An Efficient Method for Determining Full Point-to-Point Latency of Arbitrary Indirect HPC Networks

Chengchun Liu[1], Zhang Yang[2](✉), Limin Xiao[1](✉), Baicheng Yan[1], Zhihao Wang[1], and Hongyun Tian[2]

[1] School of Computer Science and Engineering, Beihang University, Beijing 100191, China
xiaolm@buaa.edu.cn

[2] Institute of Applied Physics and Computational Mathematics, No. 2 East Fenghao Road, Haidian District, Beijing 100094, China
yang_zhang@iapcm.ac.cn

Abstract. Point-to-point latency is one of the most important metrics for high performance computer networks and is used widely in communication performance modeling, link-failure detection, and application optimization. However, it is often hard to determine the full-scale point-to-point latency of large scale HPC networks since it often requires measurements to the square of the number of terminal nodes. In this paper, we propose an efficient method to generate measurement plans for arbitrary indirect HPC networks and reduces the measurement requirements from $O(n^2)$ to m, which is often $O(n)$ in modern indirect networks containing n nodes and m links, thus significantly reduces the latency measure overhead. Both analysis and experiments show that the proposed method can reduce the overhead of large-scale fat-tree networks by orders of magnitudes.

1 Introduction

Point-to-point latency is a fundamental metric of high performance computer networks, and is widely used in network performance modeling [1,2], communication performance optimization [3], and high performance computer maintenance. The first and formost step to make use of the latency is to measure the latency. A common method to get the latency is to measure the round-trip time (RTT) between any pair of nodes. While one measurement of RTT is quick enough, obtaining the full-network point-to-point latency can be extremly time-consuming since it involves $n(n-1)/2$ (or $O(n^2)$) measurements, where n is the number of terminal nodes. One may use parallel measurements to reduce the round of measurements, but parallel measurements can interfere with each other and reduce the accuracy of the results. Thus, it is essential to reduce the total

F. Zhang et al. (Eds.): NPC 2018, LNCS 11276, pp. 52–63, 2018.
https://doi.org/10.1007/978-3-030-05677-3_5

number of measurements, so as to make it possible to use these latency-based methods on modern super-computers with tens of thousands of computer nodes.

In this paper, we propose a minimal and parallel method for full-scale point-to-point latency measurements on super-computers with indirect networks (such as fat-tree, dragonfly and slimfly networks), abbreviated as *PMM*. Our method first construct a minimal set of node pairs between which the RTT is measured, given the network topology and the routing table, then compute a measurement plan to make use of the parallelism between the measurements with the gurantee that concurrent measurements will not interfere with one another. The minimal set of node pairs goes from $n(n-1)/2$ to m, where m is the number of links connecting the network interface and the routers, which is often proportional to the number of nodes, thus reduces the number of measurements from $O(n^2)$ to $O(n)$. The parallel measurement plan can further reduce the round of measurements, for example, by 33.3% in our experimental settings.

The reset of this paper is organized as follows. In Sect. 2, we introduce some related works on network latency measurement. In Sect. 3, we present our latency measurement method in detail. In Sect. 4, we prove the effectiveness of our methods by theoretical analysis and experiments. We also present performance analysis of the method itself. In Sect. 5, we discuss the possible applications of our proposed method. In the last section comes the conclusions.

2 Related Works

Communication latency or distance measurement are investigated in some literatures. Authors in [4] proposed a latency system based on GNP for fast obtaining latency information between arbitrary web client pairs distributed in wide area networks. This method has been used in the Google's content distribution network which helps to find the nearest data center for a web client. This method can estimate latency results quickly only with a small number of CDN modifications and decouples with web client, but is not suitable for the dense network such as HPC network or data center network. The literatures [5,6] also aim to obtain the latency in wide area network environment in different ways, but those methods are not suitable for dense networks.

