Optimizing Computing Job Scheduling and Path Planning with Multi-objectives

Haoran Song[1], Chengxiao Yu[2(✉)], Kang Liu[2], Deyun Gao[1], Xuening Shang[1], and Hanxiao Yan[1]

[1] School of Electronic and Information Engineering, Beijing Jiaotong University, Beijing 100044, China
{haoransong,19111028,gaody,xueningshang,hanxiaoyan}@bjtu.edu.cn
[2] Department of New Networks in Peng Cheng Laboratory, Shenzhen, China
yuchx@pcl.ac.cn

Abstract. Machine learning model training relies on parameter server architecture and data parallel mechanism. It is important to achieve deadline-guarantee, energy-saving, and efficient network bandwidth usage objectives simultaneously. In this paper, an integer programming model is formulated to optimize the problem in the scenario of machine learning training. We then propose a heuristic Computing Job Scheduling and Routing Planning (CSRP) method to minimize the violation rate of user deadlines, the used server number, and the network cost. CSRP schedules the computing jobs and selects paths based on computing job characteristics and network status. Due to the features of the same computing parameters requirements for parallel computing, the bandwidth consumption can be further reduced by path aggregation. Therefore, we further propose Aggregated CSRP to select the aggregation node and aggregated paths. We evaluate the performance of our proposed algorithms on trace-driven experiments with results showing that CSRP and Aggregated CSRP outperform other methods in terms of deadline guarantee, energy saving, and efficient network bandwidth usage.

Keywords: Job scheduling · Path planning · Parameter server · Parallel computing

1 Introduction

Machine learning has become an indispensable tool in a wide range of applications. The growing number of parameters [1] in machine learning models has led to significant improvements in performance but also poses a new challenge in the increased demand for computing. Due to the computation-intensive characteristic of model training, a single machine fails to complete the computing jobs timely. Thus, distributed computing based on data parallelism becomes the mainstream solution. Nowadays, using the parameter server architecture to complete distributed computing jobs has become a new trend. The parameter server

V. C. M. Leung et al. (Eds.): Qshine 2023, LNICST 574, pp. 147–157, 2024.
https://doi.org/10.1007/978-3-031-65123-6_11

is used to manage, store, and synchronize parameters among different computing nodes, enabling efficient communication and flexible consistency [2]. In the parameter server architecture, the parameter server distributes the same model parameters to the distributed computing nodes, and the computing nodes send the calculated gradient data to the parameter server to update the computing parameters until the job is completed. And the nodes compute different data blocks in parallel. During this process, there is a large number of communication requirements [3] between the parameter server and computing nodes, which consume high bandwidth tremendously. Therefore, planning appropriate paths for computing jobs is a key issue. To meet users' demands and improve resource utilization, it is necessary to schedule jobs and plan paths. Thus, this paper aims to optimize computing job scheduling and path planning problems.

Computing job scheduling requires comprehensive consideration of user demands and economic benefits. It's crucial to guarantee the deadline for user satisfaction. Besides, minimizing energy consumption is essential for reducing the operating costs of computing clusters. Path planning issue is rarely considered when scheduling computing jobs. In job scheduling, the shortest path algorithm is commonly used for routing, which results in inefficient network bandwidth usage and increases the unnecessary cost of operators. Therefore, in busy clusters with many computing jobs, simultaneously achieving deadline-guarantee, energy saving, and efficient network bandwidth usage objectives are worthy of research.

This paper proposes an approach to optimize computing job scheduling and path planning for training jobs under the parameter server architecture. The approach simultaneously takes consideration of satisfying deadline guarantee, energy saving, and reducing network bandwidth usage. Path planning includes three aspects: computing node selection, path selection between the parameter server and computing nodes, and path aggregation planning. Firstly, we formulate an integer programming model to optimize these problems. Secondly, we propose a heuristic algorithm, named Computing Job Scheduling and Routing Planning (CSRP) based on the improved particle swarm algorithm to solve the problem. CSRP provides innovative computing job scheduling solutions that consider the characteristics of jobs in the current queue, the computing cluster, and the network status. Besides, CSRP also provides path planning for each computing job to satisfy efficient bandwidth usage. Thirdly, we propose Aggregated CSRP, which aggregates the planned paths within the network to further reduce bandwidth consumption.
We summarize our contribution as follows:

- We propose the CSRP approach to optimize computing job scheduling and path planning, which simultaneously achieves deadline-guarantee, energy-saving, and efficient network bandwidth usage objectives.
- We propose the path aggregation algorithm to select the aggregation node and aggregated paths planned by CSRP within the network, further improving network usage.

