ISSN(Online): 2320-9801
ISSN (Print) : 2320-9798

International Journal of Innovative Research in Computer and Communication Engineering

(A High Impact Factor, Monthly, Peer Reviewed Journal)

Website: www.ijircce.com

Vol. 5, Issue 11, November 2017

Cloud Infrastructure Resource Allocation for BigData Applications

Priyanka G[1], Arpita Deshpande[2], Anusha Rao[3], Ashik G G[4],Kavitha S[5]

Assistant Professor, Department of Computer Engineering, Global Academy of Technology. RR.Nagar, Bangalore, India[1]

UG Student, Department of Computer Engineering, Global Academy of Technology. RR.Nagar, Bangalore, India[2]

UG Student, Department of Computer Engineering, Global Academy of Technology. RR.Nagar, Bangalore, India[3]

UG Student, Department of Computer Engineering, Global Academy of Technology. RR.Nagar, Bangalore, India[4]

UG Student, Department of Computer Engineering, Global Academy of Technology. RR.Nagar, Bangalore, India[5]

ABSTRACT: In this paper, we present a multi-objective optimization algorithm to trade off the performance, availability and cost of Big Data applications running on the cloud. After analyzing and modeling the interlaced relations among these objectives, we design and implement our approach on experimental environment. Increasing popular big data applications bring challenges to industrial community and academia. Cloud computing with unlimited resources seems to be the way out.However,this solution cannot play its role it cannot arrange fine allocation for cloud infrastructure resources. Finally, three sets of experiments show that this approach can run about 20% faster than traditional optimization approaches and can achieve about 15% higher performance.

KEYWORDS: -Cloud Infrastructure, Big Data, Resource Allocation, Multi-Objective Optimization

I. INTRODUCTION

The term ‘Big Data’ describes innovative techniques and technologies to capture, store, distribute, manage and analyze petabyte-or larger-sized datasets with high-velocity and different structures. Big data can be structured, unstructured or semi-structured, resulting in incapability of conventional data management methods. Data is generated from various different sources and can arrive in the system at various rates. In order to process these large amounts of data in an inexpensive and efficient way, parallelism is used. Big Data is a data whose scale, diversity, and complexity require new architecture, techniques, algorithms, and analytics to manage it and extract value and hidden knowledge from it. Hadoop is the core platform for structuring Big Data, and solves the problem of making it useful for analytics purposes. Hadoop is an open source software project that enables the distributed processing of large data sets across clusters of commodity servers.

One important quality of cloud computing is in aggregation of resources and data into data centers on the Internet. The present cloud services (IaaS, PaaS and SaaS) take in better execution effectiveness by aggregating application execution environments at different levels involving server, OS and middleware levels for distributing them. It offers computers as physical or more often as virtual machines (VMs). A cluster of VMs, virtual cluster, is often requested as a platform for users to run parallel or distributed applications such as MapReduce and Dryad applications. In order to get high throughput, fast response, load balance, low cost, and low price, many topics on VM configuration, VM placement, VM consolidation, and VM migration are explored. The network topology of a virtual cluster has a significant impact on the execution of applications running on it because the physical nodes where VMs are located can be linked in different ways.

The shorter the distance, the closer the virtual cluster. The shortest distance problem is presented to obtain the closest virtual cluster. We solve the shortest distance problem by formulating it into an integer linear programming. A heuristic VM placement algorithm is put forward to provision a virtual cluster. It is designed for MapReduce applications to improve the shuffle speed and accelerate the execution. We also optimize the virtual cluster from the

ISSN(Online): 2320-9801
ISSN (Print) : 2320-9798

International Journal of Innovative Research in Computer and Communication Engineering

(A High Impact Factor, Monthly, Peer Reviewed Journal)

Website: www.ijircce.com

Vol. 5, Issue 11, November 2017

global angle, i.e., provisioning virtual clusters for a request queue rather than a single request. The online heuristic VM placement algorithm and the global sub-optimization algorithm are compared by simulations. The former has lower time complexity while the latter returns shorter average distance for multiple requests. We analyze the performance of our approach through experiments. In the experiment, we adopt different virtual cluster architectures to test different MapReduce applications. Two metrics of application runtime and cluster affinity show the efficiency of virtual cluster optimization.

