Resource Allocation in Datacenters for Data-Parallel Frameworks

Aseem Raj Baranwal UG201210005 | Sachin Grover UG201210030 | Nishit Parekh UG201210042 | Purvi Tiwari UG201213027

I. Abstract

Datacenters are the standard computing platforms for web service providers like Google and Facebook where diverse applications from PageRank to machine learning are implemented. These applications are computationally intensive and often to deal with the large amount of data processing, parallel computing is used. Data parallel frameworks like Map Reduce and Dryad are at the core of most of these applications. Every Map Reduce job is split into various segments. Each of these segments is processed parallel with map computations tasks and the intermediate results obtained after this first step are moved through the data center to be processed by the reduce computation tasks. The performance of each of these applications depends on computation resources like CPU and memory as well as network resources such as bandwidth. This proposal aims at maximizing the performance of data parallel applications by determining the allocation strategy for the bandwidth (network resource) among multiple applications.

The important thing to note is that this proposal is presented for private datacenters. The notion of fairness is commonly accepted in all literature when it comes to bandwidth allocation. Traditionally, VMs have been considered as units of allocation, especially in public datacenters, because they follow the practice of *pay as you go*. Inspired from [2], a scheme of application based allocation of bandwidth for ensuring weighted- performance centric fairness, has been proposed;Ṫhis idea serves as the guideline principle of this project and simply states that application with same priority sharing resources of the same private datacenter must enjoy same performance. The goal is to optimize network performance of these parallel computing applications with performance centric fairness under consideration. Throughout this paper, Map Reduce is used as the representative framework for data parallel applications, primarily because it is the most studied as well as used framework.

II. Introduction

With the computation and storage gradually moving towards the cloud, data centers have emerged as indispensable role players who support the cloud computing and storage services, and the enterprise networks. Email, navigation, data-analytics and online gaming are to name a few of these services. Most of these services use data parallel applications to account for scalability. Parallel computing applications usually perform in several computation stages that require communication among them. The computation stages are extensively studied. But, it has been found in most practical scenarios the communication stage forms the bottleneck for completion of an application. This is why, the notion of Performance centric fairness was introduced for bandwidth allocation.

A. Overview and Problem Statement

So far it is known that the system consists of private datacenter which deals with data parallel applications. Bandwidth resources are allocated to a data parallel application in the communication stage and their performance can be aptly measured using the transfer time taken by the slowest task of an application i.e. the time for data transfer in the communication stage.

The most important trait of data parallel frameworks is the presence of a *barrier* which simply means that an application won't complete until all its constituent flows are completed. Again taking the example of MapReduce, any job would not finish until the reduction process is finished, hence barrier exists at the end of a job. Barrier exists at the end of a computation stage, so clearly the performance of an application depends on the time when the last computation task completes. This approach assume without loss of generality, the computation times for all parallel l tasks to be the same. Common techniques of load balancing [3] and resource allocation back this assumption. On the contrary, the transfer time in the communication stage is variable as different applications are competing for the same network resources. Therefore it is valid to gauge the performance of an application (for a globally shared network resource) in terms of it transfer time - the completion time of the slowest flow in the communication stage.

The main idea behind this approach is that the bandwidth allocation strategy should be aware of the *barrier*. The allocation should take place such that the amount of bandwidth assigned to each flow should be proportional to the data being transferred via that flow, so that all these flows terminate at the same time. The argument being simple, the bandwidth of a faster flow is reduced such that it finishes at the same time as the slowest flow. The performance of the application hasnt been affected,

effectively as well as the saved bandwidth can better utilized by the slowest flow of some other application to improve its performance. Here an important distinction is needed between two types of communication patterns for the application in the private datacenter under consideration. Suppose there is an application A with tasks A_1, A_2, A_3 and A_4. Assuming, 500 MB of intermediate data was generated by two tasks: A_1 and A_2 (250 MB each) needs to be transmitted to A_3 and A_4. Now if it is a MapReduce type application which employs a data shuffle, the data generated by each task A_1 and A_2 is split into two sets of 125 MB each and these are sent to A_3 and A_4. On, the other hand if a broadcast communication is considered, then all the data is sent to all the receivers. A_1 sends 250 MB data to both A_3 and A_4. Similar behavior is shown by A_2 also.

