Efficient Resource Allocation Algorithm for CERN SC15 Demo

Qiao Xiang
xiangq27@gmail.com

1 Offline Model

1.1 System Settings

We use a directed graph $G = (V, E)$ to denote the network. And we consider the system operating in a discrete-time, where time is divided into time slots with equal length and indexed as $t = 1, 2, \dots, T$. At the beginning of time slot 1, we are given a set of data transfer requests, denoted as $\mathbb{M}$. Each request $m \in \mathbb{M}$ is associate with a priority level $p = 1, 2, \dots, P$, where a larger p represents a higher transfer priority. Hence the set of data transfer can be further partitioned into a series of subsets, denoted as $\mathbb{M}_1, \mathbb{M}_2, \dots, \mathbb{M}_P$. Each request in the same request subset has the same priority level.

Given a data transfer request m, it is composed of several data flows. Each data flow k is defined as a 3-tuple $\{s_k, d_k, v_k\}$. In this tuple, s_k denotes the source location of data, d_k denotes where the data should be sent to, and v_k denotes the volume of data in the request. Each data flow k belongs to only one data transfer request m. For any two data flows $k', k'' \in m$, we have $d_{k'} = d_{k''}$ and $v_{k'} = v_{k''}$. This setting is to represent the case that the data requested by the client have multiple replicas at different endhosts. Because the data volume in each transfer request is usually large, we allow the data in the same transfer request m to be transfered to the client in pieces from different sources, i.e., data slicing and assembling.

Decision Variables: We have two sets of decision variables in the model:

- $r_k(t)$, the data volume of data transfer request k that is transferred from s_k to d_k in time slot t.
- $f_{ij}^k(t)$, the actual data volume of data transfer k along link $(i, j) \in E$.

And we develop the following model for the resource allocation problem (**RAP**) in CERN SC'15 demo.

$$\textbf{RAP} \quad \text{minimize} \quad T \tag{1}$$

subject to

$$\sum_k f_{ij}^k(t) \le u_{ij}(t) \qquad \forall (i,j) \in E \text{ and } t, \tag{2a}$$

$$\sum_{j \in V} f_{ij}^k(t) - \sum_{j \in V} f_{ji}^k(t) = 0 \qquad \forall k, t \text{ and } i \in V/\{s_k, d_k\}, \tag{2b}$$

$$\sum_{j \in V} f_{ij}^k(t) - \sum_{j \in V} f_{ji}^k(t) = r_k(t) \qquad \forall k, t \text{ and } i = s_k, \tag{2c}$$

$$\sum_{j \in V} f_{ij}^k(t) - \sum_{j \in V} f_{ji}^k(t) = -r_k(t) \qquad \forall k, t \text{ and } i = d_k, \tag{2d}$$

$$\sum_{k \in m} \sum_{t=1}^{T} r_k(t) = v_k \qquad \forall m, k, t \tag{2e}$$

$$r_k(t) \ge 0 \qquad \forall k \text{ and } t \le l_k, \tag{2f}$$

$$f_{ij}^k(t) \ge 0 \qquad \forall (i,j) \in E, k \text{ and } t \le l_k, \tag{2g}$$

For simplicity, we assume all data transfer requests have the same priority p in the **RAP** problem. We will discuss the multi-priority case in later sections. **RAP** aims to minimize the maximum transfer time T of all data transfer requests. Even with this simplification, finding such T is very challenging. To this end, we develop an optimal max-min fair resource allocation algorithm in the next subsection.

1.2 An Optimal Max-Min Fair Resource Allocation Algorithm

To present the optimal max-min resource allocation (OMFRA) algorithm, we first define the Max-Min Fair $\mathbf{MMF}$ problem for each time slot t. [1]

$$\mathbf{MMF}(t) \quad \text{maximize} \quad z(t) \tag{3}$$

subject to

$$\sum_k f_{ij}^k(t) \le u_{ij}(t) \qquad \forall (i,j) \in E \tag{4a}$$

$$\sum_{k \in m} r_k(t) \ge z(t) \cdot v_k \qquad \forall m \in M_{unsat}(t) \tag{4b}$$

$$\sum_{k \in m} r_k(t) = z_{sat}^m(t) \cdot v_k \qquad \forall m \in M_{sat}(t) \tag{4c}$$

$$\sum_{j \in V} f_{ij}^k(t) - \sum_{j \in V} f_{ji}^k(t) \begin{cases} = r_k(t) & \text{if } i = s_k \\ = 0 & \text{if } i \in V/\{s_k, d_k\} \\ = -r_k(t) & \text{if } i = d_k \end{cases} \qquad \forall k \tag{4d}$$

$$f_{ij}^k(t) = 0, \text{ if } \{i,j\} \notin path_k(t) \qquad \forall k \tag{4e}$$