II. Related work

In [1] authors proposed and evaluated two effective load balancing approaches to data skew handling for MR-based entity resolution. Note that MR's inherent vulnerability to load imbalances due to data skew is relevant for all kind of pairwise similarity computation, e.g., document similarity computation and set-similarity joins. Such applications can therefore also benefit from our load balancing approaches though we study MR-based load balancing in the context of ER only. The BlockSplit approach is conceptionally simpler than Pair Range but achieves already excellent results.
In [2] authors followed a different approach to increase network performance: we reduce the amount of traffic in the shuffle phase. Systems like MapReduce already exploit the fact that most reduce functions are commutative and associative, and allow aggregation of intermediate data generated at a server using a combiner function. It has also been shown that in oversubscribed clusters, performance can be further improved by performing a second stage of partial aggregation at a rack-level. However, in it is shown that at scale performing rack-level aggregation has a small impact on performance. One of the reasons is that the network link of the aggregating server is fair-shared across all the other servers in the rack and becomes a bottleneck. We have been exploring the benefit of pushing aggregation into the core network, rather than performing it just at the edge. According to, the average final output size in Google jobs is 40.3% of the intermediate data set sizes. In the Facebook and Yahoo jobs analyzed, the reduction in size between the intermediate and the output data is even more pronounced: in 81.7% of the Facebook jobs with a reduce phase, the final output data size is only 5.4% of the intermediate data size (resp. 8.2% in 90.5% of the Yahoo jobs). This demonstrates that there is opportunity to aggregate data during the shuffle phase and, hence, to significantly reduce the traffic. In the current network environment, that relying on a single terminal to check the Trojan virus is considered increasingly unreliable. [3] This paper analyzes the characteristics of current cloud computing, and then proposes a comprehensive real-time network risk evaluation model for cloud computing based on the correspondence between the artificial immune system antibody and pathogen invasion intensity. The paper also combines assets evaluation system and network integration evaluation system, considering from the application layer, the host layer, network layer may be factors that affect the network risks. The experimental results show that this model improves the ability of intrusion detection and can support for the security of current cloud computing. The model deletes mutated self-antigens in time through surveillance. The falsenegative error is reduced.In [4] explores how two binary DDP patterns, i.e., CoGroup and Match, could also be used in these tools. We re -implemented an existing bioinformatics tool, called CloudBurst, with three different DDP pattern combinations. We identify two factors, namely, input data balancing and value sparseness, which could greatly affect the performances using different DDP patterns. Our experiments show: (i) a simple DDP pattern switch could speed up performance by almost two times; (ii) the identified factors can explain the differences well. To understand the differences, we identified two affecting factors, namely input data balancing and value sparseness, on their performance differences. The feasibility of these two factors is verified through experiments. We believe many tools in bioinformatics and other domains have a similar logic with CloudBurst as they need to match two input datasets, and therefore could also benefit from our findings. For future work, we plan to investigate more tools that are suitable for multiple DDP patterns and their performances on other DDP engines like Hadoop, which will generalize our findings. We will also study how to utilize the identified factors to automatically select the best DDP pattern combination from multiple available ones. In [5] authors make the case for a declarative foundation for data-intensive machine learning systems. Instead of creating a new system for each specific flavor of machine learning task, or hard-coding new optimizations, argue for the use of recursive queries to program a variety of machine learning algorithms. By taking this approach, database query optimization techniques can be utilized to identify effective execution plans, and the resulting runtime plans can be executed on a single unified data-parallel query processing engine. In essence, the separation identifies "modules" (such as the plan execution engine or the optimization of the logical Datalog program) where localized enhancements lead to higher overall efficiency. To illustrate our approach, we will describe a new high-level programming language called ScalOps, in which the data scientist encodes their

machine learning task.They described the ScalOps translation into Datalog for the example of a specific class of supervised machine learning algorithms. Lastly, demonstrated that an appropriately chosen data-intensive computing substrate, namely Hyracks, is able to handle the computational require-ments of such programs through the application of dataflow processing techniques like those used in parallel databases.

III. PROPOSED ALGORITHM

A. *Problem Statement:*

The infrastructure data allocation problem in clouds includes three constraints which are cost, performance and availability. The optimization solution data allocation deployment is to achieve the highest performance and availability with lowest cost. In general, the higher the performance and availability, the higher the cost . Heterogeneous data is the important characteristics of cloud-based application. As a result we need to take heterogeneity into considerations, which data allocation problem extremely complicated to solve.