Authors in [7] proposed a system called Pingmesh for latency measurement and analysis in large scale data center networks. The latency measurement system represents the network topology as three complete graphs, namely the server complete graph, the switch complete graph, and the data center complete graph. The method needs to select some representative node pairs and measure the latency information between those nodes. With these information, the method can approximately estimate the latency between different nodes in the same switch, in different switches, or in different data centers. But this method measures only partially the network and can not be used in full-network measurements.

The work [8] is the most similar to our work. They proposed a method to measure the communication distance between nodes on the Internet. This method also needs to construct the communication distance equations through a large number of measurements and then solve the least squares solution of the equations, which is considered as the distance. The main concern of the method is whether the calculation result of the communication distance is accurate without considering the time cost caused by the inappropriate measurement set. In contrast, our method carefully selects a minimal measurement set and then measures the latency between node pairs in the set in parallel to reduce the total time cost.

3 The PMM Method

3.1 Definitions

In order to simplify the introduction of our measurement method, we introduce some definitions, mathematical symbols and necessary assumptions in this section. Data transmission in the network is a complex process, which is affected by communication protocol, network topology, and hardware architecture. Since point-to-point latency on direct networks can be easy, we only focus on indirect networks in this paper. The data is transmitted from the source NIC, through the links, to routers, and direct to other routers, and finally to the destination NIC, as shown in Fig. 1. The NIC is connected to a computing node, which is called a terminal node. We also assume the network uses static routing instead of adaptive routing.

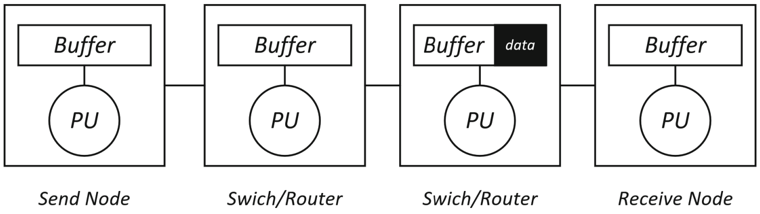

Fig. 1. Data transmission in indirect networks. The data is transmitted from the source terminal node to the destination through links and routers.

Definition 1. *a single link refers to a physical link between any adjacent devices in an indirect network. The latency of a single link refers to the time for a measuring packet to pass through the link from the buffer of the device at one end of the link to the buffer of the device at another end.*

Definition 2. *a measuring path refers to the entire path contained in the transmission of data between two communication nodes in an indirect network, which passes through some middle routing devices and physical links. The latency of the measuring path refers to the sum of latency of all single links in the path.*

Definition 3. *an aggregated link refers to a subpath of a measuring path which consists of one or more adjacent links. The method is not able to calculate the latency of any single link in an aggregated link, but is able to calculate the latency of the aggregated link.*

We provide some mathematical symbols to represent the elements in the method, as shown in Table 1.

Table 1. All mathematical symbols used in the method

Symbol	Description
k_x	Computing node
$P_{x,y}$	The measuring path from node x to node y
$P_{x,y}^{rtt}$	The round-trip measuring path between node x and node y
l_x	Single link
$a_{<x,y>,z}$	The times the single link z appears in the path $P_{x,y}^{rtt}$
$\alpha_{<x,y>}$	The vector form of a path whose elements are $a_{<x,y>,z}$
o_x	The latency of link x
O_x	The latency of path x
S	The set of path whose elements are $\alpha_{<x,y>}$
$S_x^{'}$	A maximal linearly independent subset of S

3.2 Method

Now we describe our latency measurement method in detail. Our method assumes that one can get the route of arbitrary node pairs. Through our paper, we use a simple network as shown in Fig. 2 for illustration. The network consists of 3 switches, 6 nodes and 8 single links. We can find many redundant measurements when we measure the latency between all node pairs. We take the 4 nodes connected by r_1 as an example. When measuring all pairs, we need to measure the latency of 6 paths, i.e., $P_{k_1,k_2}^{rtt}, P_{k_1,k_5}^{rtt}, P_{k_1,k_6}^{rtt}, P_{k_2,k_5}^{rtt}, P_{k_2,k_6}^{rtt}, P_{k_5,k_6}^{rtt}$. But if we just measure $P_{k_1,k_2}^{rtt}, P_{k_1,k_5}^{rtt}, P_{k_1,k_6}^{rtt}, P_{k_2,k_5}^{rtt}$ for latency, and make use of the fact link latency is additive, we can get Eq. 1.