- We conduct trace-driven simulations, and the results show that CSRP and Aggregated CSRP outperform other methods in terms of deadline guarantees, energy saving, and efficient network bandwidth usage.

2 Related Work

Recently, many job schedulers have been proposed to meet various user needs, Hadoop Yet Another Resource Negotiator (YARN) [5] system utilizes three schedulers: FIFO (First-In-First-Out), where the order of job submission determines their scheduling sequence. The Capacity scheduler [6] is the default scheduler of Hadoop 2.0. It sets up queues for users, and each user queue is allocated a certain amount of computing resources. The fair scheduler [7] guarantees the fair sharing of computing resources among users.

Some schedulers used to meet user deadline guarantees have been extensively studied. DAPS [8] maximizes the satisfaction of job deadlines through preemptive job scheduling. Elastic [9] is an improved scheduling algorithm to ensure the constrained-deadline of elastic tasks by periodically modeling real-time workloads and computing periods for a system of elastic constrained-deadline tasks. Lyapunov-based methodology [10] is proposed to solve scheduling problem with both deadline and throughput constraint, demonstrating sharp performance.

There are also some studies focused on job schedulers that improve resource utilization and reduce costs. Stratus [11] tends to make the allocated resources either mostly full or empty while considering the tradeoff between the deadline guarantee and the monetary cost based on estimated JCT (job completion time). DCOTS [12] tries to complete users' jobs by the deadline and maximize the providers' profit within budget. These works cannot satisfy the deadline guarantee, energy saving, and efficient utilization of network resources simultaneously, and none of the above studies takes into account the path planning problem when scheduling jobs.

3 Problem Formulation

In this section, we model the problem of computing job scheduling and path planning in a computing cluster as an integer programming problem. And select the job scheduling time and transmission paths to meet the deadline guarantee of as many jobs as possible while minimizing energy consumption (determined by the number of active servers) and the network cost (determined by network bandwidth usage).

Deadline-Guarantee Objective. To provide deadline guarantee computing service to task t_j, the sum of its actual start time and its execution time interval should be no larger than its deadline, that is, We use J_v to denote the number of jobs whose deadlines are violated.

$$J_v = |J| - |\{t_j | t_j \in J \wedge e^r_{t_j} + \Delta_{t_j} \leq e^d_{t_j}\}|. \tag{1}$$

where J represents the set of all tasks of all jobs. $e^r_{t_j}$, $e^d_{t_j}$ and Δ_{t_j} are task t_j's actual running start time, deadline, and execution time respectively. The scheduler needs to minimize the deadline cost, which is defined as the deadline violation rate of all jobs, that is R_{dv}.

$$R_{dv} = \frac{J_v}{J}. \tag{2}$$

Energy-Saving Objective. We use S_u to denote the set of all used computing servers through the entire scheduling process, which is defined as energy cost. The scheduler needs to minimize energy cost, that is S_u.

$$S_u = \{s_k | s_k \in S \wedge \sum_{t_j \in J} c_{t_j,s_k} > 0\} \tag{3}$$

where S represents the set of all computing servers and c_{t_j,s_k} represents whether task t_j is assigned to run on server s_k.

Efficient Network Bandwidth Usage Objective. The network load are parameter information and gradient information. There are frequent information interactions in the computation, thus consuming a large amount of network bandwidth and resulting in network costs. The network cost (denoted by N) is calculated by.

$$N = l \cdot h \tag{4}$$

where l represents the path load degree and h represents the hop count of the path. The bottleneck bandwidth is the link with the smallest remaining bandwidth on the path (denoted by BW_b). Let the link capacity be C, and the path load degree l can be calculated as follows.

$$w_{ij} = \begin{cases} BW_b \cdot f^2 & \frac{BW_b}{C} > 90\%. \\ BW_b \cdot f & 50\% < \frac{BW_b}{C} \leq 90\% \\ BW_b & \frac{BW_b}{C} \leq 50\% \end{cases}. \tag{5}$$

where f is the path load factor, which is a constant set by the operator. The definition of network cost is based on network bandwidth resource, therefore a lower network cost indicates efficient network bandwidth usage.