$$f_{ij}^k(t) \le maxBdwh, \text{ if } \{i,j\} \in path_k(t) \qquad \forall k \tag{4f}$$

$$f_{ij}^k(t) \ge minBdwh, \text{ if } \{i,j\} \in path_k(t) \qquad \forall k \tag{4g}$$

$$\sum_{k \in m} r_k(t) \le v_m \qquad \forall m \in M \tag{4h}$$

$$f_{ij}^k(t) \ge 0 \qquad \forall (i,j) \in E, \text{ and } k \tag{4i}$$

$$z(t) \ge 0 \tag{4j}$$

Then we present the OMFRA algorithm as shown in Algorithm 1. The basic idea of OMFRA is to iteratively maximize the minimum data transfer satisfaction rate until all the requests are saturated, i.e., achieving maximum $z(t)$. In each iteration, newly saturated requests are removed from the subsequent process by fixing their corresponding rate value. In particular, it first solves the **MMF** problem (Line 6). Then each unsaturated transfer request m goes through the saturation test (Line 8-15). In this test, OMFRA first checks for residual paths in the network for request m (Line 8). If there is no such path, a further test is performed by solviing **MMF** with a new setting, in which $M_{unsat}(t)$ only contains request m, and all other request which were $M_{unsat}(t)$ are moved to $M_{sat}(t)$ with a satisfaction rate $z(t)$ (Line 9). If the new $z^{temp}(t)$ is the same as $z(t)$, it means that the request m is indeed saturated. We then update both $M_{sat}(t)$ and $M_{unsat}(t)$ (Line 10-14). After all data transfers are saturated in time slot t, we can derive a feasible set of $f_{ij}^k(t)$ for each data flow k in every transfer request m, and the corresponding $r_k(t)$ (Line 22-23). If the total satisfaction rate of a transfer request m is 1, it means that this request has been fulfilled, and we can remove it from the unfinished set of requests (Line 24-26), which will be pushed for next time slots. In this way, OMFRA achieves a balance between fairness of data transfers and network utilization. Furthermore, it has a low computational complexity.

[1] Whether Constraint (**??**) should use $\ge$ or $=$ requires a bit more thinking.

Note: Here we allocate not only bandwidth, i.e., scheduling, but also routes to different transfer requests. However, OMFRA can certainly handle the case where routes are fixed ahead by having the routes included in the formulation of **MMF** problem via a set of linear constraints.

Algorithm 1 OMFRA: an Optimal Max-Min Resource Allocation Algorithm for the **RAP** problem

```
1: $t \leftarrow 0$
2: while $\mathbb{M}_p \neq \emptyset$ do
3:     $t \leftarrow t + 1$
4:     $M_{sat}(t) \leftarrow \emptyset, M_{unsat}(t) \leftarrow \mathbb{M}_p$
5:     while $M_{unsat} \neq \emptyset$ do
6:         Solve $\mathbf{MMF}(t)$ and get the optimal value $z(t)$
7:         for each $m \in M_{unsat}(t)$ do
8:             if There is no residual path for transfer request $m$ to accommodate more flows then
9:                 Solve $\mathbf{MMF}(t)$ with revised inputs $M_{unsat}^{temp}(t) \leftarrow m$, $M_{sat}^{temp}(t) \leftarrow M_{sat}(t) \cup M_{unsat}(t)/\{m\}$
                   and $z_{sat}^{m}(t) \leftarrow z(t)$ for any $m \in M_{unsat}(t)/\{m\}$, and get the optimal value $z^{temp}(t)$
10:                if $z^{temp}(t) == z(t)$ then
11:                    $M_{sat}(t) \leftarrow M_{sat}(t) \cup \{m\}$
12:                    $M_{unsat}(t) \leftarrow M_{unsat}(t)/\{m\}$
13:                    $z_{sat}^{m}(t) \leftarrow z(t)$
14:                end if
15:            end if
16:        end for
17:    end while
18:    for each $m \in M_{sat}(t)$ do
19:        if $\sum_{t_0=1}^{t} z_{sat}^{m}(t_0) \geq 1$ then
20:            $z_{sat}^{m}(t) \leftarrow 1 - \sum_{t0=1}^{t-1} z_{sat}^{m}(t0)$
21:        end if
22:        Derive a feasible set of $f_{ij}^{k}(t)$ and the corresponding $r_k(t)$ for each flow $k$ in data transfer request
           $m \in \mathbb{M}_p$ based on constraints (4a), (4b) (4c) (4d) and (4i).
23:        if $\sum_{t_0=1}^{t} z_{sat}^{m}(t_0) == 1$ then
24:            $\mathbb{M}_p \leftarrow \mathbb{M}_p/\{m\}$
25:        end if
26:    end for
27: end while
28: return $T \leftarrow t$
```

1.3 Extension to multi-priority scenario

We can then extend the OMFRA algorithm to the scenario where data transfer requests have multiple prioities, and develop the OMFRA-MP algorithm as shown in Algorithm 2. OMFRA-MP uses the same technique as OMFRA does. The only difference is that when we allocate network resources to transfer requests with higher priority first. In this way, we get an simple yet efficient solution for this more general scenario.

