Distributed Multipath Routing Algorithm for Data Center Networks

Eun-Sung Jung, Venkatram Vishwanath, Rajkumar Kettimuthu, Mathematics and Computer Science Division
Argonne National Laboratory
Email: {esjung,venkatv,kettimut}@mcs.anl.gov

***Abstract*—Multipath routing has been studied in diverse contexts such as wide-area networks and wireless networks in order to minimize the finish time of data transfer or the latency of message sending. The fast adoption of cloud computing for various applications including high-performance computing applications has drawn more attention to efficient network utilization through adaptive or multipath routing methods. However, the previous studies have not exploited multiple paths in an optimized way while scaling well with a large number of hosts for some reasons such as high time complexity of algorithms.**

In this paper, we propose a scalable distributed flow scheduling algorithm that can exploit multiple paths in data center networks. We develop our algorithm based on linear programming and evaluate the algorithm in FatTree network topologies, one of several advanced data center network topologies. The results show that the distributed algorithm performs much better than the centralized algorithm in terms of running time and is comparable to the centralized algorithm within 10% increased finish time in terms of data transfer time.

I. Introduction

As data from experimental or observational facilities in scientific computing are growing [1], data-intensive computing is drawing more attention. Even in government and IT sectors, more data-intensive applications are emerging in response to increased needs for data analysis.

Many enabling technologies have been developed to run data-intensive applications in big facilities such as data centers. Data movement is one of the essential components to be improved for data-intensive computing since I/O is usually considered to be slower than computation. The data movement beyond physical machines happens in two cases: (1) data transfers over wide-area networks for distributed computing across multiple sites and (2) data transfers over interconnection networks for large-scale/high-performance computing within a single site.

In particular, the data center networks and associated data flow scheduling play an important role in large-scale data-intensive computing. Various network topologies have been proposed such that quality of services for data movements such as latency and throughput is satisfied while scaling well with large-scale applications running up to thousands of nodes. The advanced network topologies proposed recently include FatTree and Dragonfly topologies. A large amount of research also has been conducted on data flow scheduling algorithms. Particularly in the context of data center networks, recent studies [2], [3] show that exploitation of diverse paths between a sender and a receiver in an intelligent way improves the performance of data movements. However, the previous studies have not fully addressed data movement via multiple paths since they utilize multiple paths among nodes by choosing the best single path per data flow adaptively. In addition, most data flow scheduling algorithms implementing multiple paths per data flow are limited in scalability; hence, such algorithms are not useful for a large-scale network in practice.

In this paper, we focus on data movements using multiple paths over data center networks where data paths can be explicitly established by the system administrator (e.g., openflow-based networks). More specifically, our contributions include development of distributed multipath routing algorithms.

The rest of the paper is structured as follows. In Section II, we present general knowledge and current issues regarding data center networks and data-intensive applications. In Section III, we describe the problem statement and the mathematical formulation for centralized data flow scheduling. In Section IV, we present the distributed data flow scheduling algorithm derived from the centralized one. In Section V, we present experimental results evaluating our proposed algorithm, and in Section VI, we describe related work in detail. In Section VII, we summarize our work and conclude with future work.

II. Background

We describe the state of the art in data center networks and challenges of deploying data-intensive applications in data centers.

A. Data Center Networks

Data center networks should be able to guarantee high throughput and resiliency. For such reasons, typical data center networks (e.g., FatTree) [4] are evolving into high-radix networks (e.g., Dragonfly) [5]. In this paper, we consider a k-ary FatTree topology as in Figure 1 where k port switches are used in the three-layer architecture. The important features of the k-ary FatTree are that it supports $k^3/4$ hosts and there are $(\frac{k}{2})^2$ paths available between hosts in different pods. Figure 2 shows an example of the Dragonfly topology with 72 hosts. The Dragonfly topology uses a group of subnetworks as a virtual high-radix router to build high-performance networks with very low global diameter.

However, multiple paths have not been used appropriately to maximize the utilization of data center networks. Hedera

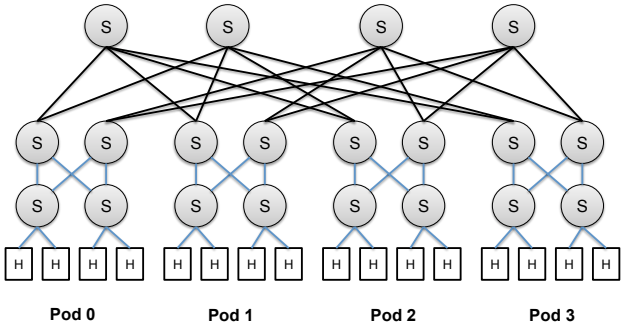

Fig. 1: FatTree topology when k=4 [4]: Circles represent switches and boxes represent hosts.