B. *Description of the Proposed Algorithm:*

In this paper our main scope is to present a multi objective optimization algorithm to trade of the performance, availability and cost of Big data applications running on cloud.We have six steps in proposed system:

1. Data Uploading
2. Segmentation
3. Task Assignment
4. Data Loading
5. Processing of Task
6. Evaluation process

Step 1: Data Uploading :

The master schedules map tasks in the workers by taking into account of data locality. The output of the map tasks is divided into as many partitions as the number of reducers for the job. Entries with the same intermediate key should be assigned to the same partition to guarantee the correctness of the execution. All the intermediate key/value pairs of a given partition are sorted and sent to the worker with the corresponding reduce task to be executed. Default scheduling of reduce tasks does not take any data locality constraint into consideration. As a result, the amount of data that has to be transferred through the network in the shuffle process may be significant.

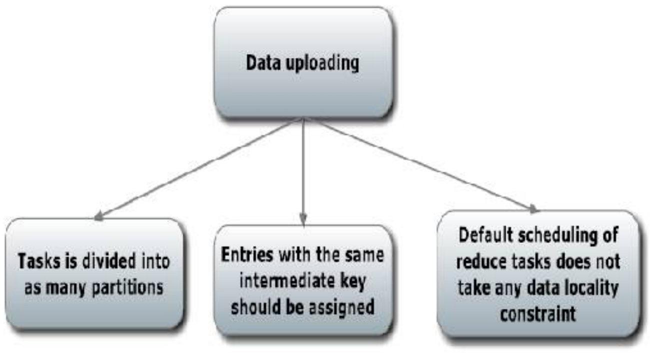

Data uploading
Tasks is divided into as many partitions
Entries with the same intermediate key should be assigned
Default scheduling of reduce tasks does not take any data locality constraint

Fig.1: Data uploading

Step 2: Segmentation:

The system consider a typical MapReduce job on a large cluster consisting of a set N of machines. We let d xy denote the distance between two machines x and y, which represents the cost of delivering a unit data. When the job is executed, two types of tasks, i.e., map and reduce, are created. The sets of map and reduce tasks are denoted by M and R, respectively, which are already placed on machines. The input data are divided into independent chunks that are processed by map tasks in parallel. The generated intermediate results in forms of key/value pairs may be shuffled and sorted by the framework, and then are fetched by reduce tasks to produce final results.

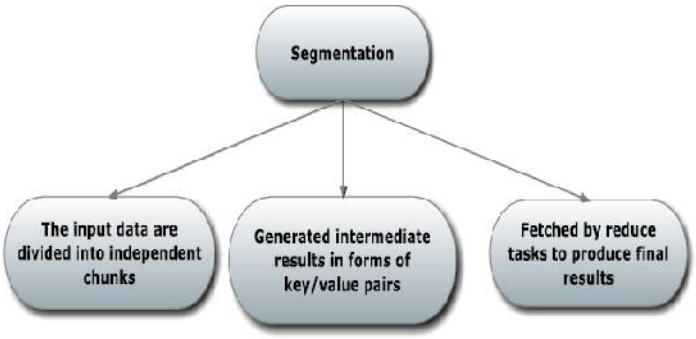

Segmentation
The input data are divided into independent chunks
Generated intermediate results in forms of key/value pairs
Fetched by reduce tasks to produce final results

Fig.2: Segmantation

Step 3: Task Assignment :

The access tier is made up of cost-effective Ethernet switches connecting rack VMs. The access switches are connected via Ethernet to a set of aggregation switches which in turn are connected to a layer of core switches. An inter-rack link is the most contentious resource as all the VMs hosted on a rack transfer data across the link to the VMs on other racks. Our VMs are distributed in three different racks, and the map-reduce tasks are scheduled.

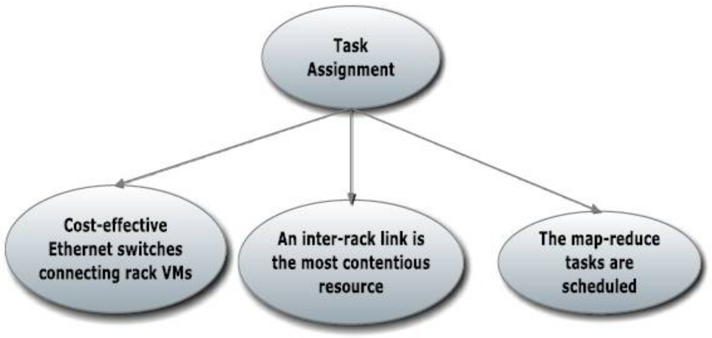

Task Assignment
Cost-effective Ethernet switches connecting rack VMs
An inter-rack link is the most contentious resource
The map-reduce tasks are scheduled

Fig.3: Task Assignment

Step 4: Data Loading :

The number of reducers is changed from 1 to 6. We observe that the highest network traffic is achieved when there is only one reduce task under all algorithms. That is because all key/value pairs may be delivered to the only reducer that locates far away, leading to a large amount of network traffic due to the many-to-one communication pattern. As the number of reduce tasks increases, the network traffic decreases because more reduce tasks share the load of intermediate data.