2 Next Step 1 - An Online Algorithm

Currently we only focuses on the offline model for the demo, in which we has all the information we need in prior. Our next step will focus on a more general online settings, in which system information such as

Algorithm 2 OMFRA-MP: an Optimal Max-Min Resource Allocation Algorithm for the **RAP** problem with multiple prioirties

```
1: for $p \leftarrow P, P-1, \ldots, 1$ do
2:     $t \leftarrow 0$
3:     while $\mathbb{M}_p \neq \emptyset$ do
4:         $t \leftarrow t+1$
5:         $M_{sat}(t) \leftarrow \emptyset, M_{unsat}(t) \leftarrow \mathbb{M}_p$
6:         while $M_{unsat} \neq \emptyset$ do
7:             solve $\mathbf{MMF}(t)$ and get the optimal value $z(t)$
8:             for each $m \in M_{unsat}(t)$ do
9:                 if there is no residual path for transfer request $m$ to accommodate more flows then
10:                    solve $\mathbf{MMF}(t)$ with revised inputs $M_{unsat}^{temp}(t) \leftarrow m$, $M_{sat}^{temp}(t) \leftarrow M_{sat}(t) \cup M_{unsat}(t)/\{m\}$
                       and $z_{sat}^{m}(t) \leftarrow z(t)$ for any $m \in M_{unsat}(t)/\{m\}$, and get the optimal value $z^{temp}(t)$
11:                    if $z^{temp}(t) == z(t)$ then
12:                        $M_{sat}(t) \leftarrow M_{sat}(t) \cup \{m\}$
13:                        $M_{unsat}(t) \leftarrow M_{unsat}(t)/\{m\}$
14:                        $z_{sat}^{m}(t) \leftarrow z(t)$
15:                    end if
16:                end if
17:            end for
18:        end while
19:        for each $m \in M_{sat}(t)$ do
20:            if $\sum_{t_0=1}^{t} z_{sat}^{m}(t_0) \geq 1$ then
21:                $z_{sat}^{m}(t) \leftarrow 1 - \sum_{t0=1}^{t-1} z_{sat}^{m}(t0)$
22:            end if
23:            Derive a feasible set of $f_{ij}^{k}(t)$ and the corresponding $r_k(t)$ for each flow $k$ in data transfer request
               $m \in \mathbb{M}_p$ based on constraints (4a), (4b) (4c) (4d) and (4i).
24:            if $\sum_{t_0=1}^{t} z_{sat}^{m}(t_0) == 1$ then
25:                $\mathbb{M}_p \leftarrow \mathbb{M}_p/\{m\}$
26:            end if
27:        end for
28:    end while
29:    $t_p \leftarrow t$
30: end for
31: return $T \leftarrow \max\{t_p\}$
```

data transfer requests and network topology and resource information is not known ahead. With an online algorithm developed, we will be able to further improve the performance of the CERN system. One direct solution is to execute OMFRA-MP in an online fashion, i.e., only executing flow assignment (Line 3-29) for the current time slot. This would yield the same performance as the offline algorithm because we only care about achieve a max-min fair resource allocation at every time slot. The allocation decision made in time slot t_1 is independent from the allocation decision in time slot t_2.

3 Next Step 2 - The case with no file slicing

3.1 Route is fixed

Possible Solution:

1. Min-Hopcount
2. The path with lowest utilization rate in the past δ time slots.

3.2 Route is unspecified

At time slot t, order all requests by their arrival time. For each request, compute the possible saturate rate for each flow if added to the current set $\mathbb{M}$. Select the one with the lowest saturate rate and move on to next request.

4 Related Work

The large data transfer problem in CERN is highly similar to the inter-datacenter bulk data transfer problem. Our solution is motivated by existing work [1, 2].

References

[1] X. Lu, F. Kong, X. Liu, J. Yin, Q. Xiang, and H. Yu. Bulk savings for bulk transfers: Minimizing energy cost on inter-data-center traffic. Under review, 2015.

[2] Y. Wang, S. Su, S. Jiang, Z. Zhang, and K. Shuang. Optimal routing and bandwidth allocation for multiple inter-datacenter bulk data transfers. In *Communications (ICC), 2012 IEEE International Conference on*, pages 5538–5542, 2012.