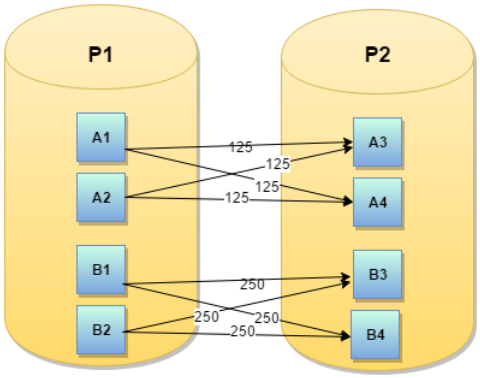

Fig. 1: Two server system for Performance-Centric Fair Allocation

To define Performance-centric fairness consider an application with shuffle communication pattern, let it be A, and an application with broadcast communication pattern, let it be B. The same type of instance is considered as above: Each application's two tasks have to send 500 MB of data to the other two tasks. Let the tasks be labelled as $\{A_1,A_2, A_3,A_4\}$ and $\{B_1,B_2, B_3,B_4\}$ respectively. As show in figure the sending tasks: $\{A_1,A_2, B_1,B_2\}$ be on one server S_1 with an egress bandwidth of 500 MB/sec and similarly the receiving tasks: $\{A_3,A_4, B_3,B_4\}$ be on another server, S_2 with ingress bandwidth 500 MB/sec. Let the priorities of the two applications be denoted by weights w_A and w_B. Given that higher weights mean higher priorities. Given t_A and t_B as the transfer times for A and B respectively. Performance centric fairness implies $t_A : t_B = w_A : w_B$. Suppose both applications have equal weights, i.e. $w_A = w_B$ then flows of A can be assigned a bandwidth of $125/3$ MB/sec and links of B can be assigned bandwidth of $250/3$ MB/sec which implies both these applications will terminate in 3 seconds which is Performance-centric fair. In the same setup, if A would be twice as important as B, i.e. $w_A : w_B = 2 : 1$, then bandwidth assigned to B would be equal to that of A because B has to transfer twice as much data. So if we assign links of A as well as B a bandwidth of $125/2$ MB/sec, A will terminate in 2 seconds whereas B will terminate in 4 seconds, again $w_A : w_B = t_A : t_B = 2 : 1$, which is weighted performance-centric fair.

This notion of fairness is defined at the application which is quite different from traditional concepts of fairness which mostly exist at flow level(TCP), VM level [6] or tenant level[7] that exist in the literature on datacenters.

Two problems that are investigated in this proposal. First, is the problem of ensuring a performance-centric fair allocation of the bandwidth. A central as well as a distributed approach is presented. A tunable degree of relaxation has been presented to maximize social welfare. The optimizaiton problem is solved using *dual based decomposition*. The Second step is the modeling of server failures and arrival of new applicaitons to the datacenter. The objective is to jointly optimize task placement on servers as well as ensuring performance-centric allocation of bandwidth. Short term failure are the part and parcel of any datacenter and the adverse effects of such failures cannot be overlooked. The taks on the failed servers have to be relaunched and rescheduled on the remaining servers. Along with this, the communicaiton stage information has to be retransmitted to these alternative servers where the tasks have been placed. The solution has been modelled as a Mixed-Integer Non Linear Program. Although, solving a general MINLP is an open challenge, all practical scenarios pertaining to a datcenter can be easily solved using commercial solvers such as CPLEX.

B. Motivation and Scope

Datacenters are basically centralised physical (or virtual) repositories for storage, management and distribution of data and information. Thus, they serve as a large center for all the world's imminent computing needs, when done over a wired/wireless connection. Cloud computing is one of the most prominent users of datacenters, where the whole business model of a provider depends upon the datacenters it uses for carrying out such tasks. As such, datacenters form a very important part of the whole system of interconnected devices that operate in today's world. Naturally, the load upon every datacenter is tremendous, and hence performance of a datacenter is crucial to the performance of the whole system.

Some common uses of Datacenters are in the following approaches, listed along with their crucial requirements:

- **Remote Storage** (E.g. DropBox): Huge amount of storage + quick I/O
- **Distributed Data** (E.g. Google Maps): Reliable connectivity among different servers
- **Cloud Processing** (E.g. Hadoop): Fast communication between nodes

The growth of Facebook, with more than 1 billion active users, points to the enormous increase in the amount of datacenter usage, thus underlining the potential of such systems. As such, improvements in this area are beneficial to a lot of scenarios, and hence we want to work on solving problems that are rife in this sector.

Parallel computing is integral to almost all kinds of applicaitons that are dealt by private datacenters. Data Parallel frameworks need to have high computation as well as communicaition speeds to ensure efficient and accurate results. In this proposal, the efficiency of the communication stage is the prmary motive as very few works in the past have addresed this problem as compared to the computation stage efficiency which has been extensivel studied.