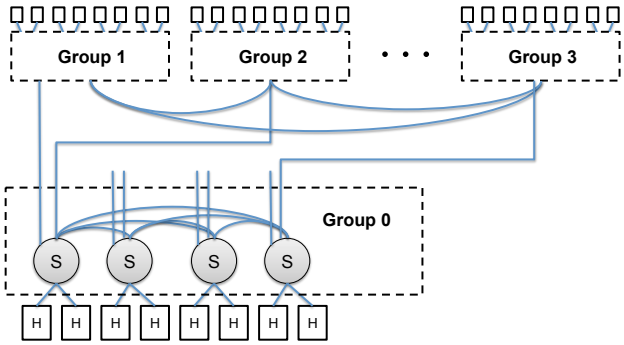

Fig. 2: Example of Dragonfly topology with 72 hosts.

[2] tried to improve the network utilization by mitigating the network bottleneck caused by multiple data transfers through rerouting some of them. Some researchers proposed fault-tolerant routing algorithms using path redundancy in data center networks [6], [7]. Even though multipath algorithms for multiple data transfers have been proposed in the context of wide-area networks, few of them has been applied to interconnection networks because of their high time complexities. At the protocol level, MPTCP has been deployed on interconnect networks and shows some promising results [8].

B. Data-Intensive Applications

Many applications including MapReduce and scientific computing for big data can be considered as data-intensive applications. Examples of data-intensive scientific computing are high energy physics and climate science [1].

One of the requirements of data-intensive applications for data center platforms is fast data movement because the number of data transfers in data-intensive applications is larger than in compute-intensive or commodity applications and often leads to a network bottleneck in overall performance [9], [10]. In order to mitigate network bottleneck in data center platforms, data center networks should be able to provide enough bandwidth among hosts (i.e., bisection bandwidth of a data center network,) and the data flow management system should be able to achieve high data transfer throughput by efficient routing. For instance, if two data flows are allocated two single paths sharing a low bandwidth link, the network utilization would be much lower than two data flows with multiple disjoint paths.

III. Problem Formulation

In this section, we present system models for our problem formulation and an LP-based formulation for centralized data flow scheduling.

A. System Model

Most of our notations are adopted from [11], and we will use the terms data transfer and job interchangeably in this paper. The interconnection network is represented by $\mathcal{G} = (\mathbf{V}, \mathbf{E}, \mathbf{TB})$, where V is a set of nodes (switches), E is a set of edges representing connectivity among switches, and TB is a set of time-bandwidth lists representing bandwidth availability over time. For example, TB_i is associated with ith edge, $e_i \in \mathbf{E}$. $TB_i(t)$ is the available bandwidth at time slice t.

The bulk data transfer requests are given as a set $\mathbf{R} = \{r | r = (s_i, d_i, D_i, S_i), 1 \leq i \leq n\}$, where s_i and d_i are a source and a destination of data transfer, D_i is the size of data, and S_i is the time when the data become available for transfer.

These models are for both on-demand and in-advance job scheduling. In general, the in-advance job scheduling problem is more complex than the on-demand job scheduling problem because the time-varying bandwidth on network links and future job requests should be considered additionally for the in-advance job scheduling problem. In this paper, to make the problem simple, we focus only on the on-demand job scheduling problem where TB is a list of only one entry and S_i becomes 0, representing the current time.

B. Multipath Routing Problem

In this paper, we define the multipath routing problem as follows: Given a network $\mathcal{G} = (\mathbf{V}, \mathbf{E}, \mathbf{TB})$ and a set of job requests $\mathbf{R} = \{r | r = (s_i, d_i, D_i, S_i = 0), 1 \leq i \leq n\}$, find the multipath routing to minimize the finish time until all the job requests are completed.

Depending on how the controllers manage the network paths for the job requests, we can classify the problem further into two categories: the centralized multipath routing problem (CMRP) and the distributed multipath routing problem (DMRP). The CMRP is to find the centralized multipath routing algorithm that is run at a single management node while the DMRP is to find the distributed multipath routing algorithm that is run at multiple management nodes to reduce the running time of the algorithm or to determine routing paths based on locally available information.

With any routing algorithm, we may run the algorithm in a different triggering period. For example, if the triggering period is 1 second, we can run a routing algorithm for job requests that arrived during the past 1 second. Therefore, the triggering period affects the number of job requests that the algorithm should cope with and affects the response time of the job requests. Another factor, which has an influence on the running time of a routing algorithm when deployed in a real system, is whether the algorithm considers previous job requests that are already in progress with allocated paths.

