Cluster Comput (2016) 19:1089–1103
DOI 10.1007/s10586-016-0593-6

P-Aware: a proportional multi-resource scheduling strategy in cloud data center

Hang Zhou[1] · Qing Li[1] · Weiqin Tong[1] · Samina Kausar[1] · Hai Zhu[2]

Received: 16 February 2016 / Revised: 19 June 2016 / Accepted: 25 June 2016 / Published online: 9 July 2016

Abstract Concentrating on a single resource cannot efficiently cope with the overall high utilization of resources in cloud data centers. Nowadays multiple resource scheduling problem is more attractive to researchers. Some studies achieve progresses in multi-resource scenarios. However, these previous heuristics have obvious limitations in complex software defined cloud environment. Focusing on energy conservation and load balancing, we propose a preciousness model for multiple resource scheduling in this paper. We give the formulation of the problem and propose an innovative strategy (P-Aware). In P-Aware, a special algorithm PMDBP (Proportional Multi-dimensional Bin Packing) is applied in the multi-dimensional bin packing approach. In this algorithm, multiple resources are consumed in a proportional way. Structure and details of PMDBP are discussed in this paper. Extensive experiments demonstrate that our strategy outperforms others both in efficiency and load balancing. Now P-Aware has been implemented in the resource management system in our cooperative company to cut energy consumption and reduce resource contention.

✉ Hang Zhou
zhouhang.shu@gmail.com

Qing Li
qli@shu.edu.cn

Weiqin Tong
profweiqin@sina.com

Samina Kausar
saminamalik7@yahoo.com

Hai Zhu
zhuhai@mail.xjtu.edu.cn

[1] School of Computer Engineering and Science, Shanghai University, Shanghai 200444, China

[2] School of Computer Science and Technology, Xi'An JiaoTong University, Xi'An 710049, Shanxi, China

Keywords Cloud data center · energy conservation · load balancing · multiple dimensions · multiple resource scheduling · bin-packing · SDCloud

1 Introduction

Cloud computing has emerged as a new computing paradigm with rapid development and prosperous future. Benefiting from virtualization and other key technologies, traditional data centers have been transformed into cloud data centers (CDCs), to achieve high flexibility in hardware resource usage. Now more and more corporations' applications have been migrated to the cloud (private or public). However, energy consumed by CDC which is equipped with massive hardware resources has been a heavy burden. Consequently, the rapid increases in applications demand and reasonable management of large scale cluster require more efficient strategy to achieve high utilization of massive hardware in CDC.

Resource allocation is a good way to make better use of hardware resources. It tries to deal with the problem how to share the limited resources among different users with fairness (or weighted fairness sometimes). In the beginning, servers in cluster are recognized as a single resource. A widely used algorithm to achieve fairness with single resource is Max-Min [1], which is an attempt to increase the allocation of some other participants with fairness. With the growing attention to multiple resource consumption models, another famous resource allocation algorithm DRF (dominant resource fairness) appeared in which each user's dominant share (the maximum ratio of any resource that the user has been allocated in a server) is equalized. Because of its highly desirable fairness properties such as sharing incentive, strategy-proofness, envy-freeness, Pareto efficiency, etc.

[2] DRF has received significant attention in literature [3–5]. Different from traditional parallel computing frameworks such as Hadoop [6] in which "slot" share is applied, Mesos [7], a state of the art resource management system achieves fine-grained multi-resource scheduling based on DRF [8]. Though DRF and its subsequent work address the problem of multi-resource sharing, the application scenario of these theories is that the resource capacity of HPC or cluster is much shorter than the total resource demand, and has to reduce users' possessive resources to an appropriate proportion for fairness. However, this assumption does not always conform to the actual condition. Most private CDCs deploy sufficient hardware resources to ensure fast and stable service. In public cloud especially in IaaS mode, CDC is regarded as an unlimited resource pool and every user can get the resource allocation as they need. The above-mentioned algorithms and methods cannot fit these actual scenarios very well.

Besides resource allocation, resource scheduling (sometimes job scheduling in another perspective) is another main research area to achieve efficient utilization of multiple resources in CDC. Under the premise of sufficient hardware resources, jobs in queue can obtain adequate resources as they need. It concentrates on the underlying resource assignment issue which tries to make the optimal mapping decision between servers in cluster and jobs in queue. Though the elastic demand and supply is convenient to users, details of underlying resource scheduling are rather complex and need dynamic adaptive capacity. Some research work has been done in this area. S. Di et al. designed PID-CAN (Proactive Index Diffusion CAN) algorithm to make the mapping between users and servers in p2p network on cloud. [9,10] focused on the scheduling details for HPC and achieves quite good performance in load balancing. Artificial Neural Networks is applied in [11] to make the distributed scheduling between resources and virtual machines. M.A. Vasile et al. put forward a novel algorithm HySARC in which the available resources are clustered before the scheduling starts [12,13].

Although above-mentioned scheduling strategies perform well in some specific scenarios, they can hardly cope with the variant and complex cloud environment. Nowadays, Scheduling strategies in CDCs need to be flexible and scalable enough to adapt the frequent changes on resource configuration. The state of the art form (Software-Defined) is to separate control plane from infrastructure layer. Software-Defined Networking (SDN), for instance, simplifies the network management by separating the control plane from the data plane [14,15]. SDstorage, SDCompute and SDSecurity [16–18] also follow the same way and eventually evolved into SDCloud. In the Software-Defined Cloud (SDCloud) architecture (Fig. 1), various types of hardware resources are abstracted into resource pool and all of the forecasting, monitoring and scheduling issues are realized in the control layer [19]. Although users on IaaS cloud have no concern with internal mechanism of CDC, it does not mean the underlying resource scheduling processes are non-existent. The concealed mapping issue between applications and infrastructure layer face greater changes in the complex SDCloud system.

This paper focuses on this mapping issue between jobs (application layer) and servers (infrastructure layer) in CDC. There are some other properties or challenges such as SLA (service-level agreement) [20], task deadline [21], we consider the minimum number of servers and efficient load balancing as the primary goals of our strategy as less active servers consume less energy while load balancing can reduce resource-traffic. Bin-Packing is the most common way to deal with the optimal mapping problem. Single and multi-capacity bin packing problems with generalized scheduling problem have been studied in [22–26].

Based on a novel multi-capacity bin packing algorithm we developed, this paper proposes P-Aware: an efficient multi-dimensional resource integrated scheduling strategy. In this paper to solve the scheduling problem, P-Aware manages to assign heterogeneous application jobs to proper server nodes in cluster in CDC. This heuristic strategy is an approximation of progressive filling in which the usage of all resources increases at a similar rate or one dimension after another.

The rest of the paper is organized as follows. In Sect. 2, we present the related work in multiple resource consumption models, and propose 3 motivational application scenarios. Section 3 is about the preciousness model and problem formulation. The architecture and corresponding algorithm will be described in Sect. 4. Experiments and performance evaluation is in Sect 5, and we draw the conclusion in Sect. 6.

2 Related work and motivation

2.1 Related work

CPU and memory are traditionally the two considerations in multi-resource scenarios. Some existing studies [27–30] address how to achieve higher utilization for energy conservation in CDC. These algorithms try to improve CPU utilization to reach the threshold below which the performance degradation of application jobs can hardly happen, thus it reduces energy consumption because fewer cluster nodes are required. Studies [27,31–33] concentrate on the scheduling strategy with memory consumption. In these scenarios, large amount of memory-intensive jobs appear in job queue and resource demand competition exists among application jobs. Other studies [34,35] on parallel computing model (e.g. Map-Reduce) have completed some experiments and conclude that disk I/O is the prominent bottleneck when read-write operation of massive intermediate data are executed frequently. They try to find the efficient ways of sharing

