) 127–134
45. Samimi, P., Patel, A.: Review of pricing models for grid & cloud computing. In: Computers & Informatics (ISCI), 2011 IEEE Symposium on, IEEE (2011) 634–639
46. Wang, Q., Ren, K., Meng, X.: When cloud meets ebay: Towards effective pricing for cloud computing. In Greenberg, A.G., Sohraby, K., eds.: INFOCOM, IEEE (2012) 936–944
47. Meng, X., Isci, C., Kephart, J.O., Zhang, L., Bouillet, E., Pendarakis, D.E.: Efficient resource provisioning in compute clouds via VM multiplexing. In Parashar, M., Figueiredo, R.J.O., Kiciman, E., eds.: ICAC, ACM (2010) 11–20
48. Zhang, W., Qian, H., Wills, C.E., Rabinovich, M.: Agile resource management in a virtualized data center. In Adamson, A., Bondi, A.B., Juiz, C., Squillante, M.S., eds.: WOSP/SIPEW, ACM (2010) 129–140
49. Garg, S.K., Gopalaiyengar, S.K., Buyya, R.: SLA-based resource provisioning for heterogeneous workloads in a virtualized cloud datacenter. In Xiang, Y., Cuzzocrea, A., Hobbs, M., Zhou, W., eds.: ICA3PP (1). Volume 7016 of Lecture Notes in Computer Science., Springer (2011) 371–384
50. Urgaonkar, B., Shenoy, P., Chandra, A., Goyal, P.: Dynamic provisioning of multi-tier internet applications. In: Autonomic Computing, 2005. ICAC 2005. Proceedings. Second International Conference on, IEEE (2005) 217–228
51. Gong, Z., Gu, X., Wilkes, J.: Press: Predictive elastic resource scaling for cloud systems. In: Network and Service Management (CNSM), 2010 International Conference on, IEEE (2010) 9–16
52. Padala, P., Hou, K.Y., Shin, K.G., Zhu, X., Uysal, M., Wang, Z., Singhal, S., Merchant, A.: Automated control of multiple virtualized resources. In: Proceedings of the 4th ACM European conference on Computer systems, ACM (2009) 13–26
53. Xu, J., Zhao, M., Fortes, J., Carpenter, R., Yousif, M.: Autonomic resource management in virtualized data centers using fuzzy logic-based approaches. Cluster Computing **11**(3) (2008) 213–227
54. Gmach, D., Krompass, S., Scholz, A., Wimmer, M., Kemper, A.: Adaptive quality of service management for enterprise services. ACM Transactions on the Web (TWEB) **2**(1) (2008) 8
55. Beloglazov, A., Abawajy, J., Buyya, R.: Energy-aware resource allocation heuristics for efficient management of data centers for cloud computing. Future Generation Computer Systems **28**(5) (2012) 755–768
56. Xiong, K., Perros, H.G.: SLA-based resource allocation in cluster computing systems. In: IPDPS, IEEE (2008) 1–12
57. Gu, J., Hu, J., Zhao, T., Sun, G.: A new resource scheduling strategy based on genetic algorithm in cloud computing environment. Journal of Computers **7**(1) (2012) 42–52
58. Hu, J., Gu, J., Sun, G., Zhao, T.: A scheduling strategy on load balancing of virtual machine resources in cloud computing environment. In: Parallel Architectures, Algorithms and Programming (PAAP), 2010 Third International Symposium on, IEEE (2010) 89–96
59. Vavilapalli, V.K., Murthy, A.C., Douglas, C., Agarwal, S., Konar, M., Evans, R., Graves, T., Lowe, J., Shah, H., Seth, S., Saha, B., Curino, C., O'Malley, O., Radia, S., Reed, B., Baldeschwieler, E.: Apache hadoop YARN: Yet another resource negotiator. In: SoCC. (2013)
60. Zaharia, M., Borthakur, D., Sarma, J.S., Shenker, S., Stoica, I.: Job scheduling for multi-user mapreduce clusters. Technical Report No. UCB/EECS-2009-55, Univ. of Calif., Berkeley, CA (April 2009)
61. Zhang, X., Zhong, Z., Feng, S., Tu, B., Fan, J.: Improving data locality of mapreduce by scheduling in homogeneous computing environments. In: Parallel and Distributed Processing with Applications (ISPA), 2011 IEEE 9th International Symposium on, IEEE (2011) 120–126