Constrains. A valid job schedule needs to ensure that at any time $\in$ during T, there are no more than $C^c_{s_k}$ tasks running in any server $s_k \in S$, where T represents the entire time during when jobs in J are scheduled.

$$\forall s_k \in S, n_{s_k} \leq C^c_{s_k}. \tag{6}$$

where n_{s_k} denotes the largest number of concurrently running tasks on s_k during T, which is calculated by.

$$n_{s_k} = \max\{|\{t_j | t_j \in J \wedge e^r_{t_j} \leq \Theta \leq e^r_{t_j} + \Delta_{t_j} \wedge c_{t_j,s_k} = 1\}|\}_{\forall \Theta \in T}. \tag{7}$$

A valid job schedule needs to ensure that for any task $\in J$ its scheduling time cannot exceed the start time of the job.

$$e^s_{t_j} \leq e^r_{t_j}. \tag{8}$$

where $e^s_{t_j}$ is task t_j's start time. A valid path planning needs to ensure that the adjacent nodes selected by path planning should be available links in the network. Define the network topology as G. $G = (V, E)$, where V and E represents the set of nodes and the set of edges respectively.

$$(v_i, v_{i+1}) \in E, i = 1, 2, ..., h. \tag{9}$$

Also, the end node of the path should be a computing node.

$$v_{h+1} \in S. \tag{10}$$

Optimization Objective. To address the problem of deadline-guarantee and energy-saving computing job scheduling and path planning, we integrate the objectives defined by Eqs. (2), (3), and (4) into a unified objective function, denoted as $Cost$. We assume that energy saving (Eq. (3)) is β times as important as network load minimization (Eq. (2)). Then, based on [13] that provides the F-score (or F_β) formula to represent such a relationship in combining two metrics, the problem is modeled as an integer program.

$$\min Cost = \frac{(1+\beta^2) \cdot \bar{N} \cdot S_u}{\beta^2 \cdot \bar{N} + S_u} \cdot \frac{J_v}{J} \tag{11}$$

$$subject\ to\ Constraints\ (6),\ (8),\ (9)\ and\ (10)$$

where $\bar{N}$ represents the average network cost for each job. Formula (11) first combines objectives (3) and (4) so that it minimizes the energy cost and the network cost with the consideration of weight β [13]. Then, Formula (9) combines objective (4) to minimize the deadline cost.

4 Proposed Methods

In this section, the algorithms of the CSRP and Aggregated CSRP are discussed. The problem mentioned above can be proven to be NP-hard by using a simple reduction from the generalized assignment problem [14]. In response to this problem, we propose a heuristic CSRP method based on the improved particle swarm algorithm.

4.1 CSRP: Computing Job Scheduling and Route Planning

In the iteration process, we first evaluate the fitness value of each particle and determine the individual best and swarm best positions in the current stage. Then, we calculate the updating probabilities based on the particle's current

position, its individual best position, and the swarm's best position. Finally, particles update their positions and velocities according to these updating probabilities. In summary, this algorithm utilizes the stochastic nature of probability to enhance the particles' searching ability and overall efficiency. The position of each particle represents the scheduling time and the path of the computing job, which consists of $h + 2$ dimensions.

$$p_i = \{p_{i_0}, p_{i_1}, ..., p_{i_{h+1}}\}. \tag{12}$$

where p_i represents the position of the ith particle, p_{i_0} represents the scheduling time of computing jobs, and the other dimensions represent each hop of the path. The study found that a larger w helps the particle to search a larger space and find a new solution domain, while a smaller w is conducive to searching for a better solution in the current solution space. To adjust the search speed of the position space and avoid the algorithm from falling into a local optimal solution, we use the following method to update the inertial parameters.

$$w_k = w_{max} - (w_{max} - w_{min}) \cdot \frac{r}{n}. \tag{13}$$

where r represents the serial number of the algorithm iteration round and n represents the number of algorithm iterations. w_{max} and w_{min} represent the maximum value and the minimum value of the set inertia parameter respectively. The speed update of particles is as follows, c_1 and c_2 represent the learning factors of particles, r_1 and r_2 represent a random number between $[0, 1]$, $pbest_i$ represents the current optimal position of the ith particle, and $gbest$ represents the optimal position of all particles.