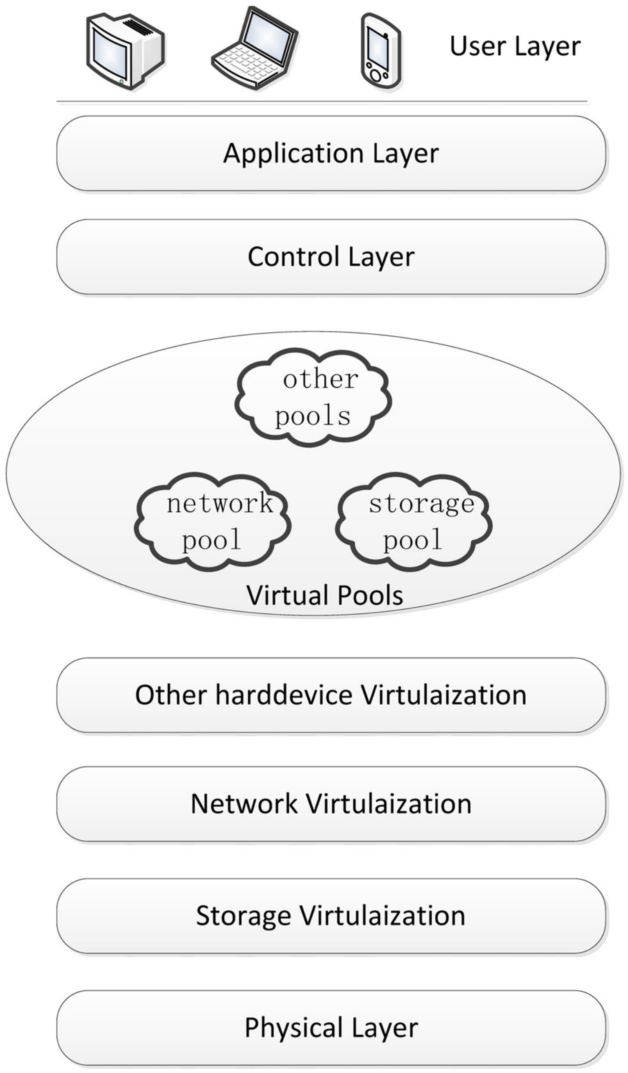

Fig. 1 Main layers of SDCloud architecture on IaaS Cloud

the disk I/O resource. More recently, due to the increasing concerns on emerging bandwidth intensive applications, bandwidth requirements are also taken into account when computing the resource placement [36,37]. While aforementioned research generates high performance in resource allocation and scheduling results, they have obvious limitations in the context of CDC, especially in SDCloud. Firstly, most of these schemes typically consider either a one-dimensional resource type or treat all resources as one abstract capacity when determining the sequence or combination of applications. Such resource models do not adequately capture the fact that application jobs consume simultaneously a variety of resources including CPU, memory, storage and network bandwidth, etc. Secondly, although a few studies take into account diverse resources in scheduling, they only individually extend the constraints from one to multiple dimensions and set constraint value for each dimensional resource. As a result, they consider only one-dimensional resource scheduling problem with multiple constrains and cannot handle the scenario of various application jobs consuming multiple resources simultaneously very well. The last but most important, there is some research which has focused on the scheduling strategy of multi-dimensional resource models in a CDC. They have made some progress in coordinating the consumption of multiple resources [38–40].

However, they lack two important characteristics which are the key features to be applied in various CDCs. One is characteristic of a CDC. CPU, memory, disk I/O, network bandwidth, etc. may have different configurations in different CDCs, furthermore, the application types may also differ a lot thus need different resource quotas. Only characteristic should be taken into full consideration, it can be widely applied to various scenarios in CDC. The other is time-variance. Both the resource configuration and the resource demand from applications fluctuate over time. Resource configuration may change dramatically in SDCloud environment and these conventional scheduling strategies will fail in the continuously changing context. Algorithms or strategy without this characteristic can hardly be applied to SDCloud scenarios. Therefore, providing a multi-dimensional resource scheduling strategy with fine-grained and characteristic property has more practical significance in CDC [41]. Our model fits these conditions well and the strategy we propose outperforms others both in energy conserving and load balancing. Furthermore, our strategy is designed to be lightweight, flexible and scalable enough in order to follow the SDCloud norm.

2.2 Motivational scenarios

Here we present motivational scenarios. Firstly we set the quantitative capacity value for each dimensional resource in a server machine. Then, considering the complexity of CDC environment, we put forward three scenarios in turn from simple to complex. These three scenarios will gradually depict CDC's actual situation which are involved in our strategy.

Assume a PM (we use "server", "server node", "physical machine" and PM interchangeably in this paper) consists of 4 types of resources: CPU, memory, disk I/O and network bandwidth. To define the problem more clearly, here we ignore the unit of measurement and the method we adopt to normalize the proportion of value in multiple resources. We use vector <12, 12, 12, 12> to represent the capacity value for C, M, D, N (the corresponding abbreviated symbols for CPU, memory, disk I/O and network bandwidth) in a PM. This means that the upper bound value for these 4 types of resources is 12. It is noteworthy that the total resource demand of jobs hosted in a shared PM cannot exceed this value. For simplicity we assume that there are 8 jobs in the job

Table 1 Resource demand of jobs in the queue

ID	C	M	D	N
1	6	6	1	1
2	5	4	4	5
3	4	3	4	5
4	2	1	1	2
5	1	2	1	1
6	2	4	2	7
7	3	4	6	7
8	1	3	1	6

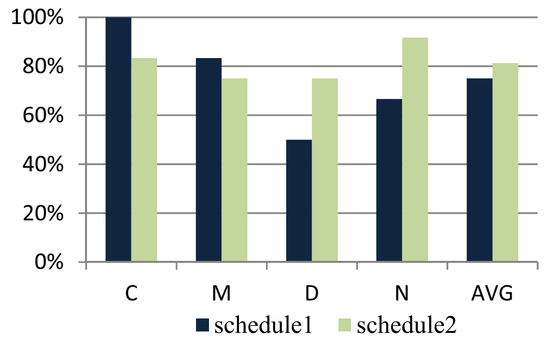

Fig. 2 Multi-resource utilization in scenario 1

queue Q. Table 1 shows the corresponding 8 vectors which represent the resource demand of 8 jobs in the queue. In order to expose the scheduling problem more intuitively, we choose "large" jobs compared with the PM capacity.

2.3 Average multi-resource utilization

In the first scheduling, job1, job3 and job4 are packed together to PM 1 as they obtain the upper boundary 100 % in utilization of C. However, the utilizations of other three resources in PM 1 are not high, especially in D and N (50 and 75 % respectively). In order to make full use of each type of resource, the average multi-resource utilization should be considered. As a comparison, the combination of job2, job3 and job5 in scheduling 2 achieves higher average utilization (81.3 vs. 75 %) and better performance in load balancing which is illustrated in Fig. 2.

2.4 Resource preciousness

In general, the performance of schedule 2 is satisfying for its rather high average utilization. However, it is based on a precondition that all types of resources have equal priority. Most of the previous research hold this precondition, but this is not always the case in actual application scenarios in CDC. Sometimes high-performance hardware equipment (such as mass memory or SSD) may be very expensive and thus it is relative precious compared with other resources.

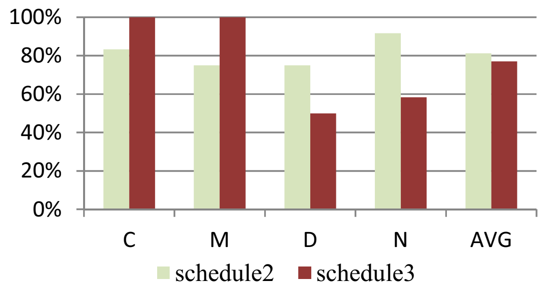

Fig. 3 Multi-resource utilization in scenario 2

Sometimes the total demand of a particular resource jumps within a particular time. For example, heavy disk I/O requests always appear between 11:20 AM and 12:00 AM. Disk I/O are more precious than others in these 40 minutes each workday. To these above-mentioned precious resources, higher priority should be given. Although most of time they will not confront shortage in CDC, we still have to try to improve their utilization in each PM, otherwise more PM would be activated and more energy would be consumed. As a result, trying to improve utilization of precious resource is one of our criteria.

In scenario 2, we assume that M is the precious resource, as compared to C, D and N. Therefore the importance of M should be fully considered to avoid wastage. Although schedule 2 has rather high average utilization of multi-resource, yet its utilization of M is only 75 %. This luxury of wasting precious resources will significantly lower the overall performance of scheduling in CDC. As a result, the preciousness of resource should be taken into account in scheduling algorithm. In contrast, schedule 3 which consists of job 1, job 2 and job 5 reaches 100 % utilization in M, at the same time it almost maintains quite high average utilization which is shown in Fig. 3. Schedule 3 makes full use of precious resource at the expense of minor decline in resource average utilization, thus it is a better way.

2.5 Global optimization

For C, M, D and N, the accumulative additions of resource demand from 8 jobs in Table 1 are 24, 27, 20 and 34 respectively. Therefore M and N are the two precious resources in Q when other data remains the same. We also make a comparison test in this scenario, schedule 4 pursues high utilization of M while schedule 5 emphasizes N. As the multi-dimensional bin packing problem is APX-hard, simple heuristic algorithm is applied to achieve higher utilization of precious resource. Firstly, job 1, job 2 and job 5 are hosted in PM 1 with 100 % utilization of M. Then the combination of job 3 and job 6 are mapped into PM 2. As the upper bound of each resource is 12 in a PM, therefore, PM 2 cannot hold any job as utilization of N has reached 100 %. Then PM 3 hosts job 4 and job 7 while

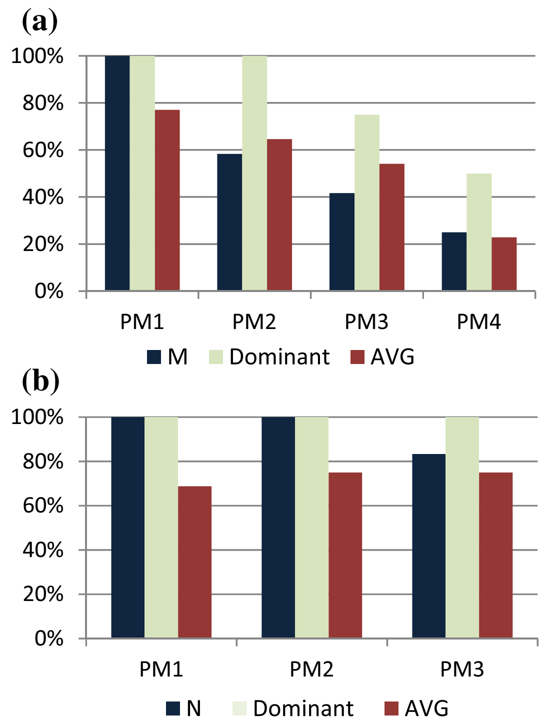

Fig. 4 **a** Bin Packing Result of schedule. **b** Bin Packing Result of schedule 5

another PM is needed for the remaining job 8. Compared with schedule 4, schedule 5 only uses 3 PMs. The corresponding job sets are <job2, job6>, <job3, job7>, <job1, job4, job5, job8>. Fig. 4 shows the comparison results of schedule 4 and schedule 5. The bar of "Dominant" means the highest utilization rate of a single resource among the multi-resource set.