III. Contributions of this work

Inspired by [2], performance-centric fairness has been modelled. A novel scheme for modelling server failures has been proposed where every time a server fails, the tasks placed on that server can be rescheduled and placed on the existing servers in the datacenter which haven't failed. The already existing work considers a very restricted version of this scenario where it is implicitly assumed that each application has only one task on a failed server which may not be the case if we consider practical situations. The whole idea of parallel computing is based on balancing the communication overhead and data backups. If data is kept in a more scattered fashion then there may be more communication involved where if data is kept in a congregate manner then there is a chance of loss of large amount of data in case the server fails. But, still the assumption that only one task of each application is there on a server is a little too strict. This assumption is removed by modelling the problem as a mixed integer quadratic constraint program.

Besides removing the above mentioned assumption, this proposal also introduces a provision for adding new applications to the system. The new application arrival mechanism is designed by assuming a dummy server in the system that has all tasks of the incoming application. Failure of this dummy server is equivalent to placement of the tasks of the incoming applicaiton over the existing servers in the system.

IV. Deliverables

The deliverables of this project include a theoretical formulation for performance-centric fairness and problem formulation of server failures as a Mixed Integer Non Linear Program (MINLP). Along with the mathematical formulation the deliverable also include implementation of these algorithm of server failures, dynamic application addition mechanism as well as the centralised and distributed approach from [2].

V. Literature survey

This section describes the work done by the researcher community in the field of this project that was read and analyzed, in order to arrive at the solution for the problem.

1) **SecondNet**
   SecondNet explored in [4], aims at solving the problem of Bandwidth Guaranteed service between VMs in a Data center by introducing the concept of a Virtual Data Center (VDC) as a unit of allocation to tenants. It proposes a Centralized allocation algorithm for bandwidth guaranteed physical mapping. The features that make it desirable are scalability and high utilization, flexibility as it has the elasticity to change the SLA to be adjusted according to the customers demands and ability to be deployed with commodity switches and servers which makes it partially topology independent.
   SPT only for signaling purposes, traffic volume is small and the priority is high.
   The static allocation scheme leaves the network underutilized because. Also, to some extent SecondNet is topology dependent. SecondNet only takes into account the intra-datacenter bandwidth for allocation ignoring the Internet-access bandwidth for running web applications.

2) **Oktopus**
   Oktopus [1] proposes the approach of a virtual network, which is offered to tenants, isolating them from the physical network. This network is an abstraction for the tenants, but is also compatible with the physical network underneath. Two proposals namely Virtual Cluster(VC) and Virtual Oversubscribed Cluster(VOC) are put forward. The idea of VC states that all VMs are connected to a single, non-oversubscribed (virtual) switch. This is good for applications that are data-intensive, and characterize all-to-all broadcasting whereas VOCs have an oversubscribed 2-level network and is better for VMs which have large amount of local communication. Oktopus has two major goals: tenant suitability and provider flexibility.

Oktopus has shown that a symbiotic relationship can exist between the tenant and the provider, where explicit requests, using Virtual Networks, can prove beneficial for both. Virtual Networks allow tenants to focus on aspects such as completion time, or a non-performance metric (e.g. reliability, failure resiliency etc). VC proves to be an effective tool to carry out any kind of such abstraction, thus not letting the tenant be affected by the underlying topology. Pricing can be made more intuitive, transparent and efficient, making the tenant pay only for the resources it was supposed to use. Dynamic changes in environment can lead to chain reactions in case of oversubscribed Cluster Allocation and it may happen that unavailable resources may be allocated to tenants. Like SecondNet, it misses to take into consideration the Internet-access bandwidth as a parameter.

3) **Gatekeeper**
Gatekeeper [8] was an approach designed to achieve the objective of providing network performance isolation for co-located untrusted tenants in a virtualized Data Center. The model aims to achieve a balance between determinism and bandwidth utilisation, however, a VM may exceed its minimum guaranteed bandwidth when unused bandwidth is available.
The Gatekeeper architecture uses distributed approach at the virtualization layer and achieves scalability using simple point to point protocols and minimal control state. Minimum bandwidth guarantees are obtained using admission control mechanism. A traditional weight scheduler is used for scheduling transmission bandwidths. Feedback messages are used at the receiver side if the receiver bandwidth exceeds the limit. Each interface has rate limiters and besides the root rate limiter, others are created dynamically. Gatekeeper can deal with misbehaving tenants as well and can utilise unused bandwidth at both transmit and receive ends.