For example, if 10 job requests were being handled by the previous triggering period and are being serviced, we can either incrementally schedule newly arrived job requests or schedule all the job requests again. Obviously, the incremental policy has reduced running time at the price of a longer finish time of jobs because of fewer optimized routing results.

C. LP-Based Formulations for CMRP

In general, the scheduling problems of data transfers can be categorized into two classes from the perspective of data transfer start time: (1) the in-advance scheduling problem and (2) the on-demand scheduling problem. In this section, we derive LP formulations for the on-demand scheduling problem from a previous LP formulation for the in-advance scheduling problem in [11]. Figure 3 presents the LP formulation for in-advance multipath routing with the objective of minimizing the finish time of all data transfers. A network flow problem can be formulated as an edge or path formulation that puts constraints on edges or paths, respectively. In that regard, the LP formulation in Figure 3 belongs to the edge formulations.

In short, we can describe the formulation as follows. The notation used for formulations in the paper is summarized in Table I for ease of reference. Equation 1 and 2 are flow conservation constraints that ensure that the total amount of incoming flows is same as the total amount of outgoing flows if nodes are not a source or a destination of a job; otherwise the total amount of incoming or outgoing flows is equal to the demand of a job, and Equations 3 and 4 are link capacity constraints that ensure that the total amount of all flows on a link should not exceed the capacity of the link. Readers interested in the detailed derivation of the formulation can refer to [11].

$$minimize\ T_f$$

$$s.t.$$

$$\sum_{k:(l,k)\in E} f_{lk}^j(t) - \sum_{k:(k,l)\in E} f_{kl}^j(t) = 0, \quad \forall j \in J, \forall l \in V, l \neq s_j, l \neq d_j, t = 0...q \tag{1}$$

$$\sum_{t=0}^{q}\Big(\sum_{k:(l,k)\in E} f_{lk}^j(t) - \sum_{k:(k,l)\in E} f_{kl}^j(t)\Big) = \begin{cases} D_j & \text{if } l = s_j \\ -D_j & \text{if } l = d_j \end{cases}, \forall j \in \mathbf{J} \tag{2}$$

$$\sum_{j=1}^{p} f_{lk}^j(t) \leq b_{lk}(t) \times (T_{t+1} - T_t), \forall (l,k) \in E, t = 0...(q-1) \tag{3}$$

$$\sum_{j=1}^{p} f_{lk}^j(t) \leq b_{lk}(t) \times (T_f - T_t), \forall (l,k) \in E, t = q \tag{4}$$

$$f_{lk}^j \geq 0, \forall j \in \mathbf{J} \tag{5}$$

$$T_f \geq 0 \tag{6}$$

Fig. 3: LP-based edge formulation for in-advance multipath routing to minimize the finish time of all data transfer.

Accordingly, we can easily get the formulation for on-demand multipath routing with the objective of minimizing the finish time of all data transfers as presented in Figure 4. In the case of on-demand multipath routing, Equations 7 – 9 are much simpler than Equation 1 – 4 since the formulation doesn't need to consider multiple time slices. More specifically, the solution for in-advance multipath routing (IAMR) can be obtained through an iterative binary search on the finish time T_f. In contrast, the derived LP formulation for on-demand multipath routing (ODMR) can be solved at one time, which leads to a much-reduced running time of the algorithm. Even though the formulation in Figure 4 is not LP, we can easily transform it into an LP formulation by substutiting Z for $1/T_f$ and $maximize\ Z$ for $minimize\ T_f$. Figure 5 shows the matrix expression corresponding to the edge formulation for ODMR in Figure 4.

$$minimize\ T_f$$

$$s.t.$$

$$\sum_{k:(l,k)\in E} f_{lk}^j - \sum_{k:(k,l)\in E} f_{kl}^j = 0, \forall j \in J, \forall l \in V, l \neq s_j, l \neq d_j \tag{7}$$

$$T_f \times \Big(\sum_{k:(l,k)\in E} f_{lk}^j - \sum_{k:(k,l)\in E} f_{kl}^j\Big) = \begin{cases} D_j & \text{if } l = s_j \\ -D_j & \text{if } l = d_j \end{cases}, \forall j \in \mathbf{J} \tag{8}$$

$$\sum_{j=1}^{p} f_{lk}^j \leq b_{lk}, \forall (l,k) \in E \tag{9}$$

Fig. 4: Edge formulation for on-demand multipath routing (ODMR) to minimize the finish time of all data transfer.

$$minimize\ -Z \tag{10}$$

$$s.t.$$

$$AF^j = ZD^j, \forall j \in \mathbf{J} \tag{11}$$

$$\sum_{j=1}^{p} F^j \leq B \tag{12}$$

$$Z \geq 0 \tag{13}$$

$$F^j \geq 0 \tag{14}$$

Fig. 5: Matrix-form LP formulation for on-demand multipath routing.