Schedule 5 in Fig. 4b maximizes utilization of N in each PM. Meanwhile it achieves rather high average utilization. In contrast, the average utilization rate is far lower than that of the dominant resource in schedule 4 in Fig. 4a. This means that the multiple resources are not consumed in a cooperative proportion, therefore it leads to resource wastage. As a result, both the preciousness of resource and the cooperative way of multi-resource consumption play an important role in the optimal resource scheduling.

Focusing on these two above–mentioned factors, this paper firstly gives the preciousness model. This model has two characteristics: one is the ability to adapt to different resource configurations in SDCloud environment. The other is time-variant which can induce dynamic response in order to better cope with the workload fluctuations in real time. We also formulate the problem in Sect. 3.2 and propose an innovative strategy to solve this problem in Sect. 4. Multi-dimensional, time-variant, regardless of configuration, the combination of these three features makes our model and strategy more suitable for actual scheduling scenarios in CDC.

3 Preciousness model and problem formulation

3.1 Preciousness model

In order to characterize the preciousness degree of each type of resource, their quantity of demand and supply in scheduling should be taken into consideration. We use a Demand-Supply model (Eq. 1) to fix the preciousness value. Supply represents the resource configuration of a CDC, while Demand reflects current total workload in job queue. When the supply value remains the same, the greater the demand for this resource, the higher will be the value of its preciousness. On the other hand, when the demand value remains the same, the smaller the supply for this resource, the higher will be the value of its preciousness. Greater preciousness value means that the resource is more precious and higher utilization of this resource should be achieved in resource scheduling.

$$\text{Preciousness} = \text{Demand}/\text{Supply} \tag{1}$$

In general, we assume that there are m types of resources in each PM. We denote R (1, 2... m) as the set of m resources. PRS_i is used to present the preciousness value of resource i,i∊ R. Let J (1, 2... n) be the set of jobs in the queue Q, where D_{ji} means the demand value of resource i for job j, i∊ R, j∊ J. Therefore, the job queue Q's overall demand of resource i equals the summation of demand value of jobs in J (Eq. 2).

$$D_{Qi} = \sum_{j\in J} D_{\text{ji}} i = 1, 2...\text{m}. \tag{2}$$

Though the demand value of multi-resource is available, it is challenging to fix a proper granularity for the metric of supply value. There are obvious flaws in previous studies regarding mismatching of complex scenarios in real CDC. For instance, X. Sun, et al. used VM (virtual machine) as the basic unit of resource supply [38]. In fact, it is hard to compute the accurate supply quantity for a VM, because it fluctuates with the adjustment of hardware resource quota in VMware, Xen and other virtual platforms [42]. For example, the VMware vSphere DRS dynamically adjusts VM's resources by three parameters (Reservation, Shares, Limit). Resource contention is another important factor which cannot be ignored. It will reduce the real amount of resource provided by a VM [43]. In this paper, we use PM as the basic unit of resource supply because the resource quantity in a PM is stable and its performance parameters are available from its configuration instructions. Here the mapping issue between PMs and jobs is the fundamental problem in multi-dimensional bin packing scenarios. Moreover, VMs in virtual pools cannot be considered as the resource headstream in SDCloud scheduling. The central control layer in Fig. 1 has to monitor the resource status in physical infrastructure layer and PMs are the original resource provider in scheduling strategy in SDCloud.

We denote P (1, 2... k) as the set of PMs needed in scheduling. Let C_{pi} be the supply value of resource i in PM p, where pϵ P. Taking into account the homogeneous architecture of modern CDCs, PMs in P are assumed to have the identical resource capacities (our algorithm based on preciousness model also works in heterogeneous cluster with some adjustments, we propose homogeneous cluster here to explain the P-Aware strategy more clearly). Therefore, $C_i=C_{1i}=C_{2i}=...=C_{ki}$, we denote C_i as the corresponding supply value of resource i in a PM. Combining (1) and (2) we get:

$$\forall i \in R, PRS_i = \frac{\sum_{j \in J} D_{ji}}{C_i} \tag{3}$$

As we can see from Eq. 3, for each resource i, the denominator C_i represents the resource configuration feature of a CDC. A large public CDC equipped with bandwidth (100G) differs a lot from a small private CDC with bandwidth (1G). Even for the same CDC, resource allocation also fluctuates over time. This model fully considers these factors and can be easily applied to various CDCs. The numerator in Eq. 3, on the other hand, reflects total resource requirement from applications in a CDC. Different CDCs have different situations. A private CDC whose main service is cloud storage has heavy disk I/O requests, while the workload in a supercomputing CDC is CPU-intensive. In addition, even for the same CDC, workload always fluctuates greatly over time: weekdays differ from holidays, morning differs from evening. In Eq. 3, the current resource demand in job queue is taken into account which grants this model ability in dynamic adjustment.

3.2 Problem formulation

For PM set P, we denote |P| as the number of PMs needed in scheduling. Therefore, it is the global goal to minimize |P| in this scheduling. Meanwhile, there are some restrictions as follows. Firstly, for each PM p, the accumulation addition of demand vector of jobs which are mapped into p cannot exceed the resource capacity of p.

$$\forall i \in R, p \in P, \sum_{j \in U} D_{ji} <= C_{pi} \tag{4}$$

Here U is the subset of J which is submitted to PM p.

Secondly, if we gather all jobs mapped into the PM set P, we get the job queue Q. In other words, the accumulative addition of preciousness value of each PM p must be equal to the preciousness value of job queue Q:

$$\forall i \in R, p \in P, \sum_{p \in P} \mathrm{PRS}_{pi} = \mathrm{PRS}_i \tag{5}$$

Here PRS_{pi} is denoted as the preciousness value of resource i in PM p.

Therefore, by combining the above formulas, the P-Aware scheduling problem can be formulated as follows:

$$Minimize|P|$$
$$s.t. \begin{cases} \forall i \in R, p \in P, \sum_{j \in U} D_{ji} <= C_{pi} \\ \forall i \in R, p \in P, \sum_{p \in P} \mathrm{PRS}_{pi} = PRS_i \end{cases} \tag{6}$$

Equation 6 is a typical multi-dimensional bin-packing problem with multi-dimensional constraints. Both energy conservation and load balancing are the optimization goals in our strategy. Considering the complexity of this problem, only energy conservation appears in problem formulation. The improvement in load balancing, however, is derived from the special packing way which is described in Sect. 4.

4 P-Aware: architecture and algorithm

In this section we give the architecture (Fig. 5) of P-Aware, and try to explain the core algorithm in our strategy.

4.1 Architecture

To consider P-Aware's behavior, we assume that there are n input items in the job queue Q. Scheduler is the core component in middle layer, and we gradually approach S from top and bottom respectively. Analyzer is in top while Monitor is in bottom, all of three components will be described briefly here and the details of core algorithm will be described in Sect. 4.2.

Analyzer This module is used as a background process. When a job j arrives at Q, the job type and corresponding parameters are collected by Analyzer. Analyzer will calculate the resource demand value in each dimension for job j. In other words, we will get the vector D_{ji} (i=1, 2 ... m) as soon as job j arrives. When Q is formed in the batch mode, the demand matrix D_{ji} (i=1, 2... m; j=1, 2... n) and the vector PRS_i (i=1, 2... m) is available for next step. Resource demand prediction work is already very mature based on neural network and it will not be the obstacle in Analyzer [44–46].

Monitor Monitor is designed for monitoring resource that changes in each PM [47]. Each time a new job is added to a PM, the multi-resource demand will increase in this PM. We use Monitor to perceive the changes of multi-resource in each PM and update the demand vector D_{pi}(i=1, 2... m; p=1, 2... k) when a job is submitted to PM p (Table 2).