4) **SeaWall**
Seawall [9] tries to improve on TCPs congestion control which originally has the network opened up to DoS attacks and performance interference. It divides network capacity based on an administrator specified policy and computes allocations by tunneling traffic through congestion controlled tunnels. It works irrespective of traffic characteristics like number of flows, protocols or participants. It ensures that along all links, the share of bandwidth obtained by each entity that serves as a traffic source, is proportional to its assigned weight.
Seawall allocates link bandwidth among total number of VMs on the same server belonging to two different tenants because Seawall allocates link bandwidth among the number of communicating VMs through the link, so if there are two VMs on the same server belonging to two different clients and one VM receives traffic from many VMs on other servers while one VM receives traffic from just one sender, then the older client will be allocated with higher server link bandwidth than the latter one.

5) **FairCloud**
In FairCloud [6], there are fundamentally 3 objectives to be optimized for a datacenter network. These are:
Minimum Guarantee *(Providing a minimum bandwidth to every tenant in the datacenter, even in the worst case)*;
Maximum Utilisation *(Ensuring that bandwidth on a link is fully utilised, and that no tenant using that link is deprived of its bandwidth demand, if the link is underutilised)*; and
Network Proportionality *(Guarantee that a tenant, asking for more bandwidth, should not get lesser than a tenant asking for lesser resources).*
Further, the paper goes on to prove that these 3 objectives cannot be optimized simultaneously. Formally, there will exist a fundamental tradeoff between Minimum Guarantee and Network Proportionality, and between Network Proportionality and High Utilisation. Thus, there cannot be a fully-optimal solution to this problem. This is an important implication.
Also, the paper helps realise 5 different parameters, to better understand the tradeoffs. These are: **Work Conservation**, **Strategy-Proofness**, **Utilisation incentives**, **Communication-pattern independence**, **Symmetry**. These parameters help understand the various allocation policies and their desired effects.
These policies demonstrate how it is possible to achieve optimisation on one objective, while trading off on other objectives. By changing various parameters, the objective space can be traversed while getting outputs that cater to at least one of the three objectives. This, in conclusion, provides a crucial way to analyse how the problem can be tackled using a variety of approaches.

6) **NetShare**
NetShare [7] allows the bisection bandwidth of the network to be allocated across services based on simple weights specified by a manager. Bandwidth unused by a service is shared proportionately by other services. More precisely, NetShare provides weighted hierarchical max-min fair sharing, a generalization of hierarchical fair queuing of individual links. In this model, first the manager specifies a small number of services with weights assigned to each class that indicate their relative importance. Then there is some mechanism that allocates network bandwidth in weighted max-min fair fashion among these services. The bandwidth assigned to each service is then recursively divided (again

in max-min fair fashion) among the individual flows for that service. Thus, NetShare can be viewed as a generalization of hierarchical fair queuing on a link by link level to a network level

VI. Brief description of Work Done

A crucial aspect of the problem and its formulation is that: *If the bandwidth of the faster flow is limited to a lower rate, then the **performance will not change** as long as it still completes faster than the slowest flow (the bottleneck).* So this bandwidth can be provided to the slowest flow of another application to increase the performance. [2] proposes two algorithms, among others, for bandwidth allocation to task flows, in order to achieve weighted performance centric fairness. The performance is characterized by the time taken by an application to complete. Weights are the properities of the applications. Thus, fairness ensures that a high priority application performs better. The two algorithms are based on the following mathematical model.

Consider a private data center as depicted in figure 1, with **'K' data parallel applications** running concurrently, their tasks distributed across **'N' physical machines**. On each physical machine/server n $\in \{1, 2, ..., N\}$, tasks from different applications will share its link bandwidth. A full bisection bandwidth network is assumed, where practically bandwidth is only bottlenecked at the access links of physical machines. So completion time of each flow is determined by the bandwidth at each access links. Given:

k	: An application $\in \{1, 2, ..., K\}$	m_k	: Number of tasks required by kth application
T_k	: Set of tasks required by kth application	D_k	: The network load matrix for application k
B_n^E	: Egress link capacity	B_n^I	: Ingress link capacity
r_k^{ij}	: Bandwidth for (i, j) communicating task pair of k	t_k	: Transfer time = $max_{ij} D_k^{ij}/r_k^{ij}$
w_k	: Weight associated with the application signifying its importance		

weighted performance-centric fairness is said to be achieved when for an application k, the following holds:
$\frac{1}{t_k} = \frac{w_k}{\sum_k w_k} S$, where S is called the **total performance-centric share**, which is upper bounded given the fixed amount of bandwidth capacity in the data center.