IV. Distributed Algorithms

In this section, we present the development of distributed multipath routing algorithms using the Lagrangian method. We also discuss the deployment of the distributed algorithms on real data center networks.

A. Problem Decomposition

In this paper, we use the Lagrangian method [12] to decompose the multipath routing problems into multiple subproblems

such that path computations happen in a distributed manner and the centralized server is not a bottleneck anymore. The general steps to apply the Lagrangian method to the development of large-scale distributed algorithms are as follows.

1) Step 1: Formulate an original problem as a nonlinear/linear programming.
2) Step 2: Find a Lagrangian function for the problem.
3) Step 3 Find a dual Lagrangian function in accordance with the Lagrangian function.
4) Step 4: Decompose the problem based on multiple terms in the dual Lagrangian function.

In Section III, we already formulated the problem, which belongs to step 1. In the following sections, we present detailed formulations and procedures regarding step 2 through step 4.

1) Lagrangian Function: The following formulations are Lagrangian functions for the previous LP formulations. For simplicity of expression, we use the matrix form as in Figure 6.

$$
\begin{aligned}
&L(Z, F, \lambda, \nu) \\
&= -Z + \lambda^T (\sum_{j=1}^{p} F^j - B) + \sum_{j=1}^{p} \nu_j^T (AF^j - ZD^j) \\
&= \sum_{j=1}^{p} (\lambda^T + \nu_j^T A) F^j - (1 + \sum_{j=1}^{p} \nu_j^T D^j) Z - \lambda^T B \qquad (15) \\
&\lambda \geq 0 \qquad (16) \\
&\lambda \in \mathbb{R}^m : \text{ Lagrangian multiplier for inequalities} \\
&\nu \in \mathbb{R}^n : \text{ Lagrangian multiplier for equalities}
\end{aligned}
$$

Fig. 6: Matrix-form Lagrangian function for edge-form LP for on-demand multipath routing.

In Fig. 6, λ and ν variables are called *Lagrangian multipliers*, which are variables in the dual Lagrangian function.

2) Dual Lagrangian Function and Problem: The dual Lagrangian function can be defined as follows.

$$
\begin{aligned}
&g(\lambda, \nu) \\
&= \inf_{F,Z} L(Z, F, \lambda, \nu) \\
&= \inf_{F,Z} \left(\sum_{j=1}^{p} (\lambda^T + \nu_j^T A) F^j - (1 + \sum_{j=1}^{p} \nu_j^T D^j) Z - \lambda^T B \right) \qquad (17)
\end{aligned}
$$

Since F and Z are affine functions, we can further analyze Equation 17 and get Equation 18 with constraints 19 and 20.

$$
\begin{aligned}
&g(\lambda, \nu) = -\lambda^T B \qquad (18) \\
&s.t. \\
&\lambda^T + \nu_j^T A >= 0, \forall j \in \mathbf{J} \qquad (19) \\
&1 + \sum_{j=1}^{p} \nu_j^T D^j = 0 \qquad (20)
\end{aligned}
$$

We then obtain the dual Lagrangian problem as in Fig. 7 corresponding to the primal problem in Fig. 5.

$$
\begin{aligned}
&maximize\ g(\lambda, \nu) \qquad (21) \\
&s.t. \\
&\lambda \geq 0 \qquad (22) \\
&\lambda^T + \nu_j^T A >= 0, \forall j \in \mathbf{J} \qquad (23) \\
&1 + \sum_{j=1}^{p} \nu_j^T D^j = 0 \qquad (24)
\end{aligned}
$$

Fig. 7: Dual Lagrangian problem for on-demand multipath routing.

3) Decomposed Problems and Algorithms: In this subsection, we describe the decomposition of the dual Lagrangian problem in Fig. 7. Let λ_i be a Lagrangian multiplier regarding pod i. We can assign index $(k+1)$ for the rest of nodes and links that do not belong to pods. Then $\lambda_i \in \mathbf{R^{m^i}}$, where m^i is the number of links in pod i and $\lambda = [\lambda_1^T ... \lambda_{k+1}^T]^T$. Similarly, we can let ν_{ji} be a Lagrangian multiplier regarding job j and pod i. Then $\nu_{ji} \in \mathbf{R^{n^i}}$, where n^i is the number of nodes in pod i and $\nu_j = [\nu_{j1}^T ... \nu_{jk+1}^T]^T$.