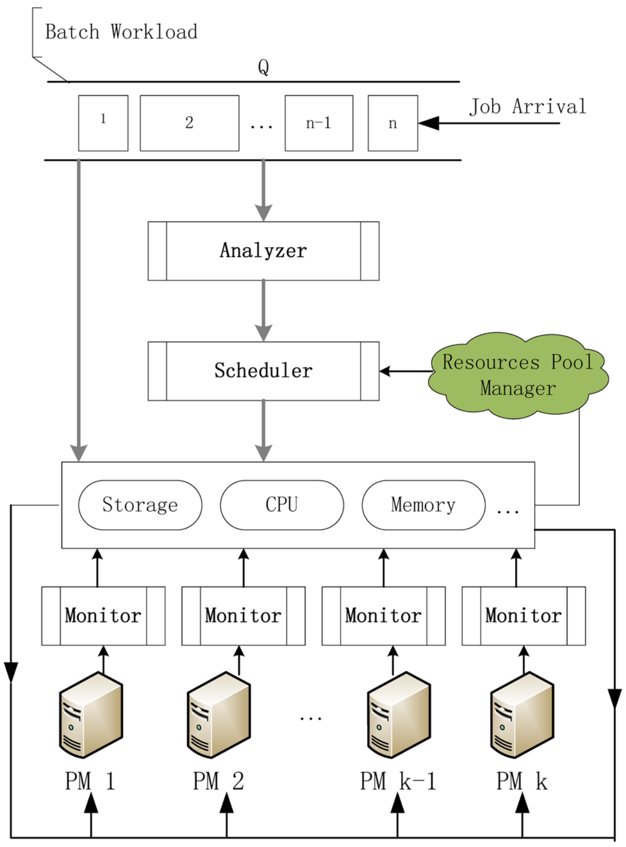

Fig. 5 Architecture of P-Aware Strategy

Table 2 Notations used in P-Aware and PMDBP

Notations	Definition
R	The set of multiple resources
P	The set of servers needed in job scheduling
Q	The job queue
J	The job set in Q
U	The subset of J which are hosted by server p
D_{Qi}	The demand value of resource i in Q
D_{pi}	The demand value of resource i in server p
D_{ji}	The demand value of resource i with job j
C_i	The supply value of resource i in a PM
PRS_i	The preciousness value of resource i
OP	Optimal proportion of multiple resources in Q
OP_i	Optimal proportion of resource i in Q
AOP	actual Optimal proportion of multiple resources in a PM
AOP_i	Actual optimal proportion of resource I in current PM
Category i	The i-intensive job cluster
Sequence i	The descending sort result of category i

Scheduler In this core component, we firstly classify jobs by their dominant resource and thus jobs in Q are divided into m classes. Then the strategy starts multi-dimensional bin packing phase. Considering the complexity of this mapping issue, we propose a novel bin-packing algorithm: PMDBP (Proportional Multi-dimensional Bin Packing) to map the next proper job into p.

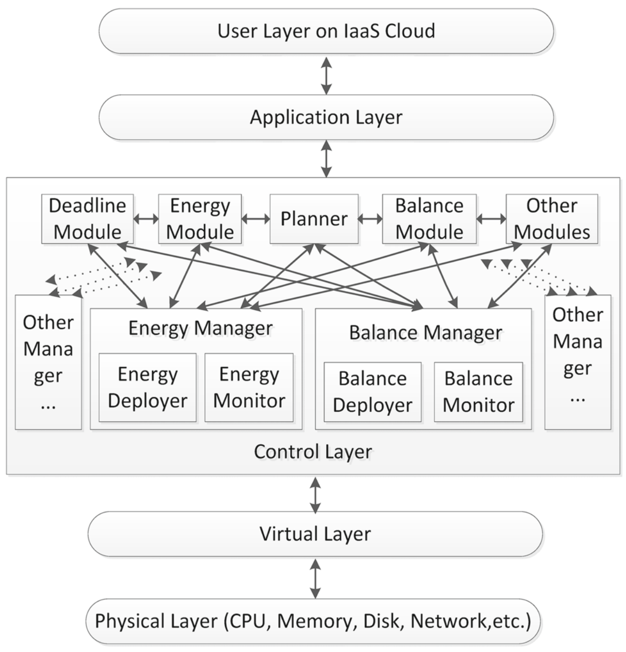

Fig. 6 A general control laer in SDCloud Architecture

The most potential feature of our strategy, which we emphasize in this paper, is that it fits well into the SDCloud. SDCloud architecture consists of several layers among which control layer is the brain in this complex system [48]. General control layer in SDCloud consists of three parts, which are Monitor, Deployer and Planner (Fig.6). The mechanism of P-Aware works in accordance with the general control layer in SDCloud. The Monitor collects the status of physical resources (CPU, Memory, Disk, Network, etc.) in runtime. The Analyzer in Fig. 5.A plays the analytical role like the Planner in Fig. 6. These object-oriented Modules in control layer in Fig. 6 are responsible for the specific optimal target in scheduling, for instance, Energy Module manages to make the decisions of Planner to be cost-effective [48]. As the claim in Sect. 1, only energy conservation and load balancing are considered in P-Aware strategy in this paper. Deployers perform the implementation of resource mapping between application jobs and physical nodes which is done in Scheduler in P-Aware. Fig. 5 clearly outlines the structure of the P-Aware and marks its location in SDCloud architecture.

4.2 PMDBP scheduling algorithm

PMDBP which is applied by the Scheduler module has 3 steps:

STEP 1: Fixing Optimal Proportion. Our algorithm is based on a principle that the more precious a resource is, the higher utilization should be achieved. As a result, we will fix OP (optimal proportion) by the proportional relationship between D_{Q1}, D_{Q2}, ... D_{Qm}. We normalize OP by setting the maximal preciousness value OP_i to 1, then we get an m-dimensional vector OP $<OP_1, OP_2, ...OP_m>$. After that, we need to fix AOP (the actual optimal proportion in current PM p) where $\forall p \in P, \forall i \in R, AOP_i = C_{pi} \times OP_i$

Algorithm 1 Fixing Algorithm

```
This program is invoked when a new job j arrives
at Q which is available in Analyzer.
1. Procedure ($D_{ji}$ ,&$D_{Qi}$)
2.  While j<=n
3:    For i = 1 To m
4:      $D_{Qi}$+=$D_{ji}$ ;
5:    End For
6:  End While
7:  While (j=n)
8:    Calculate and Normalize $OP_i$ ;
9:    Calculate $AOP_i$
10:  End While
12: End Procedure
```

STEP 2: Job Classification and Sorting. Classification is necessary improve accuracy in scheduling in CDCs [49]. We use a simple way to classify jobs in Q. Firstly, we locate the corresponding dominant resource for each job in Q. The method is that when the vector D_{ji} of a new job j is available in Analyzer, we calculate the ratio between D_{j1}, D_{j2} ... D_{jm}. After comparing the ratio with AOP, the dominant resource of job j will be located. We put job j in category i if its dominant resource is i. All jobs in Q are clustered by their corresponding dominant resources. In this way, for example, all I/O intensive jobs (compared with AOP_i) are put into i-th category which is separated from memory-intensive category, network-intensive category, etc. Note that job sorting is carried out simultaneously in each category. Considering category i, here jobs are sorted with the descending order of D_{ji} (j ϵ category i) value. When a new job j is assigned to category i, it will be inserted into the sorting sequence i according to D_{ji}.

Algorithm 2 Classification and Sorting

```
This program is invoked on availability of AOP.
1: Procedure (AOP)
2:  Initialize m categories;
3:  For j =1 To n
4:    Compare ratio between $D_{ji}$ with AOP;
5:    Fix the dominant resource of job j;
6:    Add j to the corresponding category i;
7:    Insert j into the sorting sequence in
      category i according to $D_{ji}$;
8:  End For
9: End Procedure
```

STEP 3: Job Mapping. Job mapping is the last but most crucial step to assign proper jobs to corresponding PM. Considering PM p as the current bin, firstly we should find out the resource i whose low utilization in p has the maximum deviation from AOP_i. In PMDBP, multiple resources are consumed in a proportional way. As a result, an i-intensive job will be chosen to increase the utilization of resource i in p. The job selection criteria from category i is that larger demand value of resource i has greater priority. Here are some guidelines in step 3: a) the default resource for bin packing is the resource with the maximal OP value. This guideline acts when the bin is empty or each resource is just on its AOP. b) the inner loop for iteration is job selection. After mapping the prior job j into current PM p, we remove j both from J and j category. Meanwhile, Monitor records job j's arrival and adjust AOP in p for next loop. Here the AOP adjustment is to detect which resource has the maximum deviation from AOP. Thus the utilization of these resources rises alternately and this way guarantees the superior performance in load balancing. c) the outer loop is alternation of current PM. The current PM will not alternate until it can't find any proper candidate job within the constraint of multiple resource capacity in formula (6).

Algorithm 3 Mapping Algorithm

```
This program is invoked as the classification and
sorting work is completed in step 2.
1:Procedure ( )
2: While Q!=Null
3:   Initialize a new PM;
4:   p = p + 1;
5:   Find out the lowest resource i which has
     the maximum deviation from AOP in p.
6:   While category i != Null
7:     Choose the current job j in sequence i;
8:     IF j meets the multiple resource
       constraints after packing this job into p
9:       Add this job to subset U in p;
10:      Adjust $D_{pi}$ ($i \epsilon U$);
11:      Remove job j from category i;
12:    Else
13:      IF the current job is the tail of
         sequence i
14:        Goto outer while loop ;
15:      Else Set the current job to be the
         next job in sequence i;
16:      End IF
17:    End IF
18:  End While
19: End While
20:End Procedure
```

4.3 Complexity analysis

Intuitively, the complexity of PMDBP algorithm is related to these 3 steps which are discussed in Sect. 4.2. As step 1 is working with the process of jobs' arrival, therefore it is the prepossessing work in BMDBP and will not increase the time complexity in the actual application scenarios. In step 2, there is a double loop, the sorting method here is insertion thus its time complexity is O (n^2). In addition, it takes m operations (m is the number of resource categories) to fix the dominant resource in each iteration, therefore the whole time complexity in Step 2 is O (mn^2). In Step 3, the outer loop still has n times iterations while the inner loop needs mL' times calculations (L' is the length of item category). As the average value of L' is (n/m), thus the time complexity of Step 3 is O (n^2). The final time complexity is the accumulative result of Step 2 and Step 3 which equals O (mn^2).