$$\begin{cases} o_{l_1} + o_{l_2} = 1/2 \cdot O_{P_{k_1,k_2}^{rtt}} \\ o_{l_1} + o_{l_7} = 1/2 \cdot O_{P_{k_1,k_5}^{rtt}} \\ o_{l_1} + o_{l_8} = 1/2 \cdot O_{P_{k_1,k_6}^{rtt}} \\ o_{l_2} + o_{l_7} = 1/2 \cdot O_{P_{k_2,k_5}^{rtt}} \end{cases} \tag{1}$$

By solving Eq. 1, we can obtain $o_{l_1}, o_{l_2}, o_{l_7}, o_{l_8}$ and calculate $O_{P_{k_2,k_6}^{rtt}} = 2 \cdot (o_{l_2} + o_{l_8}), O_{P_{k_5,k_6}^{rtt}} = 2 \cdot (o_{l_7} + o_{l_8})$. Further more, there are redundant measurements between the nodes connected to different switches. Suppose we have measured the path latency between some nodes directly connected to the same switch. We need to measure $P^{rtt}_{k_1,k_3}, P^{rtt}_{k_1,k_4}, P^{rtt}_{k_2,k_3}, P^{rtt}_{k_2,k_4}, P^{rtt}_{k_5,k_3}, P^{rtt}_{k_5,k_4}, P^{rtt}_{k_6,k_3}, P^{rtt}_{k_6,k_4}$ for latency when measuring one by one. In fact, we can only measure $P^{rtt}_{k_1,k_3}$ to get $o_{l_1} + o_{l_3} + o_{l_4} + o_{l_5} = O_{P^{rtt}_{k_1,k_3}}$ and calculate $o_{l_3} + o_{l_4}$. In addition, we can measure node pairs which do not share any link in parallel. For example, we can measure the latency of $P^{rtt}_{k_1,k_2}$ and $P^{rtt}_{k_3,k_4}$ in parallel.

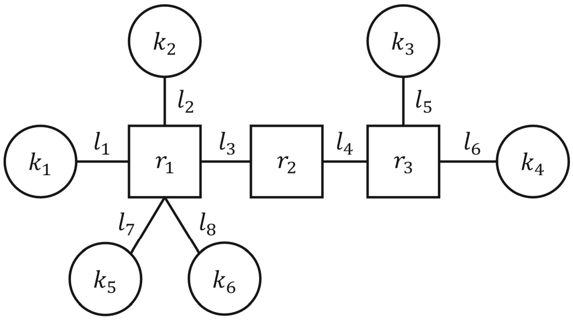

Fig. 2. A sample network with 6 nodes, 8 single links and 3 switches. Only 7 rather than 15 measurements are necessary for full-network point-to-point latency.

The example above illustrates the core idea of our method. By assuming the node-to-node latency is the addition of link latencies, we can select a number of node pairs which covers all links in the network and measure the node-to-node latencies, then recover the link latencies by solving a linear equation. The measurement can further be done in parallel. Although we only consider link latency here, our method applies to cases where both link and router latency are included, since they only add more variables and does not change the additive nature of latency.

Concretely, for a network containing n nodes and m links, the method includes the following steps.

a. Construct full measurement path set S, which contains all measuring paths.