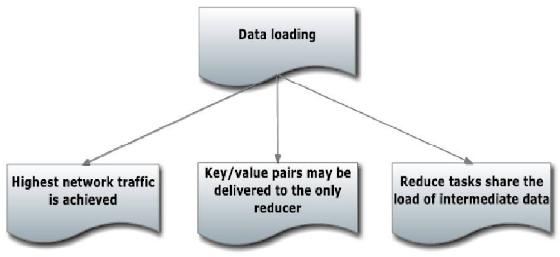

Data loading
Highest network traffic is achieved
Key/value pairs may be delivered to the only reducer
Reduce tasks share the load of intermediate data

Fig.4: Data loading

Step 5: Processing of Task:

The system divide the execution of a MapReduce job into several time slots with a length of several minutes or an hour. We denote the parameters collected at time slot t with no assumption about their distributions. As the job is running, an existing data partition and aggregation scheme may not be optimal anymore under current . To reduce traffic cost, we may need to migrate an aggregator from machine j to j0 with a migration cost jj 0. Meanwhile, the key assignment among reducers is adjusted.

Step 6: Evaluation process:

The system evaluate the performance of proposed algorithm under online cases by comparing it with other two schemes: OHRA and OHNA, which are online extension of HRA and HNA, respectively. The default number of mappers is 20 and the number of reducers is 5. The maximum number of aggregators is set to 4 and we also vary it to examine its impact.

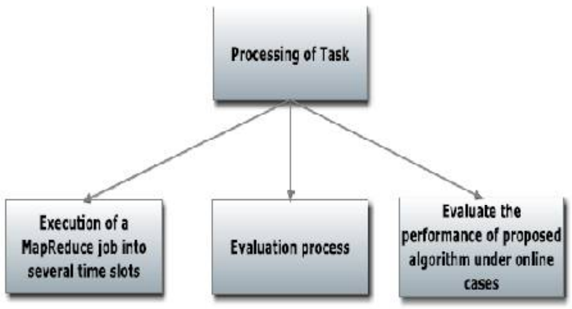

Processing of Task
Execution of a MapReduce job into several time slots
Evaluation process
Evaluate the performance of proposed algorithm under online cases

Fig.5: Task processing

Algorithm

BRA ALGORITHM:

1) Build three tables for recording every sub-solution with lowest cost, highest performance, and highest availability. Set the amount of VMs to 0.

2) Add the amount of VMs by 1. Traverse all VMs to find whether there is some solution can meet all the requirements with 1 VM. If the result is YES, return the one with lowest cost, highest performance, and highest availability. If NO, go to step 3.

3) Update the three tables with the sub-solutions with the lowest cost, highest performance, and highest availability.

4) Add the amount of VMs by 1. Calculate current required response time using the highest performance sub-solution obtained from previous step. Using current required response time to find candidate VMs, and calculate the total cost and availability.

5) Check the solutions obtained from previous step to determine whether some of them can meet all the requirements. If the result is YES, return the one with lowest cost, highest performance, and highest availability. If NO, go to step 3.

V. Simulation Results

This section describes the screens of the "Cloud Infrastructure Resource Allocation for Big data Applications". The snapshots are shown below for each:

ISSN(Online): 2320-9801
ISSN (Print) : 2320-9798

International Journal of Innovative Research in Computer and Communication Engineering

(A High Impact Factor, Monthly, Peer Reviewed Journal)