5 Performance evaluation

In this section, we carry out a number of experiments and the results are analyzed to evaluate our scheduling algorithms.

5.1 Algorithms comparison, metrics and goals

Algorithms like FCFS Max-Min, DRF and Slot-Shared take the assumption that resources are limited and jobs cannot get resource as they need in scheduling. As this assumption does not fit the scenarios we discuss in this paper, we compare PMDBP algorithm with some state of the art bin packing algorithms like FFDProd, FFDAvgSum [47], and Norm-based-Greedy [50]. To the best of our knowledge, these algorithms represent highest level and outperform others in multi-dimensional cases. Both FFDProd and FFDAvgSum are based on the classical FFD (First Fit Decreasing) algorithm which is guaranteed to find an allocation with at most 11/9 OPT + 1 bins in the one dimensional case [51]. The way of sorting reflects the difference between these two algorithms, the former sorts out items by their dominant resource and ignores resources in other dimensions, while the latter introduces the weight coefficient for multiple resources. Norm-based-Greedy (L2) is applied by Microsoft's Azure [26]. This algorithm is based on the difference between the demand vectors of unassigned items and the residual capacity r. Then it chooses the Item I^l that minimizes the quantity $\sum_i a^i (I^l - r_i)^2$.

We coded up these algorithms in C# and ran them on random instances for simulation. To make the experimental parameters get closer to real application scenarios, we have carefully analyzed the historical data in the CDC in our cooperative company and get some characteristics. One is random, resource demand from various jobs are disorganized and can hardly be predicted. Obviously jobs with oversize resource demand are rare in queue as they usually get parallelized split. Therefore with some biased data that have been cleaned, resource demand value from jobs can be considered to follow uniform distribution in a certain interval. Besides randomness, irrelevance is another feature. There is no direct or strong correlation among demand values in multi-dimensional resources. Thus the data parameters from different dimensions should be independently sampled which is shown in table 3. We have done lots of experiments, and the experimental results are consistent. In other words, the simulation parameters can be adjusted freely as long as they follow the above-mentioned two criteria. Simulation data in table 3 is just a common instance. To be more straightfor-

Table 3 primary input cases

Input case	r1	r2	r3
1	[1, 40]	[10, 50]	[25, 75]
2	[10, 40]	[30, 70]	[5,75]
3	[25, 55]	[10, 80]	[6,44]
4	[10, 80]	[30, 50]	[5,40]
5	[10, 50]	[30, 60]	[20,80]

ward, here we normalized the resource demand value and set the relative value of bin capacity in each dimension to 100.

In order to make better evaluation and comparison, two important indexes are introduced. The number of bins N is adopted to measure the efficiency of bin packing. CV_i (coefficient of variation for demand value of resource i) is used to measure resource consumption balance among different bins. If an algorithm uses fewer bins to allocate jobs in queue, it is considered to be a better algorithm. Meanwhile, an algorithm with smaller CV value outperforms other algorithms in load balancing. Using these two indexes, we compare our PMDBP algorithms with other three algorithms.

5.2 Performance comparison

Firstly we set number of Items to 200 and compare these four algorithms with 3D version. The demand value in each resource dimension is random and independent with U $[\alpha, \beta]$ distribution which is shown in each cell in Table 3 respectively. These parameters are not specific and other values can be adopted as long as they follow the two criteria which we discuss in Sect. 5.1.

With the above-mentioned demand value of input cases, the comparison of bin packing efficiency is shown in Fig. 7. Fig. 7 shows that in 3-dimensional version with 200 Items, L2 and PMDBP use fewer bins than other two algorithms while FFDProd has the maximum value in most input cases. The bin packing efficiency of PMDBP outperforms 2∼9 percent over FFDProd in this configuration. All algorithms have 100 iterations, and we use average number in order to eliminate the random error.

The CV value in Fig. 8 is introduced to evaluate load balancing. We first use a two-dimensional array *Box* to record packing data in each box. The number of bins N and the vector *Box* will be achieved in the program, thus the variance of i-th row data of *Box* can represent the degree of load balancing in resource i. As the mean of resource i in these input cases is not the same, therefore we use CV to show intuitively the effect of different algorithms on load balancing. The CV value can be calculated as follows:

$$\mathrm{CV}_i = \sqrt{var(Box(i))}/mean(Box(i)). \tag{7}$$

The following 3 sub-figures in Fig. 8 describe the CV percentage comparison on three resources respectively. Fig. 8 shows that performance of PMDBP is the best because PMDBP algorithm always keeps the cooperative proportion among multi-dimensional resources.

Variance of configuration parameters is needed to evaluate the robustness of PMDBP. Both the number of items and dimensions are taken into consideration in order to evaluate this algorithm again.

Items We set the number of items to 100, 200 and 500 respectively, and the corresponding performances are shown in Fig. 9. The number of bins needed in FFDProd is set as the reference because it is the most commonly used algorithm. Fig. 9 shows that PMDBP and L2 still outperform others within the scope of item numbers. Meanwhile the efficiency of bin packing is directly related to the number of items. PMDBP saves 4∼10 % bins than FFDProd when number of items is 100. This advantage decreases to 2∼8 % with 200 items and 1∼2 % with 500 items. The conclusion is that the packing efficiency of FFDAvgSum is close to FFDProd with small amplitude fluctuation, while the L2 and PMDBP always perform better. The percentage of saving bins over FFDProd is related to the number of items: the more the number of items, the smaller will be the advantage of PMDBP. PMDBP can hardly gain advantage in efficiency over FFDProd when the number of items increases to 500 as it enriches the diversity of combination for multiple resources.

Further experiments are done to test the changes in CV percentage under different number of items. For an intuitive display of relationship between CV value and the number of items, we choose the fifth input case and fixed the observing point at first dimension. Fig. 10 shows that PMDBP is always the optimal algorithm followed by FFDProd in the range of variation of item number. Though the advantage of PMDBP decreases along the positive direction of the x-axis, it is still 9 % ahead over FFDProd and 13 % ahead over L2 with 500 items. We also did further experiments with other input cases and other dimensions, and found the same trends. Therefore, the conclusion is that PMDBP performs better in load balancing in 3-dimensional scenarios.

Dimensions Here we fix the number of items and observe the performance of these algorithms when change the number of dimensions. We set the number of dimensions to 2, 3, 4 and 6 respectively while fix the item number to 200. Firstly we evaluate the bin packing efficiency which is shown in Fig. 11. We still set FFDProd as the reference and compare other algorithms using relative value. Fig. 11 shows that L2 leads other algorithms in efficiency in 2D, 3D and 4D followed by PMDBP. PMDBP achieves more obvious advantage when the number of dimensions increases and becomes the optimal algorithm in 6D. We calculate the average number of bins based on 10 input cases for each dimensional change.

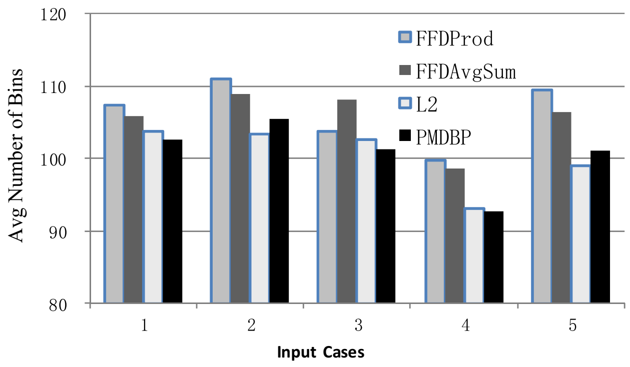

Fig. 7 Average Number of Bins in 3D version with 200 Items

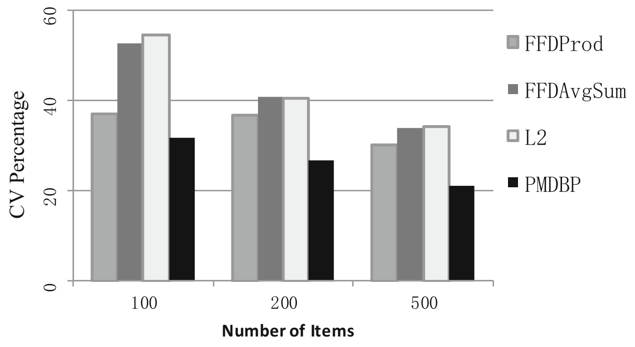

Fig. 10 CV Comparison with different number of items (r is resource I, Input Case = 5, Dimensions = 3)