Accordingly we can partition A, B, and D into A_i, B_i, and D_i, where $i = 1...(k+1)$. Therefore, we can rewrite the formulation as in Fig. 7 into the decomposed formulation as in Fig. 8.

$$
\begin{aligned}
&minimize\ \ \phi = \sum_{i=1}^{k+1} \phi_i, \text{where}\ \ \phi_i = \lambda_i B_i \qquad (25) \\
&s.t. \\
&\lambda_i \geq 0, i = 1...(k+1) \qquad (26) \\
&\lambda_i^T + \nu_{ji}^T A_i >= 0, \forall j \in \mathbf{J}, i = 1...(k+1) \qquad (27) \\
&y_i + \sum_{j=1}^{p} \nu_{ji}^T D_i^j = 0, i = 1...(k+1) \qquad (28) \\
&\sum_{i=1}^{k+1} y_i = 1 \qquad (29)
\end{aligned}
$$

Fig. 8: Decomposed dual Lagrangian problem for on-demand multipath routing.

When the distributed algorithm is deployed in real systems, we assume that there are a centralized job request handler and multiple schedule computation elements. When a centralized job request handler is triggered periodically, it sends the job information to the multiple schedule computation elements, and each schedule computation element sends the result of a decomposed dual problem back to the centralized job request handler. The centralized job request handler then establishes the paths for the job requests. This procedure is summarized in Algorithm 1.

V. Experimental Evaluation

We evaluate our algorithms through extensive simulations. The simulations have been conducted by using synthetic

Category	Symbol	Description
Regular	T_f	Finish time of all file transfers (jobs).
	Z	$1/T_f$.
	T_i	The start time of ith time slice.
	D_j	The demand of job j.
	$\mathbf{J}$	A set of job requests.
	$\mathbf{J}_g$	A set of job requests belonging to group g.
	n	The number of nodes ($= \|\mathbf{V}\|$).
	m	The number of edges ($= \|\mathbf{E}\|$).
	p	The number of jobs ($= \|\mathbf{J}\|$).
	q	The number of time slices to be considered.
	$F^j_{lk}(t)$	Flow of job j on edge $(l,k) \in E$ in time slice t.
	$b_{lk}(t)$	Available bandwidth on edge $(l,k) \in E$ in time slice t.
Vector/Matrix	A	Incidence matrix for a network, $A \in \mathbb{R}^{n \times m}$, $A_{ij} = \begin{cases} 1 & \text{if arc j enters node i.} \\ -1 & \text{if arc j leaves node i.} \\ 0 & \text{otherwise.} \end{cases}$
	F^j	Flow vector for job j, $F^j \in \mathbb{R}^m$.
	D^j	Demand vector for job j, $D^j \in \mathbb{R}^n$.
	B	Bandwidth constraint matrix for a network, $B \in \mathbb{R}^m$.

TABLE I: Notations for formulations

Algorithm 1 Distributed algorithms based on decomposition of Lagrangian dual problem

Input: Job requests and local network status.

· The centralized job request handler sends job requests to local schedule computation elements.

· y_i is set to $1/(k+1)$.

· Each schedule computation element sends the result of a decomposed dual problem back to the centralized job request handler and the centralized job request handler update y in proportion to the objective value. This step is iteratively executed until the values of y are stabilized.

· The centralized job request handler converts dual solutions into primal solutions using LP complementary slackness property.

· The centralized job request handler establishes the paths for the job requests.

network topologies and job requests for data flows. The experimental results are evaluated to show how well the distributed algorithm performs compared with the centralized algorithms and whether those distributed algorithms are feasible in real platforms.

A. Algorithm Performance Evaluation

1) Simulation configuration: We use synthetic network topologies for FatTree [4] by varying k from 2 up to 16 by multiplying by 2, which corresponds to the number of hosts ranging from 2 to 1,024. The bandwidth of all network links is set to 10 Gbps (∼1,280 MB/s). We also synthetically generate random data flows in a similar way as in [4] such that the destinations of data flows are randomly selected, the number of flow per host (FPH) is increased from 1 to 5 to simulate different network traffic loads in data center networks, and the length of data flows is pareto distributed where $\alpha = 1$ and the value is multiplied by 1024.

We use AMPL [13] to implement our LP-based algorithms and snopt [14], which is developed by Stanford University, as an LP solver. Fig. 9 shows our experimental configuration. The host on which simulations are run has 16 AMD Opteron(tm) processors with 2 GHz, but only a single CPU is used because of the limitation of the LP solver, snopt.