The objective is to achieve this upper bound. Substituting $t_k = max_{ij} \frac{D_k^{ij}}{r_k^{ij}}$ gives:

$min_{ij} \frac{r_k^{ij}}{D_k^{ij}} = \frac{w_k}{\sum_k w_k} S, \forall\ i, j \in T_k$ and $D_k^{ij} \neq 0$

A binary variable $X_{k,n}^i$ denotes whether task i of application k is placed on server n.

$$X_{k,n}^i = \begin{cases} 1, \text{ when } T_k^i \text{ is placed on server } n \\ 0, \text{ otherwise} \end{cases} \tag{1}$$

The total egress rate of each task T_k^i placed on server n is $\sum_{j, X_{k,n}^j = 0} r_k^{ij} . X_{k,n}^i$. If the task T_k^j receiving the intermediate data from T_k^i is also placed on server n, there will be no data sent through the network. The constraint of $X_{k,n}^j = 0$ in the summation accounts for this case.

Summing over all tasks of an application placed on server n, and summing over all the K applications on that server, the total egress rate of server n is obtained, which must not exceed the egress link capacity:

$$\sum_k \sum_i \sum_{j, X_{k,n}^j = 0} r_k^{ij} . X_{k,n}^i \leq B_n^E \tag{2}$$

The same analysis will apply to the ingress link of each server. To maximize performance along with maintaining weighted performance-centric fairness:

$$\max_{r} \quad S \tag{3}$$

$$\text{s.t.} \quad \min_{i,j} \frac{r_k^{ij}}{D_k^{ij}} = \frac{w_k}{\sum_k w_k} S, \forall\ k \in \{1,2,...,K\}, \forall\ i,j \in T_k, D_k^{ij} \neq 0 \forall\ n \in \{1,2,...,N\} \tag{4}$$

$$\sum_k \sum_i \sum_{j, X_{k,n}^j = 0} r_k^{ij}.X_{k,n}^i \leq B_n^E\ \forall\ n \in \{1,2,...,N\} \tag{5}$$

$$\sum_k \sum_j \sum_{i, X_{k,n}^i = 0} r_k^{ij}.X_{k,n}^j \leq B_n^I\ \forall\ n \in \{1,2,...,N\} \tag{6}$$

Let α_k denote the performance of application k, i.e., the reciprocal of its transfer time:

$$\alpha_k = \frac{1}{t_k} = \min_{ij} \frac{r_{ij}^k}{D_k^{ij}},\ \forall i,\ j \in T_k, D_k^{ij} \neq 0 \tag{7}$$

[2] clearly shows that the the above problem can be modifed to the problem mentioned below keeping the objective value same. The problem is converted into the following:

$$\max_{\alpha} \quad S \tag{8}$$

$$\text{s.t.} \quad \alpha_k = \frac{w_k}{\sum_k w_k} S, \forall\ k \in \{1,2,...,K\} \tag{9}$$

$$\alpha_k \leq \frac{w_k}{\sum_k w_k} S, \forall\ k \in \{1,2,...,K\}, \forall\ i,j \in T_k, D_k^{ij} \neq 0 \tag{10}$$

$$\sum_k \alpha_k \sum_i \sum_{j, X_{k,n}^j = 0} D_{ij}^k.X_{k,n}^i \leq B_n^E\ \forall\ n \in \{1,2,...,N\} \tag{11}$$

$$\sum_k \alpha_k \sum_j \sum_{i, X_{k,n}^i = 0} D_{ij}^k.X_{k,n}^j \leq B_n^I\ \forall\ n \in \{1,2,...,N\} \tag{12}$$

The intuition is that since the performance of each application is determined by the completion time of its slowest flow, it is efficient to make all the flows of an application finish at the same time, by allocating flows the amounts of bandwidth that have the same proportionality to their network load. The flowchart for the central algorithm is provided in figure 2.