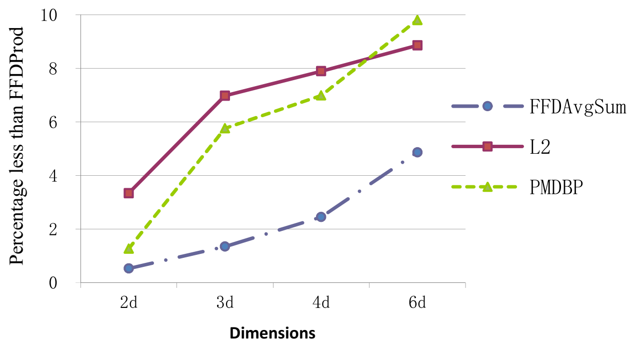

Fig. 11 Comparison of N with FFDProd with different number of dimensions

Now we observe the effect of variance of dimensions on load balancing. Obviously each dimension has its CV value. Therefore the numbers of CV values are different with different dimensions. For example, experiment with 6D produces 6 CV values from CV_r1 to CV_r6 while experiment with 2D only has CV_r1 and CV_r2. In order to make counterpart comparison, we introduce CV–_Sum which equals the sum of CV_r1, CV_r2… CV_rn with m dimensions. Fig. 12 shows both the CV percentage of each dimension and the CV_Sum value. PMDBP outperforms other algorithms in each dimension and the CV_Sum value of PMDBP is 30 % lower than that of L2. CV_Sum can present the overall load balancing well and thus we set it to be the index for comparison with different dimensions. Fig. 13 shows that PMDBP always gets the minimum CV_Sum in different dimension scenarios, thus it is the most 'balanced' algorithm. It is about 20 % lower than others with 2 dimensions and the gap grows when the dimension increases. As a result, we have come to the conclusion that PMDBP performs better in load balancing especially with high dimensions.

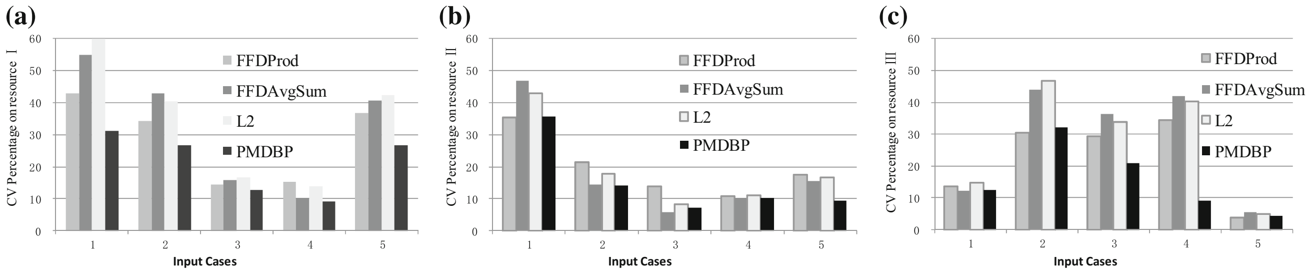

Fig. 8 CV percentage comparison for 3 dimensional resources with 200 Items. **a** resource I, **b** resource II, **c** resource III

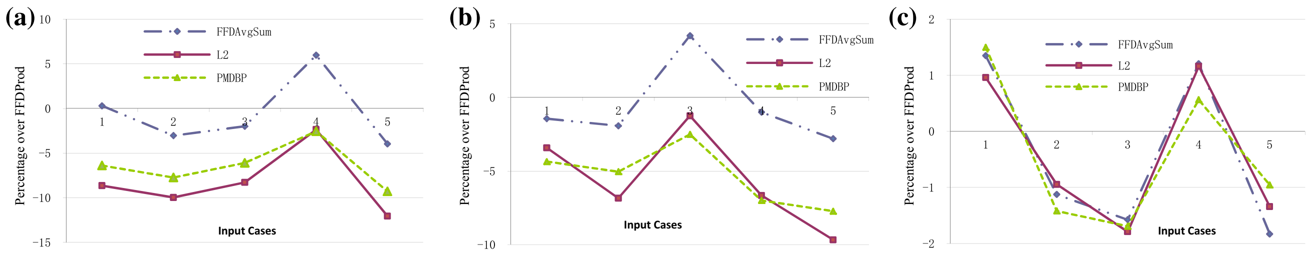

Fig. 9 Comparison of bin packing efficiency with 3 Dimensions with different number of Items. **a** 100 Items, **b** 200 Items, **c** 500 Items

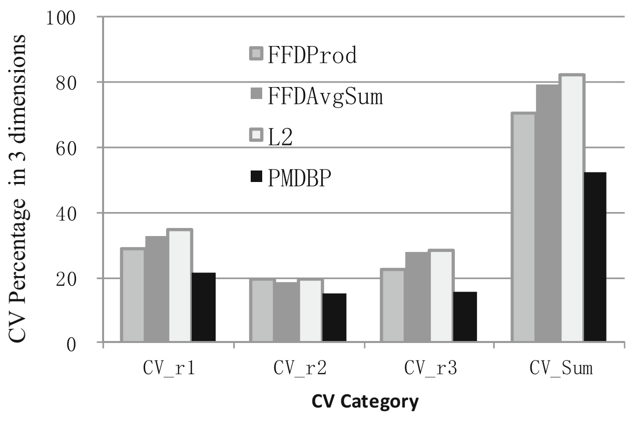

Fig. 12 Comparison of CV values with 3 Dimensions

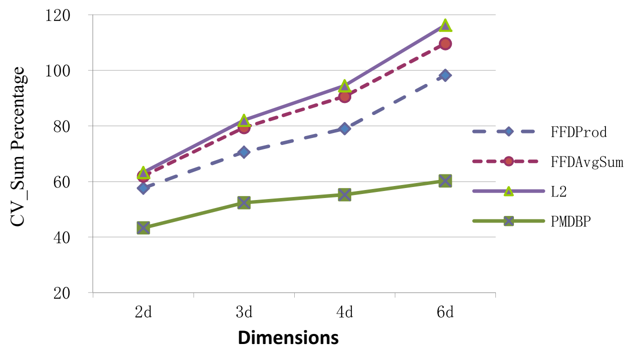

Fig. 13 Comparison of Load Balancing in different Dimension scenarios

5.3 Consolidation overhead and scalability

Finally, we evaluated the consolidation overhead of PMDBP in terms of execution time and memory used. The experimental PM has Intel i5-4590 CPU 3.30GHz with 4GB memory. We tested PMDBP with three consolidation instances and the overhead execution time is only 1.3, 2.9, 0.8 seconds respectively. We also recorded the additional memory consumption by our algorithm and neither of memory usage is higher than 6M. The result is that PMDBP can cope with the actual resource scheduling scenarios with a small memory cost and sustainable execution time.

Furthermore, we examine the scalability of our algorithm and obtain complementary insights. We also change the number of dimensions (m) and number of items (n) which is shown in Figs. 14 and 15. We firstly set N to 500 and vary the value of m. Fig. 14 shows that the runtime of PMDBP is shorter than that of FFDProd and the advantage becomes more obvious when the number of dimensions increases. PMDBP is quite suitable for high-dimensional scheduling scenarios, especially in scenario with more than 3 types of resources. Secondly, we fix m to 3 and varied the value of n. Fig. 15 shows that neither of these algorithms has a linear relationship between execution time and n. Among

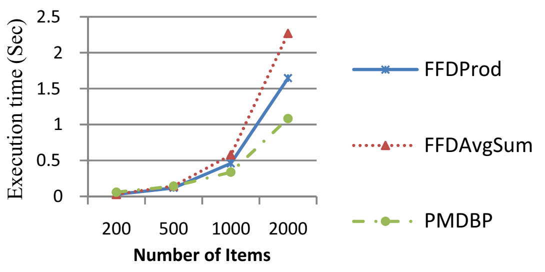

Fig. 14 Overhead Comparison under variance of m (n=500)

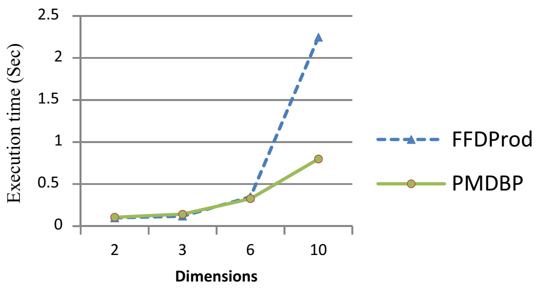

Fig. 15 Overhead Comparison under variance of n (m=3)

these algorithms, L2 has the longest execution time which is excluded from this figure. PMDBP uses the least time when N rises to 1000 or larger and it takes only about 1 second to consolidate 2000 items. We also tested PMDBP with DRF in a cooperative way and it achieved better performance than DRF+FF, DRF+NF, DRF+FFD. The above-mentioned experiments demonstrate that our strategy P-Aware is robust, lightweight and flexible enough to be a good choice in multiple resource scheduling scenarios in CDC in SDCloud norm.