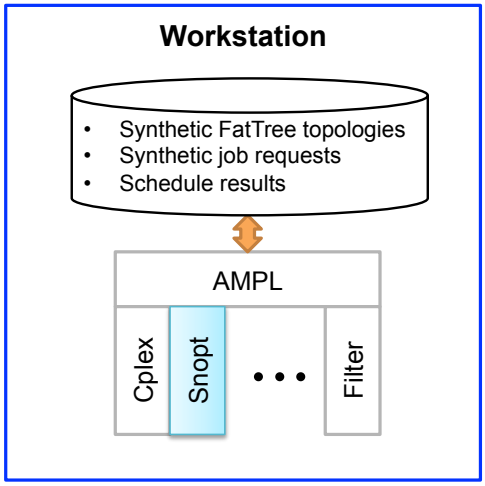

Fig. 9: Test environment.

2) Results and evaluation: Fig. 10 shows that the running time of the centralized algorithm as the network size grows from $k = 2$(# hosts=2) to $k = 16$(# hosts=1024) when FPH is 1. The running time of the centralized algorithm increases dramatically from a few seconds when $k = 2, 4, 8$ to over an hour when $k = 16$.

Fig. 11 shows that the running time of the centralized algorithm as the number of jobs grows. In contrast to the results of running time vs. network size, the running time almost linearly grows as the number of jobs grows.

Fig. 12 shows that the running time of the distributed algorithm as the network size grows from $k = 2$(# hosts=2) to $k = 16$(# hosts=1024) when FPH is 1. The distributed algorithm could achieve the running time of a few minutes even for $k = 16$ and the finish time of jobs is 10% more than the optimal solution of the centralized algorithm.

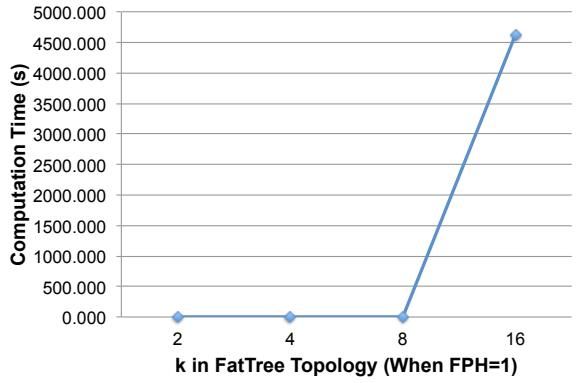

Fig. 10: Running time of the centralized algorithm as the network size grows.

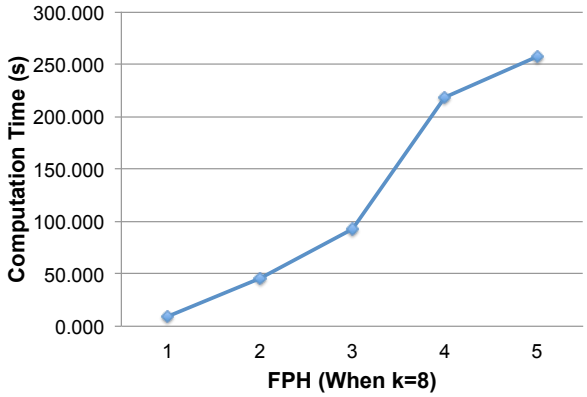

Fig. 11: Running time of the centralized algorithm as the number of jobs grows.

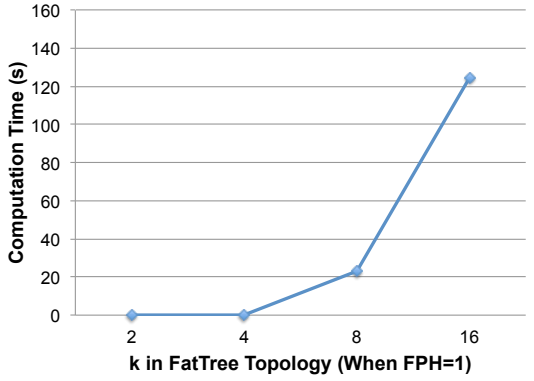

Fig. 12: Running time of the distributed algorithm as the network size grows.

VI. Related Work

We describe related work in the diverse contexts including multipath and distributed routing algorithms. Many prior studies have been done regarding multipath data transfer. To capture the whole picture of those studies, we classify them with regard to constraints on a data transfer request, path properties, and the goal of an algorithm. Table II summarizes multipath routing algorithms for various multipath routing problems based on those three criteria.

In the following, we present the details of the related work and shortly highlight our contribution in comparison with those works.

A user request for data transfer may comprise multiple tuples with different constraints as follows.

- Deadline: If a deadline is given per request, the multipath schedule for requests should meet the deadlines or at least minimize the extent that the data transfer time passes the deadline.
- Latency/delay: In some cases such as interactive data manipulation, latency or delay is an important factor to consider. The constraints on latency/delay are usually closely related to delays on network links. These constraints may be represented by *path length* or *hop number*, which has close a relationship with latency and delay.
- In-advance: In contrast to on-demand data transfer requests, in-advance requests require network path reservations in the future, which often lead to time-varying network resource management and sophisticated routing algorithms for optimized resource utilization.