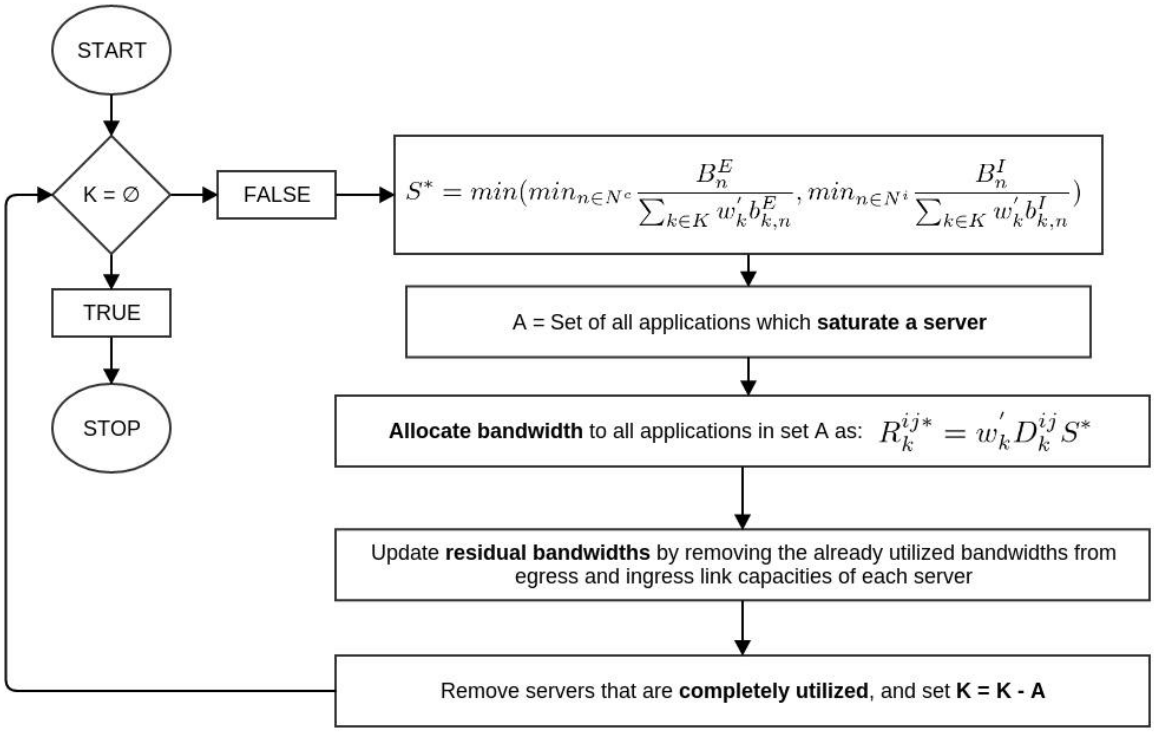

Fig. 2: Central algorithm flowchart

A. The Tradeoff : Fairness vs Utilization

Implementing the two algorithms in [2] for bandwidth allocation (Central and Distributed), it is verified that a tradeoff exists between fairness and efficiency. Following the strict definition of weighted performance-centric fairness results in a reduction in efficiency. We can look at this tradeoff from the perspective of bandwidth utilization by considering an example case: Let two applications A and B have weights 3 and 5 respectively. To keep the allocation absolutely fair by weights, it is required that the bandwidth is allocated to applications A and B in the ratio 3:5. But after this allocation, it might be the case that a server's ingress or egress link is not completely utilized. This counts as usable bandwidth, and allocating it to application A for instance, will although disrupt the strict performance fairness, but will also increase the utilization, and thus the productivity of the data center. Hence comes the notion of *relaxed fairness*, which is iteratively implemented in the two algorithms for bandwidth allocation. Please refer to [2] for the mentioned algorithms.

B. Extending the problem - Server failure

The two algorithms mentioned above give the bandwidth allocation for all the task pairs of all applications assuming that the task placement is already done. Ruling out this assumption is the natural extension of this problem. This calls for reformulating the problem. To formulate this extended problem, some more parameters are required, which are defined below:
On each server n $\in$ {1, 2, 3, ..., N}, tasks from different applications share its CPU resource, with the capacity of C_n, and its link bandwidth, including both the egress link with capacity B_n^E and the ingress link with capacity B_n^I. If we use c_k^i to denote the CPU requirement by task T_k^i, then

$$\sum_k \sum_i c_k^i . X_{k,n}^i \leq C_n \tag{13}$$

This completes the joint optimization algorithm, where the task placement and the bandwidth allocation are jointly optimized as failures occur in private data centers.