$$v_i(k+1) = w_k v_i(k) + c_1 r_1 (pbest_i(k) - p_i(k)) + c_2 r_2 (gbest(k) - p_i(k)) \tag{14}$$

We use the probability mapping method to update the position as shown below. The sigmoid function is used to map the velocity to the $[0, 1]$ interval as the probability. This value is the probability that the particle will take a value of 1 in the next step.

$$s(v_{ik}) = \frac{1}{1 + exp^{(-v_{ik})}} \tag{15}$$

The position is updated based on a probability map as follows.

$$p_i = \begin{cases} 1 & rand() < s(v_{ik}) \\ 0 & otherwise \end{cases} \tag{16}$$

Use CSRP to prioritize high-priority computing tasks and select light-load paths.

4.2 Aggregated CSRP

Aggregation transmission can save network bandwidth usage while meeting communication requirements in computation. It is effective to aggregate paths within

Algorithm 1: Computing job scheduling and path planning

Data: Computing Job information, link state information, computing node set S;
Result: Scheduling time $e_{t_j}^r$ and path for each job;

Initialize: Set the maximum and minimum inertia parameters w_{max} and w_{min}, learning parameters c_1 and c_2, the number of particles X, and the number of iterations n.
for *each particle* i **do**
  Initialize the particle's attributes: p_i, v_i, $pbest_i$.
end
for $k = 1$ **to** n **do**
  Update the inertia parameter w.
  Get the current optimal position of the particle swarm *gbest*.
  for *each particle* i **do**
    Update Particle Velocity v_i.
    Update particle position with constraints p_i.
    Update group optimal position *gbest*.
  end
end

the network to reduce network bandwidth usage. Therefore this paper proposes a path aggregation algorithm to aggregate multiple paths of the same computing job at the current time and selects appropriate aggregation nodes and paths. The aggregation node is responsible for parameters multicast and gradient aggregation within the network. Specifically, Aggregated CSRP first traverses the paths of the computing nodes that complete the same computing job at the current time, which are obtained by CSRP, and selects the appropriate aggregation node and the aggregated paths between the parameter node and the aggregation node using the Dijkstra algorithm.

Suppose there are m computing nodes currently completing the same computing job. The path of each computing node is given by algorithm 1. The weight of the link in the topology is defined as $\frac{BW_u}{C}$, where BW_u and C represent the used bandwidth and the capacity of the network link respectively.

5 Evaluation

In this section, we conduct trace-driven simulation experiments and evaluate the performance of CSRP and Aggregated CSRP, then compare the performance with other methods.

5.1 Experiment Setup

In the simulation, we build up a computing cluster consisting of 1500 computing nodes. For the setup of our simulation, we utilized Google cluster-usage Traces 2011 [15] as a reference. We perform a random selection of 9000-24,000 jobs

Algorithm 2: Path aggregation.

```
Data: Computing job paths and the network topology G;
Result: The aggregation node and the aggregation paths;
for i = 1 to m do
    for j = 1 to h do
        for k = 1 to m do
            Use the Dijkstra algorithm to get paths between the node j and
            other computing nodes.
        end
        Calculate the sum of weight in the paths.
    end
    Select the path with the minimum sum of weight and identify its
    aggregation node.
end
```

with a step size of 3000 and get resource usage and computing information from the traces. The topology of the computing cluster adopts the Jupiter network topology [16] of the Google data center. The computing capacity of each server was set to 8 computing slots by default according to [17]. And we assume that the data set of the computing jobs has been distributed in parallel among the computing nodes and set the constant f to 3. The default value of β in objective (11) is 4, implying that the cluster manager regards energy cost as 4 times more significant than network cost.

We use Aggregated CSRP to denote the use of the path aggregation algorithm combined with CSRP. We compared our methods with the following job schedulers: $FIFO$, *Fair* [6], *Capacity* [7], and *Deadline aware* [8]. These schedulers complete the computation on the nearest available computing node from the parameter server using the shortest path principle.