By querying routing information, we can get the single link set of any path between node k_i and k_j. The lateny of path $P^{rtt}_{k_i,k_j}$ can be expressed as $Latency(P^{rtt}_{k_i,k_j}) = a_{<i,j>,1} \cdot o_{l_1} + a_{<i,j>,2} \cdot o_{l_2} + \cdots + a_{<i,j>,m} \cdot o_{l_m} = \alpha_{<i,j>} \cdot \beta$ where $\alpha_{<i,j>} = (a_{<i,j>,1}, a_{<i,j>,2}, \cdots, a_{<i,j>,m-1}, a_{<i,j>,m})$, $\beta = (o_{l_1}, o_{l_2}, \cdots, o_{l_{m-1}}, o_{l_m})$. The full measuring path set S=$\{\alpha_{<1,2>}$,$\alpha_{<1,3>}$,$\cdots$, $\alpha_{<n-2,n>}$, $\alpha_{<n-1,n>}\}$ which consists of $n(n-1)/2$ measuring paths. For the network shown in Fig. 2, $S = \{\alpha_{<k_1,k_2>}, \alpha_{<k_1,k_3>}, \alpha_{<k_1,k_4>}, \alpha_{<k_1,k_5>}, \alpha_{<k_1,k_6>}, \alpha_{<k_2,k_3>}, \alpha_{<k_2,k_4>}, \alpha_{<k_2,k_5>}, \alpha_{<k_2,k_6>}, \alpha_{<k_3,k_4>}, \alpha_{<k_3,k_5>}, \alpha_{<k_3,k_6>}, \alpha_{<k_4,k_5>}, \alpha_{<k_4,k_6>}, \alpha_{<k_5,k_6>}\}$. Taking $\alpha_{<k_1,k_2>}$ as an example. $\alpha_{<k_1,k_2>} = (2, 2, 0, 0, 0, 0, 0, 0)$ means that the measuring path $P^{rtt}_{k_1,k_2}$ consists of l_1, l_2, l_2, l_1.

b. Select the minimal measurement path set $S^{'}$, which is the subset after removing redundant measurement path in S.

By linear algebra theory, any element in S can be expressed as a linear combination of the maximal linearly independent subset of S. Thus, we choose the maximal linearly independent subset of S as the minimal measurement path set $S^{'}$, and name it as *MMSets*. The maximal number of elements in any *MMSet* is never greater than the dimension of the linear space, which is the number of single links m. Thus, if we can find the MMSets, we can reduce the number of measurements from $n(n-1)/2$ to m. Given the fact that HPC networks contain links only proportional to the number of terminal nodes, $m = O(n)$, we reduce the total number of measurements from $O(n^2)$ to $O(n)$, which is very significant.

The *MMSets* can be found using the Gaussian elimination method. Due to different order of elements in S, the Gaussian elimination method can result in different valid *MMSets*. This suggests we have different minimal measurement path sets. For the previous sample network, we can obtain three different *MMSets* which are:

$$S_1^{'} = \{\alpha_{<k_1,k_2>}, \alpha_{<k_1,k_3>}, \alpha_{<k_1,k_4>}, \alpha_{<k_1,k_5>}, \alpha_{<k_1,k_6>}, \alpha_{<k_2,k_5>}, \alpha_{<k_3,k_4>}\},$$
$$S_2^{'} = \{\alpha_{<k_1,k_2>}, \alpha_{<k_1,k_3>}, \alpha_{<k_1,k_4>}, \alpha_{<k_1,k_5>}, \alpha_{<k_1,k_6>}, \alpha_{<k_2,k_6>}, \alpha_{<k_3,k_4>}\},$$
$$S_3^{'} = \{\alpha_{<k_1,k_2>}, \alpha_{<k_1,k_3>}, \alpha_{<k_1,k_4>}, \alpha_{<k_1,k_5>}, \alpha_{<k_1,k_6>}, \alpha_{<k_5,k_6>}, \alpha_{<k_3,k_4>}\}$$

c. Measure the latency of paths in $S^{'}$ in parallel.

We can simultaneously measure the latency of paths that do not contain the same single link. We define a measuring path graph $MPG\,{<}V, E{>}$ in which each vertex represents a measuring path and edge between the two vertexes indicates that the two measuring paths represented by these two vertex share at least one simple link. We propose an innovative method based on graph coloring to divide the graph into a number of subsections and simultaneously measure the latency of all paths in the same subsections. The method stipulates that adjacent vertexes can not have same color. Finally, according to the graph coloring results, we can determine the number of parallel measurements and the path set to be measured in each measuring round. For graph coloring is essentially NP-Hard problem, we use an adaptive coloring algorithm, such as the Welch Powell algorithm, when the graph is large. Only when the measurement set is small enough, we make use of the divide algorithm to get an optimal scheme.