62. Kc, K., Anyanwu, K.: Scheduling hadoop jobs to meet deadlines. In: Cloud Computing Technology and Science (CloudCom), 2010 IEEE Second International Conference on, IEEE (2010) 388–392

63. Tang, Z., Zhou, J., Li, K., Li, R.: MTSD: A task scheduling algorithm for mapreduce base on deadline constraints. In: IPDPS Workshops, IEEE Computer Society (2012) 2012–2018
64. Schwiegelshohn, U., Tchernykh, A.: Online scheduling for cloud computing and different service levels. In: Proc. 9th High-Performance Grid & Cloud Computing – 9th HPGC'12, Proc. IEEE International Parallel and Distributed Processing Symposium Workshops & PhD Forum (26th IPDPS'12), IEEE Computer Society (2012) 1067–1074
65. Venugopal, S., Buyya, R.: An scp-based heuristic approach for scheduling distributed data-intensive applications on global grids. Journal of Parallel and Distributed Computing **68**(4) (2008) 471–487
66. Chang, R.S., Chang, J.S., Lin, P.S.: An ant algorithm for balanced job scheduling in grids. Future Generation Computer Systems **25**(1) (2009) 20–27
67. Kolodziej, J., Khan, S.U., Xhafa, F.: Genetic algorithms for energy-aware scheduling in computational grids. In: P2P, Parallel, Grid, Cloud and Internet Computing (3PGCIC), 2011 International Conference on, IEEE (2011) 17–24
68. Lee, Y.H., Leu, S., Chang, R.S.: Improving job scheduling algorithms in a grid environment. Future generation computer systems **27**(8) (2011) 991–998
69. Samuel, T.K., Baer, T., Brook, R.G., Ezell, M., Kovatch, P.: Scheduling diverse high performance computing systems with the goal of maximizing utilization. In: High Performance Computing (HiPC), 2011 18th International Conference on, IEEE (2011) 1–6
70. Balman, M.: Failure-awareness and dynamic adaptation in data scheduling (November 14 2008)
71. Chowdhury, M., Zaharia, M., Ma, J., Jordan, M.I., Stoica, I.: Managing data transfers in computer clusters with orchestra. In: SIGCOMM, ACM (2011) 98–109
72. Seo, S., Jang, I., Woo, K., Kim, I., Kim, J.S., Maeng, S.: Hpmr: Prefetching and pre-shuffling in shared mapreduce computation environment. In: Cluster Computing and Workshops, 2009. CLUSTER'09. IEEE International Conference on, IEEE (2009) 1–8
73. Çatalyürek, Ü.V., Kaya, K., Uçar, B.: Integrated data placement and task assignment for scientific workflows in clouds. In: Proceedings of the fourth international workshop on Data-intensive distributed computing, ACM (2011) 45–54
74. Xie, J., Yin, S., Ruan, X., Ding, Z., Tian, Y., Majors, J., Manzanares, A., Qin, X.: Improving mapreduce performance through data placement in heterogeneous hadoop clusters. In: Parallel & Distributed Processing, Workshops and Phd Forum (IPDPSW), 2010 IEEE International Symposium on, IEEE (2010) 1–9
75. Zeng, W., Zhao, Y., Ou, K., Song, W.: Research on cloud storage architecture and key technologies. In: Proceedings of the 2nd International Conference on Interaction Sciences: Information Technology, Culture and Human, ACM (2009) 1044–1048
76. Abad, C.L., Lu, Y., Campbell, R.H.: DARE: Adaptive data replication for efficient cluster scheduling. In: Proc. '11 IEEE International Conference on Cluster Computing (13th CLUSTER'11), Austin, TX, USA, IEEE Computer Society (September 2011) 159–168
77. Castillo, C., Tantawi, A.N., Arroyo, D., Steinder, M.: Cost-aware replication for dataflows. In: NOMS, IEEE (2012) 171–178
78. Chervenak, A.L., Deelman, E., Livny, M., Su, M.H., Schuler, R., Bharathi, S., Mehta, G., Vahi, K.: Data placement for scientific applications in distributed environments. In: GRID, IEEE Computer Society (2007) 267–274