5.2 Results Analysis

Figure 1 shows the deadline violation rate of each scheduler versus the number of jobs. The deadline violation rate follows $FIFO \approx Capacity \approx Fair > Deadline\ aware > CSRP \approx Aggregated\ CSRP$. $FIFO$, *Capacity* and *Fair* schedule jobs without deadline awareness. *Deadline aware* and CSRP schedule jobs with deadline awareness have better performance than other schedulers. CSRP improves $FIFO$ and *Deadline aware* by 44.2% and 10.3% respectively. The *Deadline aware* algorithm prioritizes the job with the shortest waiting time and preempts the current job, which may lead to deadline violations and unsatisfactory performance due to multiple preemptions by jobs with even shorter waiting times. Aggregated CSRP has a similar deadline violation rate as CSRP since they all regard the deadline violation rate as an important part of the objective function and explore the global excellent solution.

And We measured the energy saving by the number of inactive servers. Since other scheduling algorithms do not consider the energy-saving objective, our

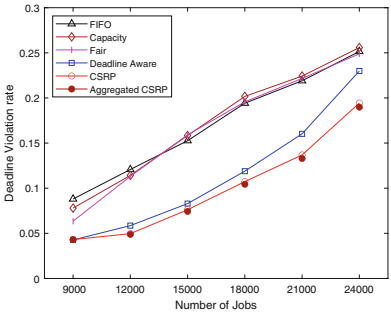

Fig. 1. Deadline violate rate

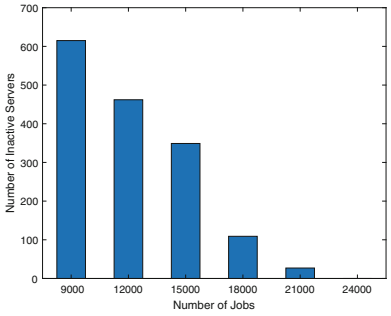

Fig. 2. Energy saving

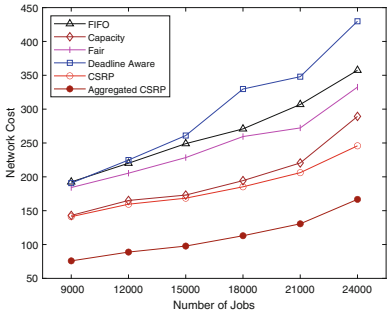

Fig. 3. Network Cost

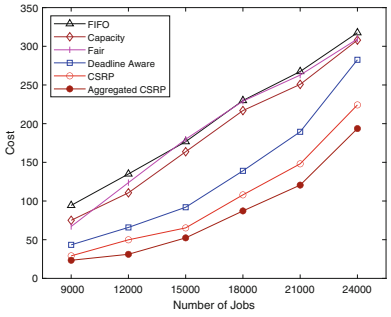

Fig. 4. Overall Cost

simulation shows that there are no idle servers to be deactivated to save energy while using these schedulers except CSRP and Aggregated CSRP. And CSRP and Aggregated CSRP have similar performance in terms of energy saving, Fig. 2 shows the number of inactive servers of CSRP, indicating that they achieve energy saving. As the number of jobs increases, the energy saving decrease, because more servers are necessary to compute more jobs. All servers will be occupied with running jobs when many jobs require computing, resulting in no energy saving.

Figure 3 shows the network cost of each scheduler. The results show that the network cost follows $Deadline\ aware > FIFO > Fair > Capacity > CSRP > Aggregated\ CSRP$. Comparing schedulers do not consider path planning and transmit information through the shortest path. The results imply that CSRP can reasonably plan paths and improve network bandwidth utilization. Further observations show that CSRP reduces network cost by 5.6%, 30.4%, and 36.3% compared with $Capacity$, $FIFO$, and $Deadline\ aware$ respectively. CSRP can achieve better performance than $FIFO$ and $Deadline\ aware$ because it has a global view of path planning. And the results show that Aggregated CSRP reduces network cost by 40.1% compared to CSRP, indicating that

Aggregated CSRP can greatly reduce network bandwidth usage compared with CSRP through path aggregation within the network.

Figure 4 shows that the cost (defined by equation (11)) follows $FIFO \approx Capacity \approx Fair > Deadline\ aware > CSRP > Aggregated\ CSRP$. The results show that CSRP and Aggregated CSRP are more effective in minimizing the objective function than the compared schedulers. Further observations show that CSRP reduces the cost by 53.7% and 25.2% compared with $FIFO$ and *Deadline aware* respectively. Aggregated CSRP reduces the cost by 21.4% by reducing network cost. Figure 4 indicates that the proposed algorithms are effective in satisfying multi-objectives.