When a server fails, the communications of all the tasks running on it would be terminated. As a result, all the intermediate data needs to be retransmitted once these tasks are relaunched on other available servers. The failed tasks are denoted by the set T_f. The set of servers that are undergoing failures is represented by N_f , while N_{-f} denotes the set of servers that are running normally. The problem of task placement is to choose servers from N_f to relaunch the failed tasks in T_f. For differentiation, we use χ = { $\chi_{k,n}^i$ — $T_k^i \in T_{-f}, n \in N_{-f}$ } to represent the placement of failed tasks as variables. For these binary variables, the following constraint must be satisfied:

$$\chi^i_{k,n} \in \{0,1\}, \sum_{n \in N_{-f}} \chi^i_{k,n} = 1, \forall \chi^i_{k,n} \in \chi \tag{14}$$

which indicates that each failed task should be placed on one of the normal servers. Thus, the optimisation problem now becomes the following:

$$\min_{p, \chi^i_{k,n} \in \chi} \quad p \tag{15}$$

$$\text{s.t} \quad \sum_k \omega_k \left(\left(\sum_{i, T_i \notin T_f} X^i_{k,n} + \sum_{i, T_i \in T_f} \chi^i_{k,n} \right) \left(\sum_{j, T_j \notin T_f} \left(1 - X^j_{k,n}\right) + \sum_{j, T_j \in T_f} \left(1 - \chi^i_{k,n}\right) \right) \right) D^{i,j}_k < B^E_n \tag{16}$$

$$\sum_k \omega_k \left(\left(\sum_{i, T_i \notin T_f} X^i_{k,n} + \sum_{i, T_i \in T_f} \chi^i_{k,n} \right) \left(\sum_{j, T_j \notin T_f} \left(1 - X^j_{k,n}\right) + \sum_{j, T_j \in T_f} \left(1 - \chi^i_{k,n}\right) \right) \right) D^{j,i}_k < B^I_n \tag{17}$$

$$\sum_k \left(\sum_{i, T_i \notin T_f} c^i_k . X^i_{k,n} + \sum_{i, T_i \in T_f} c^i_k \cdot \chi^i_{k,n} \right) \leq C_n \tag{18}$$

$$\chi^i_{k,n} \in \{0,1\}, \sum_{n \in N_{-f}} \chi^i_{k,n} = 1 \tag{19}$$

$$\chi^i_{k,n} = 0, \text{if } l(k,i,n) = 1 \tag{20}$$

$$\forall n \in N_{-f}, \forall \chi^i_{k,n} \in \chi \tag{21}$$

Insights and Preliminary Filtering

Solving this model may be time consuming for an extremely large number of servers and applications. Therefore, some insights are required in order to reduce the search space and thus apply a preliminary filter on the available servers on which the failed tasks will be placed.

Four quantities are defined for this filtering:

1) SBD: flow demand of an application to achieve its fair performance share, i.e., $D^k_{ij} w^{'}_k$ for the flow between tasks T^i_k and T^j_k
2) $SBD - t$: sum of SBD over all fows of task t
3) $SBD - n$: sum of $SBD - t$ over all tasks on server n
4) $rSBD - n$: $SBD - n$ divided by link bandwidth capacity B_n of server n

Using these quantities a preliminary filtering algorithm is developed, which is given below:

Algorithm 1 Preliminary Filtering

Initialize the candidate set C = ϕ
Sort failed tasks $T^i_k \in T^i_f$ in a decreasing order of their CPU requirements resulting in a sorted set as T^s_f
for all tasks $T^i_k \in T^s_f$ **do**
  for all server $n \in N_{-f}$ **do**
    if if 20 and 22 are satisfied **then**
      Compute rSBD-n for n if T^i_k is placed on n
      Insert n to S^n_k in increasing order of rSBD-n
  end for
  Add the first $\eta \geq 1$ servers from S^n_k to C, and remove them from N_{-f}
end for
Output:
The candidate set$C$

The main idea of preliminary filtering is that each failed task in its turn selects a number of the most promising candidate servers greedily, according to the $rSBD - n$ for server n if the task is to be placed on it. More specifically, the server n with the smallest $rSBD - n$ is the first to be selected to the candidate set.

Removing the assumption from [2], the linearity of the optimization program is lost and a Mixed Integer Quadratic Constraint Optimization Problem is obtained which is solved with traditional branch and cut algorithm. The algorithm has

been implemented with IBM's CPLEX C API. But, this trade off of linearity gets rid of a very crucial assumption as it now deals with multiple task failures instead of single tasks per application.