5.4 Algorithm insight

Here we discuss the reason why PMDBP outperforms others from perspective of algorithm itself. In one-dimensional bin packing scenario, the classical FFD algorithm performs well. However, the complexity rises dramatically when the number of dimensions increases. In these cases, FFD encounters obstacles. One of these obstacles is how to sort objects in the multi-dimensional scenario. FFDProd sorts objects by their dominant capacity. Thus only one dimension is taken into consideration. Spaces in other dimensions are wasted in this way. By setting coefficients for different dimensions, FFDAvgSum makes improvement in the sorting way. However, there is no mature way to set these coefficients. Moreover, these coefficients are fixed in the process of bin packing, thus they can hardly get the adaptive ability to be applied in the complex CDC environment. L2 represents another way in which there is no sorting process. The objective function of L2 tries to find the optimal object which can occupy maximum spaces (sum of all dimensions). This approach indeed takes the space consumption of each dimension into consideration. However, excessive pursuit of overall space consumption may cause over-steering in some dimensions. Therefore L2 is the most unbalanced algorithm in the above-mentioned experiments. In addition, though L2 achieves the closest performance to PMDBP in bin-packing efficiency, it has to traverse the whole set when it tries to find the next optimal object. The runtime of L2 becomes the highest in these 4 algorithms when number of n increases. Different from other algorithms, PMDBP tries to keep the optimal consumption rate among different dimensions in bin-packing process. This rate will be the adaptive constraint to find the next proper object. In addition, this way has no conflict with sorting. On the contrary, sorting within categories is adopted to guarantee the alternative way in multi-dimensional spaces consumption. The advantages in bin-packing efficiency and balance are derived from the specific way in which PMDBP works.

6 Conclusions and future work

As there are no two leaves exactly alike in the world, hardware configuration and resource demand vary from each other in different CDCs. Meanwhile, no man ever steps in the same river twice. The demand and allocation of multiple resources fluctuate over time in a CDC in SDCloud. Both these two important features are taken into consideration in our model thus it is flexible enough to be applied in various application contexts in SDCloud. We formulate the problem and propose a corresponding algorithm PMDBP in this paper. In this algorithm, multiple resources are consumed in a cooperative way. Based on this model and algorithm, our strategy P-Aware has good performance both in energy conservation and load balancing, as compared to other state of the art strategies. In addition, the runtime overhead and memory consumption is very well contained, even as it scales up to thousands of jobs or 10 dimensions scenario. Extensive experiments and complementary simulations show that P-Aware is an efficient, universal, robust and scalable resource scheduling strategy in multiple resource scenarios and may contribute to the multiple resource scheduling area in CDC, especially in SDCloud.

Recently SDCloud has emerged as the most promising research [19]. Conventional scheduling strategies face enormous changes to adapt the new norm. This shift will certainly bring about excessive problems for the integration of massive resources and various performance indexes in a single system. Our team is committed to several key problems in SDCloud, such as cost effective, dynamics, deadline, SLA, load balancing, resource contention, and cyber security [52–54]. Our future work includes two parts. The first one is the improvement on some related details, for instance, how to determine the optimal monitoring frequency in infrastructure layer, and how to minimize the system error (e.g. oversteering) in the control layer. The other one is the continuous research on other key factors in SDCloud, especially on dynamics (both deterministic and stochastic), deadline (and dependencies between tasks), and resource contention in QoS. We also have interest to collaborate with other teams in cloud security such as digital forensic [55–58].

Acknowledgments This work was supported in part by NSFC under Grant 61103143, in part by Innovation Action Plan supported by Science and Technology Commission of Shanghai Municipality (15DZ1100305). The authors thank to China Telecom Shanghai branch which is the cooperative company in our current fund. They also thank to Dr. Fei.Xu for his helpful advices to this paper.

References

1. Hahne, E.L.: Round-robin scheduling for max-min fairness in data networks. IEEE J. Sel. Areas Commun. **9**, 1024–1039 (1991)
2. Ghodsi, A., Zaharia, M., Hindman, B., Konwinski, A., Shenker, S., Stoica, I.: Dominant resource fairness: fair allocation of multiple resource types. In: NSDI 2011, pp. 24–24
3. Joe-Wong, C., Sen, S., Lan, T., Chiang, M.: Multiresource allocation: fairness-efficiency tradeoffs in a unifying framework. IEEE/ACM Trans. Netw. **21**, 1785–1798 (2013)
4. Parkes, D.C., Procaccia, A.D., Shah, N.: Beyond dominant resource fairness: Extensions, limitations, and indivisibilities. ACM Trans. Econ. Comput. **3**(1), 3 (2015)
5. Wang, W., Li, B., Liang, B.: Dominant resource fairness in cloud computing systems with heterogeneous servers. In: 2014 Proceedings of the IEEE 2014 INFOCOM, pp. 583–591. IEEE, Piscataway
6. “Apache Hadoop”. http://hadoop.apache.org
7. Hindman, B., Konwinski, A., Zaharia, M., Ghodsi, A., Joseph, A.D., Katz, R.H., Shenker, S., Stoica, I.: Mesos: A Platform for Fine-Grained Resource Sharing in the Data Center. In: NSDI 2011, pp. 22–22
8. “Apache Mesos”. http://mesos.apache.org
9. Di, S., Wang, C.-L., Zhang, W., Cheng, L.: Probabilistic best-fit multi-dimensional range query in self-organizing cloud. In: 2011 International Conference on Parallel Processing (ICPP), pp. 763–772. IEEE, Piscataway (2011)
10. Sarood, O., Gupta, A., Kalé, L.V.: Cloud Friendly Load Balancing for HPC Applications: Preliminary Work. In: 41st International Conference on Parallel Processing Workshops (ICPPW), pp. 200–205. IEEE, Piscataway (2012)
11. Minarolli, D., Freisleben, B.: Distributed resource allocation to virtual machines via artificial neural networks. In: 22nd Euromicro International Conference on Parallel, Distributed and Network-Based Processing (PDP), pp. 490–499. IEEE, Piscataway (2014)
12. Chiesi, M., Vanzolini, L., Mucci, C., Scarselli, E.F., Guerrieri, R.: Power-aware job scheduling on heterogeneous multicore architectures. IEEE Trans. Parallel Distrib. Syst. **26**(3), 868–877 (2015)
13. Wu, C., Li, J., Xu, D., Yew, P.-C., Li, J., Wang, Z.: FPS: A Fair-Progress Process Scheduling Policy on Shared-Memory Multiprocessors. IEEE Trans. Parallel Distrib. Syst. **26**(2), 444–454 (2015)
14. Ryoo, I., Na, W., Kim, S.: Information exchange architecture based on software defined networking for cooperative intelligent transportation systems. Clust. Comput. **18**(2), 771–782 (2015)