79. Chen, Y., Ganapathi, A.S., Griffith, R., Katz, R.H.: Analysis and lessons from a publicly available google cluster trace. Technical Report UCB/EECS-2010-95, EECS Department, University of California, Berkeley (Jun 2010)
80. Chen, Y., Ganapathi, A.S., Griffith, R., Katz, R.H.: Towards understanding cloud performance tradeoffs using statistical workload analysis and replay. University of California at Berkeley, Technical Report No. UCB/EECS-2010-81 (2010)
81. Stuer, G., Vanmechelen, K., Broeckhove, J.: A commodity market algorithm for pricing substitutable grid resources. Future Generation Comp. Syst **23**(5) (2007) 688–701
82. Teng, F., Magoulès, F.: Resource pricing and equilibrium allocation policy in cloud computing. In: CIT, IEEE Computer Society (2010) 195–202

83. Eymann, T., Reinicke, M., Villanueva, O.A., Vidal, P.A., Freitag, F., Moldes, L.N.: Decentralized resource allocation in application layer networks. In: CCGrid, IEEE (May 12 2003) 645–650
84. Padala, P., Harrison, C., Pelfort, N., Jansen, E., Frank, M.P., Chokkareddy, C.: OCEAN: The open computation exchange and arbitration network, A market approach to meta computing. In: Proc. 2nd International Symposium on Parallel and Distributed Computing (2nd ISPDC'03), Ljubljana, Slovenia, IEEE Computer Society (October 2003) 185–192
85. Peterson, L., Anderson, T., Culler, D., Roscoe, T.: PlanetLab: A Blueprint for Introducing Disruptive Technology into the Internet. In: First ACM Workshop on Hot Topics in Networks, Association for Computing Machinery (October 2002) Available from http://www.planet-lab.org/pdn/pdn02-001.pdf.
86. Ghodsi, A., Zaharia, M., Hindman, B., Konwinski, A., Shenker, S., Stoica, I.: Dominant resource fairness: Fair allocation of multiple resource types. Technical report, University of California, Berkeley (2011)
87. Mihailescu, M., Teo, Y.M.: Dynamic resource pricing on federated clouds. In: CCGRID, IEEE (2010) 513–517
88. Dutreilh, X., Rivierre, N., Moreau, A., Malenfant, J., Truck, I.: From data center resource allocation to control theory and back. In: Proc. IEEE International Conference on Cloud Computing (3rd IEEE CLOUD'10). (2010) 410–417
89. Buyya, R., Garg, S.K., Calheiros, R.N.: SLA-oriented resource provisioning for cloud computing: Challenges, architecture, and solutions. In: Cloud and Service Computing (CSC). (January 21 2012)
90. Gandhi, A., Chen, Y., Gmach, D., Arlitt, M.F., Marwah, M.: Minimizing data center SLA violations and power consumption via hybrid resource provisioning. In: IGCC, IEEE Computer Society (2011) 1–8
91. Birke, R., Chen, L.Y., Smirni, E.: Data centers in the cloud: A large scale performance study. In: Proc. 2012 IEEE Fifth International Conference on Cloud Computing (5th IEEE CLOUD'12). (June 2012) 336–343
92. Gao, Y., Rong, H., Huang, J.Z.: Adaptive grid job scheduling with genetic algorithms. Future Generation Computer Systems **21**(1) (2005) 151–161
93. Fidanova, S.: Simulated annealing for grid scheduling problem. In: Modern Computing, 2006. JVA'06. IEEE John Vincent Atanasoff 2006 International Symposium on, IEEE (2006) 41–45
94. neng Chen, W., 0003, J.Z.: An ant colony optimization approach to a grid workflow scheduling problem with various qoS requirements. IEEE Transactions on Systems, Man, and Cybernetics, Part C **39**(1) (2009) 29–43