Algorithm 2 Joint Task Placement and Performance-Centric Bandwidth Allocation

Input:
Servers with failures: N_f; Normal servers: N_{-f};
Bandwidth capacity: B_n^E, B_n^I , $\forall n \in N_{-f}$;
Network load matrix D_k^{ij} and weight w_k , $\forall k \in K$; Placement of tasks that are running on servers without failures: $X_{k,n}^i \in X$
Output:
Placement of tasks running on servers with failures: $\chi_{k,n}^i \in \chi$
Bandwidth allocation for all application flows: $r_{i,j}^k$
Obtain the candidate set C from Algorithm 1;
2: Branch-and-Cut for the problem over the reduced solution space: $\bar{\chi} = \{\chi_{k,n}^i | T_k^i, T_k^i \in T_f, n \in C\}$;
Relaunch failed task $T_k^i \in T_f$ on server n if $\chi_{k,n}^i = 1$.
4: Allocate bandwidth to flows of $k \in K$.

C. Extending the problem - New application

The next extension to the problem involves accommodation of a new application that enters an active system of servers and applications. This problem can be viewed as synonymous to server failure in the following sense.
Consider a system of N servers and K applications already running in a data center, when a new application, the $(K+1)^{th}$ application enters the system. If this incoming application is considered a part of a hypothetical system with N + 1 servers, the $(N+1)^{\text{th}}$ server having all the tasks of the new application, then if the $(N+1)^{\text{th}}$ server fails, all the tasks of the new application have to be reassigned to the remaining N servers of the hypothetical system, and the bandwidth of flows have to be re-calculated. The problem of accommodating a new application is thus reducible to the problem dealing with a server failure, and can be solved with equal efficiency.

VII. Experimental results/ Proofs

VIII. Future scope of work

The proposal primarily focuses on perforamce-centric fairness for bandwidth allocation. The methods and algorithms introduced above focus on enforcing this fairness scheme strictly or in a relaxed fashion. Task based scheduling forms the core of these approaches. With the increasing popularity of Software Defined Netowkring (SDN) and data intensive applicaitons a path selection based application level scheduling scheme can be designed, similar to the one in [2]. SDN controllers can schedule netwok transfers which can be conveyed by a central manager to the controller. Selecting paths based on barrier awareness can be implemented using SDN as the controller possesses a global view of the complete datacenter network include the states of the switches as well as the demands of each applicaiton. As a private datacenter is under consideration, it is unlikely that applcation or applicaiton managers will misreport their demands (malicious behavor is highly improbable). Path selection strategies can be enforced via the controller by controlling the flows through switches and bandwidth allocation can be enfirced based on the data provided by the central manager to the controller. A lot of work is turning up in the domain of using SDN in data parallel frameworks. [5] uses metrics like makespan and completion times as the basis of algorithms for SDN based big data frameworks. A similar dynamic approach can be included in the exetension of this work.

References

[1] Hitesh Ballani, Paolo Costa, Thomas Karagiannis, and Ant Rowstron. Towards predictable datacenter networks. In *ACM SIGCOMM Computer Communication Review*, volume 41, pages 242–253. ACM, 2011.
[2] Li Chen. Performance-centric bandwidth allocation for data parallel applications in private datacenters. Master's thesis, Edward S. Rogers Sr. Dept. of Electrical and Computer Engineering, University of Toronto, 2015.
[3] Cyril Fonlupt, Philippe Marquet, and Jean luc Dekeyser. Data-parallel load balancing strategies. *Parallel Computing*, 24:1665–1684, 1996.
[4] Chuanxiong Guo, Guohan Lu, Helen J Wang, Shuang Yang, Chao Kong, Peng Sun, Wenfei Wu, and Yongguang Zhang. Secondnet: a data center network virtualization architecture with bandwidth guarantees. In *Proceedings of the 6th International Conference*, page 15. ACM, 2010.
[5] Benxiong Huang Peng Qin, Bin Dai and Guan Xu. Bandwidth-aware scheduling with sdn in hadoop: A new trend for big data. *IEEE Systems Journal*.
[6] Lucian Popa, Arvind Krishnamurthy, Sylvia Ratnasamy, and Ion Stoica. Faircloud: Sharing the network in cloud computing.
[7] Lucian Popa, Arvind Krishnamurthy, Sylvia Ratnasamy, and Ion Stoica. Faircloud: Sharing the network in cloud computing.
[8] Henrique Rodrigues, Jose Renato Santos, Yoshio Turner, Paolo Soares, and Dorgival Guedes. Gatekeeper: Supporting bandwidth guarantees for multi-tenant datacenter networks. In *WIOV*, 2011.
[9] Alan Shieh, Srikanth Kandula, Albert Greenberg, and Changhoon Kim. Seawall: performance isolation for cloud datacenter networks. In *Proceedings of the 2nd USENIX conference on Hot topics in cloud computing*, pages 1–1. USENIX Association, 2010.