15. Zeng, D., Li, P., Guo, S., Miyazaki, T., Hu, J., Xiang, Y.: Energy minimization in multi-task software-defined sensor networks. IEEE Trans. Comput. **64**(11), 3128–3139 (2015)
16. Juliadotter, N.V., Choo, K.-K.R.: Cloud attack and risk assessment taxonomy. IEEE Cloud Comput. **2**(1), 14–20 (2015)
17. Ab Rahman, N.H., Choo, K.-K.R.: A survey of information security incident handling in the cloud. Comput. Secur. **49**, 45–69 (2015)
18. K. S. Tep, B. Martini, R. Hunt, and K.-K. R. Choo,: A taxonomy of Cloud Attack Consequences and Mitigation Strategies: The Role of Access Control and Privileged Access Management. In: 2015 IEEE Trustcom/BigDataSE/ISPA, vol. 1, pp. 1073–1080 (2015)
19. Jararweh, Y., Al-Ayyoub, M., Benkhelifa, E., Vouk, M., Rindos, A.: Software defined cloud: Survey, system and evaluation. Future Gener. Comput. Syst. **58**, 56–74 (2016). doi:10.1016/j.future.2015.10.015
20. Sandhu, R., Sood, S.K.: Scheduling of big data applications on distributed cloud based on QoS parameters. Clust. Comput. **18**(2), 817–828 (2015)
21. Calheiros, R.N., Buyya, R.: Cost-effective provisioning and scheduling of deadline-constrained applications in hybrid clouds. Web Information Systems Engineering-WISE 2012, pp. 171–184. Springer, Heidelberg (2012)
22. Coffman, J., Edward, G., Garey, M.R., Johnson, D.S.: Dynamic bin packing. SIAM J. Comput. **12**(2), 227–258 (1983)
23. Beaumont, O., Eyraud-Dubois, L., Thraves Caro, C., Rejeb, H.: Heterogeneous resource allocation under degree constraints. IEEE Trans. Parallel Distrib. Syst. **24**(5), 926–937 (2013)
24. Garey, M.R., Graham, R.L., Johnson, D.S., Yao, A.C.-C.: Resource constrained scheduling as generalized bin packing. J. Comb. Theory Ser. A **21**(3), 257–298 (1976)
25. Leinberger, W., Karypis, G., Kumar, V.: Multi-capacity bin packing algorithms with applications to job scheduling under multiple constraints. In: International Conference on Parallel Processing, pp. 404–412. IEEE, Piscataway (1999)
26. Panigrahy, R., Talwar, K., Uyeda, L., Wieder, U.: Heuristics for vector bin packing. http://www.research.microsoft.com (2011)
27. Sun, X., Wu, Q., Tan, Y., Wu, F.: MVEI: An Interference Prediction Model for CPU-intensive Application in Cloud Environment. In: 13th International Symposium on Distributed Computing and Applications to Business, Engineering and Science (DCABES), pp. 83–87. IEEE, Piscataway (2014)
28. Tsai, C.-H., Shin, K.G., Reumann, J., Singhal, S.: Online web cluster capacity estimation and its application to energy conservation. IEEE Trans. Parallel Distrib. Syst. **18**(7), 932–945 (2007)
29. Pham, C., Li, Q., Estrada, Z., Kalbarczyk, Z., Iyer, R.: A Simulation Framework to Evaluate Virtual CPU Scheduling Algorithms. In: 33rd International Conference on Distributed Computing Systems Workshops (ICDCSW), pp. 138–143. IEEE, Piscataway (2013)
30. Kinger, S., Goyal, K.: Energy-efficient CPU utilization based virtual machine scheduling in Green clouds. In: Communication and Computing (ARTCom 2013), Fifth International Conference on Advances in Recent Technologies, pp. 28–34. IET (2013)
31. Zhang, P., Chu, R., Wang, H.: MemHole: An Efficient Black-Box Approach to Consolidate Memory in Virtualization Platform. In: 41st International Conference on Parallel Processing Workshops (ICPPW), pp. 608–609. IEEE, Piscataway (2012)
32. Takouna, I., Meinel, C.: Energy and Performance Efficient Consolidation of Independent VMs in Virtualized Data Center by Exploiting VMs' Memory Demand Heterogeneity. In: IEEE/ACM 6th International Conference on Utility and Cloud Computing (UCC), pp. 315–320. IEEE, Piscataway (2013)
33. Liu, L., Chu, R., Zhu, Y., Zhang, P., Wang, L.: DMSS: a dynamic memory scheduling system in server consolidation environments. In: 15th IEEE International Symposium on Object/Component/Service-Oriented Real-Time Distributed Computing Workshops (ISORCW), pp. 70–75. IEEE, Piscataway (2012)
34. Moraveji, R., Taheri, J., Reza, M., Rizvandi, N.B., Zomaya, A.Y.: Data-intensive workload consolidation for the Hadoop distributed file system. In: Proceedings of the 2012 ACM/IEEE 13th International Conference on Grid Computing, pp. 95–103. IEEE Computer Society (2012)
35. Dong, B., Li, X., Xiao, L., Ruan, L.: Towards minimizing disk I/O contention: a partitioned file assignment approach. Future Gener. Comput. Syst. **37**, 178–190 (2014)
36. Zhou, T., Li, H., Zhu, B., Zhang, Y., Hou, H., Chen, J.: STORE: Data recovery with approximate minimum network bandwidth and disk I/O in distributed storage systems. In: IEEE International Conference on Big Data (Big Data), pp. 33–38. IEEE, Piscataway (2014)
37. Mijumbi, R., Gorricho, J.-L., Serrat, J., Shen, M., Xu, K., Yang, K.: A neuro-fuzzy approach to self-management of virtual network resources. Expert Syst. Appl. **42**(3), 1376–1390 (2015)
38. Sun, X., Su, S., Xu, P., Chi, S., Luo, Y.: Multi-dimensional resource integrated scheduling in a shared data center. In: 31st International Conference on Distributed Computing Systems Workshops (ICDCSW), pp. 7–13. IEEE, Piscataway (2011)
39. Jin, H., Pan, D., Xu, J., Pissinou, N.: Efficient VM placement with multiple deterministic and stochastic resources in data centers. In: IEEE Global Communications Conference (GLOBECOM), pp. 2505–2510. IEEE, Piscataway (2012)
40. Zhu, Q., Zhu, J., Agrawal, G.: Power-aware consolidation of scientific workflows in virtualized environments. In: Proceedings of the 2010 ACM/IEEE International Conference for High Performance Computing, Networking, Storage and Analysis, pp. 1–12. IEEE Computer Society (2010)
41. Xu, F., Liu, F., Jin, H., Vasilakos, A.V.: Managing performance overhead of virtual machines in cloud computing: a survey, state of the art, and future directions. Proc. IEEE **102**(1), 11–31 (2014)
42. Massimo, F., Christian, E., Henry, C., Choo, K.-K.R.: Live migration in emerging cloud paradigms. IEEE Cloud Comput. **3**(2), 12–19 (2016)
43. Xu, F., Liu, F., Liu, L., Jin, H., Li, B., Li, B.: iaware: Making live migration of virtual machines interference-aware in the cloud. IEEE Trans. Comput. **63**(12), 3012–3025 (2014)
44. Loumiotis, I., Adamopoulou, E., Demestichas, K., Stamatiadi, T., Theologou, M.: On the predictability of next generation mobile network traffic using artificial neural networks. Int. J. Commun. Syst. **28**(8), 1484–1492 (2015)
45. Chen, Z., Zhu, Y., Di, Y., Feng, S.: Self-Adaptive prediction of cloud resource demands using ensemble model and subtractive-fuzzy clustering based fuzzy neural network. Comput. Intell. Neurosci. **2015**, 17 (2015)
46. Chen, Z., Zhu, Y., Di, Y., Feng, S., Geng, J.: A high-accuracy self-adaptive resource demands predicting method in IaaS cloud environment. Neural Netw. World **25**(5), 519 (2015)
47. Mark, S., David, S., Frdric, V., Henri, C.: Resource allocation algorithms for virtualized service hosting platforms. Parallel Distrib. Comput. **70**(9), 962–974 (2010)
48. Buyya, R., Calheiros, R.N., Son, J., Dastjerdi, A.V., Yoon, Y.: Software-defined cloud computing: Architectural elements and open challenges. In: 2014 International Conference on Advances in Computing, Communications and Informatics (ICACCI), pp. 1–12. IEEE, Piscataway (2014)
49. Xu, D., Yang, S., Liu, R.: A mixture of HMM, GA, and Elman network for load prediction in cloud-oriented data centers. J. Zhejiang Univ. SCI C **14**(11), 845–858 (2013)
50. Microsoft Systems Center Virtual Machine Manager. http://www.microsoft.com/systemcenter/virtualmachinemanager
51. Garey, M.R., Johnson, D.S.: Approximation algorithms for bin packing problems: A survey. Anal. Design Algorithms Comb. Optim. **266**, 147–172 (1981)

52. Esposito, C., Castiglione, A., Choo, K.-K.R.: Encryption-based solution for data sovereignty in federated clouds. IEEE Cloud Comput. **3**(1), 12–17 (2016)
53. Osanaiye, O., Choo, K.-K.R., Dlodlo, M.: Distributed denial of service (ddos) resilience in cloud: review and conceptual cloud ddos mitigation framework. J. Netw. Comput. Appl. **67**, 147–165 (2016)
54. Choo, K.-K.R.: A cloud security risk-management strategy. IEEE Cloud Comput. **1**(2), 52–56 (2014)
55. Martini, B., Choo, K.-K.R.: Distributed filesystem forensics: XtreemFS as a case study. Digital Investiga. **11**(4), 295–313 (2014)
56. Quick, D., Choo, K.-K.R.: Google Drive: Forensic analysis of data remnants. J. Netw. Comput. Appl. **40**, 179–193 (2014)
57. Martini, B., Choo, K.-K. R.: Remote programmatic vcloud forensics: a six-step collection process and a proof of concept. In: 2014 IEEE 13th International Conference on Trust, Security and Privacy in Computing and Communications (TrustCom), pp. 935–942 (2014)
58. Martini, B., Choo, K.-K.R.: Remote programmatic vcloud forensics: a six-step collection process and a proof of concept. In: IEEE 13th International Conference on Trust, Security and Privacy in Computing and Communications (TrustCom), pp. 935–942. IEEE, Piscataway (2014)

Hang Zhou is a Ph.D. student in the School of Computer Engineering and Science, Shanghai University, Shanghai China. His current research interests focus on energy saving, resource contention and multiple resource scheduling in cloud computing.

Qing Li received Ph.D. degree from College of Computer Science in the Huazhong University of Science and Technology, Wuhan, China. He is a professor and tutor of Ph.D. students in the School of Computer Engineering and Science, Shanghai University. His current research interests include HPC, Parallel Algorithms and Complex Networks. He presided over several NSFC projects.

Weiqin Tong received Ph.D. degree from Department of Computer Science and Engineering in the Shanghai Jiao Tong University, Shanghai, China, in 1995. He is a visiting scholar in the Chinese University of Hong Kong from 1991 to 1993 He is a professor and tutor of Ph.D. students in the School of Computer Engineering and Science, Shanghai University. His current research interests include large-scale content distribution in the internet, Software Defined Network in datacenter, data-intensive computing. He presided over several NSFC projects and served as chairman of several academic conferences.

Samina Kausar has been working as lecturer at University of Management Sciences & Information Technology Kotli, Azad Kashmir since 2007. She has completed her MS in Computer Science from INTERNATIONAL ISLAMIC UNIVERSITY Islamabad in 2007. Before MSCS she got the degree of MIT from Mirpur University of Azad Jammu & Kashmir, Mirpur in 2004. Now, she is a Ph.D. scholar in computer science at Shanghai University in China. Her research interests include distributed database systems, data mining, cloud computing.

Hai Zhu received Ph.D. degree in the School of Computer Science and Technology, XiDian University, Xi'An, China. He is in postdoctoral research in Xi'An JiaoTong University. His current research interest is resource scheduling in cloud computing.