With these constraints on requests, routing algorithms output routing paths for data transfers. Depending on the circumstances of a system, output paths have different properties as follows.

- Dynamic: A dynamic path can vary over time. For instance, the network paths used for a certain request can be altered in the middle of data transfer. Dynamic paths are feasible in a system where the path-switching cost as well as the time complexity of a routing algorithm is reasonably low.
- Limited number: The number of routing paths for a request may need to be restricted for similar reasons to those discussed for dynamic paths. The smaller number of paths may make it easier to establish and compute paths. For example, if we explore a solution among the given feasible path set, routing algorithms usually take less time to compute optimal paths than when we explore all possible paths.

The following goals for routing algorithms are popular.

- Minimizing network congestion: This goal is to minimize the maximum ratio of used bandwidth and the link capacity for all network links. This is often achieved through algorithms adapted from the general multi-commodity flow problem [20].
- Maximizing network throughput: This goal is also to maximize network utilization. It is often achieved through algorithms adapted from the maximum concurrent flow problem (MCFP) [21].
- Maximizing fairness: This goal is to guarantee max-min fairness among flows; this is often achieved through algorithms adapted from the max-min fairness algorithm combined with the multi-commodity flow problem.
- Minimizing finish time: This goal is to finish all the data transfer requests as early as possible. This is often achieved through algorithms adapted from the maximum flow problem [20].

Constraint on a Request			Path Property			Controller	Requests	Representative	Goal
Deadline	Delay	In-Advance	Length	Number	Dynamic	Cntr/Dist	1/M	study	
√	√					Cntr	1	[15]	-
			√	√		Cntr	1	[16]	Minimizing network congestion
			√			Cntr	M	[17]	Minimizing network congestion
√		√		√	√	Cntr	M	[18]	Maximizing network throughput
				√		Cntr	M	[19]	Balancing fairness and throughput
		√		√	√	Cntr	M	[11]	Minimizing finish time

TABLE II: Multipath routing algorithms for bulk data (file) transfers

Our work is distinguished from the previous work because our proposed algorithms are distributed and scale well as the size of a network grows.

VII. Conclusions and Future Work

We developed a distributed multipath routing algorithm to minimize the finish time given the multiple jobs. We conducted experiments using synthetic FatTree networks and job requests, and the results show the distributed algorithm performs much better than the centralized algorithm in terms of running time and is comparable to the centralized algorithm within 10% increased finish time in terms of data transfer time. As future work, we plan to conduct dynamic network simulations where jobs arrive and schedulers are triggered as time passes. In this way, we will be able to measure more accurately the effects of distributed algorithms from the perspective of network utilization.

Acknowledgments

The submitted manuscript has been created by UChicago Argonne, LLC, Operator of Argonne National Laboratory ("Argonne"). Argonne, a U.S. Department of Energy Office of Science laboratory, is operated under Contract No. DE-AC02-06CH11357. The U.S. Government retains for itself, and others acting on its behalf, a paid-up nonexclusive, irrevocable worldwide license in said article to reproduce, prepare derivative works, distribute copies to the public, and perform publicly and display publicly, by or on behalf of the Government.

This material was based or was supported by the U.S. Department of Energy, Office of Science, Advanced Scientific Computing Research Program, under Contract DE-AC02-06CH11357.

References

[1] "Synergistic challenges in data-intensive science and exascale computing," http://science.energy.gov/ /media/40749FD92B58438594256267425C4AD1.ashx, Apr. 2014.

[2] M. Al-Fares, S. Radhakrishnan, B. Raghavan, N. Huang, and A. Vahdat, "Hedera: dynamic flow scheduling for data center networks," in *Proceedings of the 7th USENIX conference on Networked systems design and implementation*, ser. NSDI'10. Berkeley, CA, USA: USENIX Association, 2010, p. 1919, ACM ID: 1855730. [Online]. Available: http://portal.acm.org/citation.cfm?id=1855711.1855730

[3] B. Prisacari, G. Rodriguez, C. Minkenberg, and T. Hoefler, "Fast pattern-specific routing for fat tree networks," *ACM Trans. Archit. Code Optim.*, vol. 10, no. 4, p. 36:136:25, Dec. 2013. [Online]. Available: http://doi.acm.org/10.1145/2555289.2555293

[4] M. Al-Fares, A. Loukissas, and A. Vahdat, "A scalable, commodity data center network architecture," in *Proceedings of the ACM SIGCOMM 2008 Conference on Data Communication*. New York, NY, USA: ACM, 2008, p. 6374. [Online]. Available: http://doi.acm.org/10.1145/1402958.1402967