95. Braun, T.D., Siegel, H.J., Beck, N., Bölöni, L., Maheswaran, M., Reuther, A.I., Robertson, J.P., Theys, M.D., Yao, B., Hensgen, D.A., Freund, R.F.: A comparison of eleven static heuristics for mapping a class of independent tasks onto heterogeneous distributed computing systems. J. Parallel Distrib. Comput **61**(6) (2001) 810–837
96. Dong, F., Akl, S.G.: Scheduling algorithms for grid computing: State of the art and open problems. School of Computing, Queens University, Kingston, Ontario (2006)
97. Ren, Z., Wan, J., Shi, W., Xu, X., Zhou, M.: Workload analysis, implications and optimization on a production hadoop cluster: A case study on taobao. IEEE Transactions on Services Computing (2013)
98. Isard, M., Budiu, M., Yu, Y., Birrell, A., Fetterly, D.: Dryad: distributed data-parallel programs from sequential building blocks. In: EuroSys, ACM (2007) 59–72
99. Zaharia, M., Chowdhury, M., Franklin, M.J., Shenker, S., Stoica, I.: Spark: cluster computing with working sets. In: Proceedings of the 2nd USENIX conference on Hot topics in cloud computing. (2010) 10–10
100. Sandholm, T., Lai, K.: Dynamic proportional share scheduling in Hadoop. In Frachtenberg, E., Schwiegelshohn, U., eds.: Job Scheduling Strategies for Parallel Processing. Springer Verlag (2010) 110–131
101. Wang, L., von Laszewski, G., Dayal, J., He, X., Younge, A.J., Furlani, T.R.: Towards thermal aware workload scheduling in a data center. In: ISPAN, IEEE Computer Society (2009) 116–122

102. Ranganathan, K., Foster, I.T.: Decoupling computation and data scheduling in distributed data-intensive applications. In: HPDC, IEEE Computer Society (2002) 352–358
103. Guo, D., Li, M., Jin, H., Shi, X., Lu, L.: Managing and aggregating data transfers in data centers (2013)
104. Al-Fares, M., Radhakrishnan, S., Raghavan, B., Huang, N., Vahdat, A.: Hedera: Dynamic flow scheduling for data center networks. In: NSDI, USENIX Association (2010) 281–296
105. Sun, N.H., Xing, J., Huo, Z.G., Tan, G.M., Xiong, J., Li, B., Ma, C.: Dawning nebulae: a petaflops supercomputer with a heterogeneous structure. Journal of Computer Science and Technology **26**(3) (2011) 352–362
106. : Top500 list
107. Lumb, I., Smith, C.: Scheduling attributes and platform lsf. In: Grid resource management. Springer (2004) 171–182
108. Taobao: http://www.taobao.com
109. Chaiken, R., Jenkins, B., Larson, P.Å., Ramsey, B., Shakib, D., Weaver, S., Zhou, J.: Scope: easy and efficient parallel processing of massive data sets. Proceedings of the VLDB Endowment **1**(2) (2008) 1265–1276
110. Ghemawat, S., Gobioff, H., Leung, S.T.: The google file system. In: ACM SIGOPS Operating Systems Review. Volume 37., ACM (2003) 29–43
111. Shvachko, K., Kuang, H., Radia, S., Chansler, R.: The hadoop distributed file system. In: Mass Storage Systems and Technologies (MSST), 2010 IEEE 26th Symposium on, IEEE (2010) 1–10
112. Agarwal, S., Dunagan, J., Jain, N., Saroiu, S., Wolman, A., Bhogan, H.: Volley: Automated data placement for geo-distributed cloud services. In: NSDI. (2010) 17–32
113. McKeown, N.: Software-defined networking. INFOCOM keynote talk, Apr (2009)
114. Liu, D., Lee, Y.H.: Pfair scheduling of periodic tasks with allocation constraints on multiple processors. In: IPDPS. (2004)
115. Lee, J., Easwaran, A., Shin, I.: LLF schedulability analysis on multiprocessor platforms. In: IEEE Real-Time Systems Symposium. (2010) 25–36
116. Islam, S., Keung, J., Lee, K., Liu, A.: Empirical prediction models for adaptive resource provisioning in the cloud. Future Generation Computer Systems **28**(1) (2012) 155–162
117. Wilkes, J., Reiss, C.: Details of the clusterdata-2011-1 trace (2011)