Website: www.ijircce.com

Vol. 5, Issue 11, November 2017

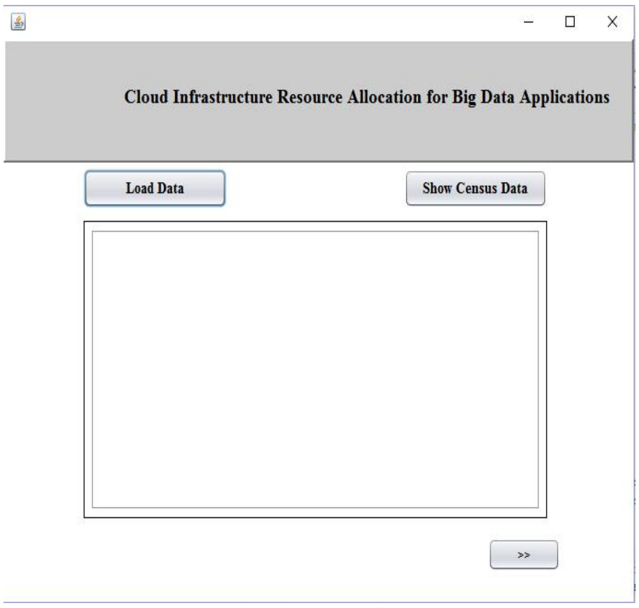

Cloud Infrastructure Resource Allocation for Big Data Applications
Load Data
Show Census Data
>>

Fig.6: Data uploading window

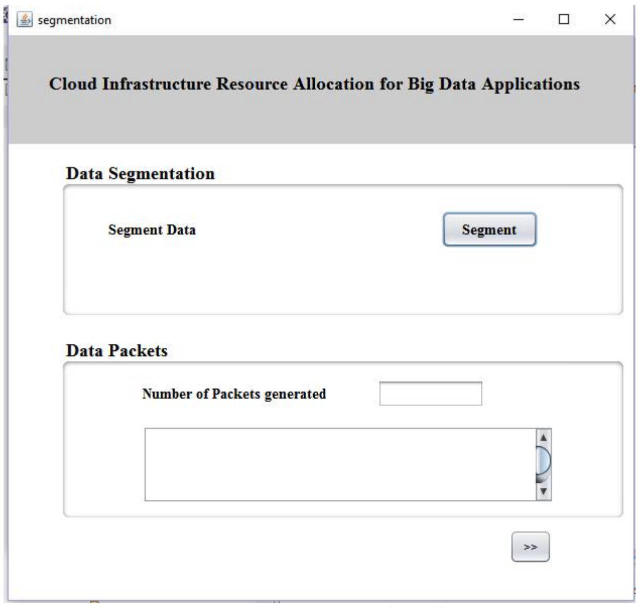
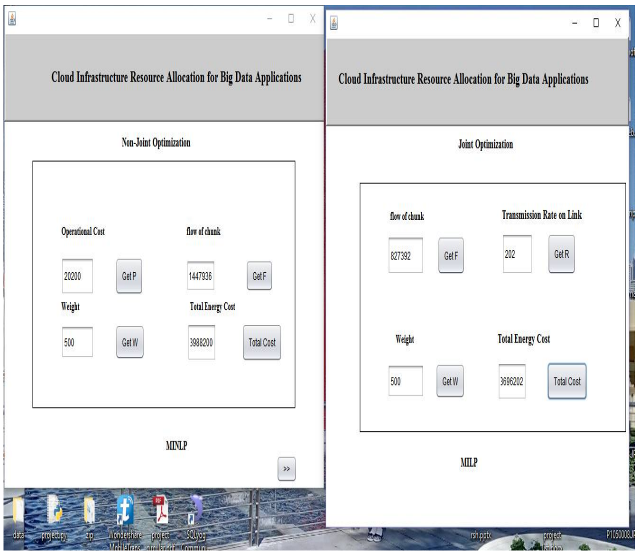

segmentation
Cloud Infrastructure Resource Allocation for Big Data Applications
Data Segmentation
Segment Data
Segment
Data Packets
Number of Packets generated
>>

Fig.7: Data segmentation and packets count


Cloud Infrastructure Resource Allocation for Big Data Applications
Non-Joint Optimization
Get P
Get F
Get W
Total Cost
MINLP
>>
Cloud Infrastructure Resource Allocation for Big Data Applications
Joint Optimization
Get F
Get R
Get W
Total Cost
MILP

Operational Cost 20200
flow of chunk 1447936
Weight 500
Total Energy Cost 3988200

flow of chunk 827392
Transmission Rate on Link 202
Weight 500
Total Energy Cost 3696202