[5] J. Kim, W. J. Dally, S. Scott, and D. Abts, "Technology-driven, highly-scalable dragonfly topology," in *Proceedings of the 35th Annual International Symposium on Computer Architecture*, ser. ISCA '08. Washington, DC, USA: IEEE Computer Society, 2008, p. 7788. [Online]. Available: http://dx.doi.org/10.1109/ISCA.2008.19

[6] V. Liu, D. Halperin, A. Krishnamurthy, and T. Anderson, "F10: A fault-tolerant engineered network," in *the 10th USENIX Symposium on Networked Systems Design and Implementation (NSDI 13)*. Lombard, IL: USENIX, 2013, p. 399412. [Online]. Available: https://www.usenix.org/conference/nsdi13/technical-sessions/presentation/liu_vincent

[7] B. Yang, J. Liu, S. Shenker, J. Li, and K. Zheng, "Keep forwarding: Towards k-link failure resilient routing," in *2014 Proceedings IEEE INFOCOM*, Apr. 2014, pp. 1617–1625.

[8] C. Raiciu, S. Barre, C. Pluntke, A. Greenhalgh, D. Wischik, and M. Handley, "Improving datacenter performance and robustness with multipath TCP," in *Proceedings of the ACM SIGCOMM 2011 Conference*. New York, NY, USA: ACM, 2011, p. 266277. [Online]. Available: http://doi.acm.org/10.1145/2018436.2018467

[9] R. R. Expsito, G. L. Taboada, S. Ramos, J. Tourio, and R. Doallo, "Performance analysis of HPC applications in the cloud," *Future Generation Computer Systems*, vol. 29, no. 1, pp. 218–229, Jan. 2013. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0167739X12001458

[10] M. Chowdhury, S. Kandula, and I. Stoica, "Leveraging endpoint flexibility in data-intensive clusters," in *Proceedings of the ACM SIGCOMM 2013 Conference on SIGCOMM*. New York, NY, USA: ACM, 2013, p. 231242. [Online]. Available: http://doi.acm.org/10.1145/2486001.2486021

[11] Y. Li, S. Ranka, and S. Sahni, "In-advance path reservation for file transfers in e-science applications," *The Journal of Supercomputing*, vol. 59, no. 3, pp. 1167–1187, 2012. [Online]. Available: http://link.springer.com/article/10.1007/s11227-010-0509-9

[12] S. Nash and A. Sofer, *Linear and nonlinear programming*, ser. McGraw-Hill series in industrial engineering and management science. McGraw-Hill, 1996. [Online]. Available: http://books.google.com/books?id=MQAoAQAAMAAJ

[13] "AMPL - STREAMLINED MODELING FOR REAL OPTIMIZATION." [Online]. Available: http://ampl.com/

[14] "SNOPT." [Online]. Available: http://www.sbsi-sol-optimize.com/asp/sol_product_snopt.htm

[15] N. Rao and S. Batsell, "QoS routing via multiple paths using bandwidth reservation," in *Proceedings of INFOCOM '98. Seventeenth Annual Joint Conference of the IEEE Computer and Communications Societies*, vol. 1, 1998, pp. 11–18.

[16] R. Banner and A. Orda, "Multipath routing algorithms for congestion minimization," *IEEE/ACM Trans. Netw.*, vol. 15, no. 2, pp. 413–424, 2007. [Online]. Available: http://portal.acm.org/citation.cfm?id=1279660.1279673&coll=GUIDE&dl=GUIDE&CFID=18436580&CFTOKEN=35906744

[17] Y. Lee, Y. Seok, Y. Choi, and C. Kim, "A constrained multipath traffic engineering scheme for MPLS networks," in *IEEE International Conference on Communications*, vol. 4, 2002, pp. 2431–2436 vol.4.

[18] K. Rajah, S. Ranka, and Y. Xia, "Advance reservations and scheduling for bulk transfers in research networks," *IEEE Transactions on Parallel and Distributed Systems*, vol. 20, no. 11, pp. 1682–1697, 2009. [Online]. Available: 10.1109/TPDS.2008.250

[19] E. Danna, S. Mandal, and A. Singh, "A practical algorithm for balancing the max-min fairness and throughput objectives in traffic engineering," in *2012 Proceedings IEEE INFOCOM*, 2012, pp. 846–854.

[20] R. Ahuja, *Network flows: theory, algorithms, and applications*. Englewood Cliffs N.J.: Prentice Hall, 1993.
[21] F. Shahrokhi and D. W. Matula, "The maximum concurrent flow problem," *J. ACM*, vol. 37, no. 2, p. 318334, Apr. 1990. [Online]. Available: http://doi.acm.org/10.1145/77600.77620