6 Conclusions

This paper aims to achieve the deadline guarantee, energy saving, and efficient network bandwidth usage objectives simultaneously in computing clusters. In this paper, we formulate the problem of Job Scheduling and Routing Planning using integer programming with multi-objectives. And we propose the CSRP algorithm based on particle swarm and path aggregation CSRP algorithm to solve this problem. Simulation experiments show that CSRP and Aggregated CSRP are effective in achieving multi-objectives compared to other schedulers.

Acknowledgment. This work is supported in part by the National Key R&D Program of China under Grant No. 2022YFB2901900, supported in part by the National Natural Science Foundation of China (grant no. 61971028), supported in part by the Major Key Project of PCL under grant PCL2023AS5-2. Corresponding author: Chengxiao Yu (yuchx@pcl.ac.cn).

References

1. Rasley, J., Rajbhandari, S., Ruwase, O., He, Y.: Deepspeed: system optimizations enable training deep learning models with over 100 billion parameters. In: Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, pp. 3505–3506 (2020)
2. Sevilla, J., Heim, L., Ho, A., Besiroglu, T., Hobbhahn, M., Villalobos, P.: Compute trends across three eras of machine learning. arXiv e-prints (2022)
3. Li, M., Zhou, L., Yang, Z., Li, A., Xia, F., Andersen, D. G., Smola, A.: Parameter server for distributed machine learning. In: Big Learning NIPS Workshop, vol. 6, no. 2 (2013)
4. Tian, F., Zhang, Y., Ye, W., Jin, C., Wu, Z., Zhang, Z.L.: Accelerating distributed deep learning using multi-path RDMA in data center networks. In: Proceedings of the ACM SIGCOMM Symposium on SDN Research (SOSR), pp. 88–100 (2021)
5. Vavilapalli, V. K., et al.: Apache hadoop yarn: yet another resource negotiator. In: Proceedings of the 4th Annual Symposium on Cloud Computing, pp. 1–16 (2013)
6. Apache Capacity Scheduler Guide. https://hadoop.apache.org/docs/stable/hadoop-yarn/hadoop-yarn-site/CapacityScheduler.html. Accessed 25 Jun 2023
7. Apache Capacity Scheduler Guide. https://hadoop.apache.org/docs/stable/hadoop-yarn/hadoop-yarn-site/FairScheduler.html. Accessed 25 Jun 2023

8. Gao, Y., Zhang, K.: Deadline-aware preemptive job scheduling in hadoop yarn clusters. In: 2022 IEEE 25th International Conference on Computer Supported Cooperative Work in Design (CSCWD), pp. 1269–1274 (2022)
9. Baruah, S.: Improved uniprocessor scheduling of systems of sporadic constrained-deadline elastic tasks. In: Proceedings of the 31st International Conference on Real-Time Networks and Systems, pp. 67–75 (2023)
10. Liu, Q., Fang, Z.: Learning to schedule tasks with deadline and throughput constraints. In: IEEE INFOCOM, pp. 1–10 (2023)
11. Chung, A., Park, J.W., Ganger, G.R.: Stratus: cost-aware container scheduling in the public cloud. In: Proceedings of the ACM Symposium on Cloud Computing, pp. 121–134 (2018)
12. Grewal, S.K., Mangla, N.: Deadline and cost optimization based task scheduling (DCOTS) in cloud computing environment. In: 2023 4th International Conference on Intelligent Engineering and Management (ICIEM), pp. 1–6 (2023)
13. van Rijsbergen, C.J.: Information Retrieval. Butterworth (1979)
14. Garey, M.R., Johnson, D.R.: Computers and intractability: a guide to the theory of np-completeness. R. V. (1980)
15. Reiss, C., Wilkes, J., Hellerstein, J.L.: Google Cluster-Usage Traces: Format+ Schema. Google Inc., Mountain View (2011)
16. Singh, A., et al.: Jupiter rising: a decade of clos topologies and centralized control in google's datacenter network. In: ACM SIGCOMM Computer Communication Review, pp. 183–197 (2015)
17. Appuswamy, R., Gkantsidis, C., Narayanan, D., Hodson, O., Rowstron, A.: Scale-up vs scale-out for hadoop: time to rethink?. In: Proceedings of the 4th Annual Symposium on Cloud Computing, pp. 1–13 (2013)