Fig.8: Joint and Non joint optimization

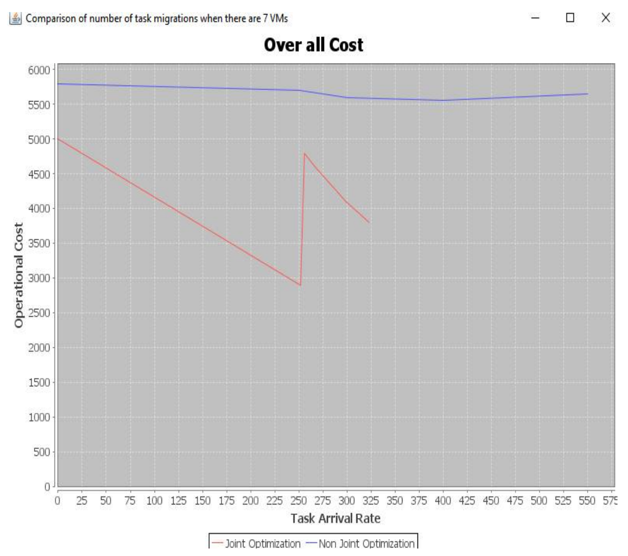


Fig.9: Comparison of task migrations for & VM's

VI. CONCLUSION AND FUTURE WORK

By analysing the relations among the cost, performance, and availability of one cloud-based big data application, and built three models. Based on these three models implement a BRA algorithm to obtain the optimal solution meeting all requirements. Then designed and implemented a complete approach to allocate resources of big data application running on cloud. Finally, the three sets of SLAs to verify the feasibility of our approach, and compared it with other approaches to show the effectiveness. In future, expand the research in two aspects. The first one is to add more constraints, including the security and data processing preference. The second one is to test our approach on advanced networking environments, such as Software- Defined Networking (SDN). Since the input for data uploading is taken as unstructured text file of any size, in future it can be enhanced with structured data, images, videos, semi structured data of any size can be used as an input for data uploading and sent for the further process of the segmentation.

REFERENCES

1. L. Fan, B. Gao, X. Sun, F. Zhang, and Z. Liu, "Improving the load balance of mapreduce operations based on the key distribution of pairs,"arXiv preprint arXiv:1401.0355, 2014.
2. W. Wang, K. Zhu, L. Ying, J. Tan, and L. Zhang, "Map task scheduling in mapreduce with data locality: Throughput and heavy-traffic optimality," in INFOCOM, 2013 Proceedings IEEE. IEEE, 2013, pp. 1609–1617.DilipKumar S. M. and Vijaya Kumar B. P. ,
3. F. Chen, M. Kodialam, and T. Lakshman, "Joint scheduling of pro-cessing and shuffle phases in mapreduce systems," inINFOCOM, 2012 Proceedings IEEE. IEEE, 2012, pp. 1143–1151.
4. Y. Wang, W. Wang, C. Ma, and D. Meng, "Zput: A speedy data uploading approach for the hadoop distributed file system," in Cluster Computing (CLUSTER), 2013 IEEE International Conference on. IEEE, 2013, pp. 1–5.
5. T. White,Hadoop: the definitive guide: the definitive guide. " O'Reilly Media, Inc.", 2009.
6. S. Chen and S. W. Schlosser, "Map-reduce meets wider varieties of applications,"Intel Research Pittsburgh, Tech. Rep. IRP-TR-08-05, 2008.
7. J. Rosen, N. Polyzotis, V. Borkar, Y. Bu, M. J. Carey, M. Weimer,T. Condie, and R. Ramakrishnan, "Iterative mapreduce for large scale machine learning," arXiv preprint arXiv:1303.3517, 2013.
8. S. Venkataraman, E. Bodzsar, I. Roy, A. AuYoung, and R. S. Schreiber, "Presto: distributed machine learning and graph pro-cessing with sparse matrices," in Proceedings of the 8th ACM European Conference on Computer Systems. ACM, 2013, pp. 197–210.
9. A. Matsunaga, M. Tsugawa, and J. Fortes, "Cloudblast: Combin-ing mapreduce and virtualization on distributed resources for bioinformatics applications," ineScience, 2008. eScience'08. IEEE Fourth International Conference on. IEEE, 2008, pp. 222–229.
10. J. Wang, D. Crawl, I. Altintas, K. Tzoumas, and V. Markl, "Com-parison of distributed data-parallelization patterns for big data analysis: A bioinformatics case study," inProceedings of the Fourth International Workshop on Data Intensive Computing in the Clouds (DataCloud), 2013.