It should be noted that there are often multiple $S^{'}$ for the same S. Although different $S^{'}$ have the same number of measuring paths, the layout of measuring paths in those set are different, which bring different coloring results. For small networks, we determine an optimal $S^{'}$ as the final *MMset* by comparing the coloring results of all $S^{'}$. For large scale networks, we randomly select some sets from all $S^{'}$ and find out the one with best dyeing scheme as the final optimized *MMSet*. In the previous network, we select $S_1^{'}$ as the final *MMSet* because there are the same coloring results for all three $S^{'}$. The $MPG\,{<}V, E{>}$ colored is showen in Fig. 3. Five rounds of measurement will be carried out finally.

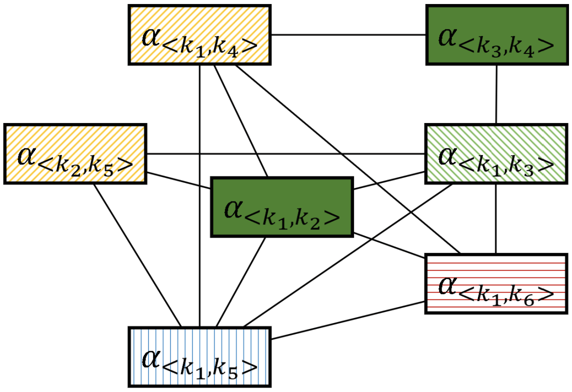

Fig. 3. A coloring result of *MMSet*. Five instead of 7 rounds of measurement is needed finally. (Color figure online)

d. Construct single link latency equations to calculate the latency of all paths in S.

Let $O^{'} = (O_1, O_2, \cdots, O_x)$ be the latency of all paths in *MMset* after parallel measuring. We construct a matrix C which contains x rows and m columns whose rows correspond to the single link composition of measuring paths in *MMset*. We can get a general solution by solving equation $C \cdot \beta^T = O^{'}$. Any solution can be used to calculate the unique latency of all measuring paths in $S^{'}$, which means that we can also calculate the unique latency of all measuring paths in S. For the previous network, suppose that the real latency of each path in the network are $O_{p^{rtt}_{k_1,k_2}} = 16$, $O_{p^{rtt}_{k_1,k_3}} = 37$, $O_{p^{rtt}_{k_1,k_4}} = 36$, $O_{p^{rtt}_{k_1,k_5}} = 18$, $O_{p^{rtt}_{k_1,k_6}} = 17$, $O_{p^{rtt}_{k_2,k_3}} = 39$, $O_{p^{rtt}_{k_2,k_4}} = 38$, $O_{p^{rtt}_{k_2,k_5}} = 20$, $O_{p^{rtt}_{k_2,k_6}} = 19$, $O_{p^{rtt}_{k_3,k_4}} = 25$, $O_{p^{rtt}_{k_3,k_5}} = 41$, $O_{p^{rtt}_{k_3,k_6}} = 40$, $O_{p^{rtt}_{k_4,k_5}} = 40$, $O_{p^{rtt}_{k_4,k_6}} = 39$, $O_{p^{rtt}_{k_5,k_6}} = 21$. After only measuring the latency of x paths in $S^{'}$, we get a solution $o_{l_1} = 3.5$, $o_{l_2} = 4.5$, $o_{l_3} = 8.5$, $o_{l_4} = 0$, $o_{l_5} = 6.5$, $o_{l_6} = 6$, $o_{l_7} = 5.5$, $o_{l_8} = 5$ which can be used to calculate the latency of all paths in S.

Although it is not necessary to calculate all aggregated links' latency for getting path latency, the latency of the aggregated link reflects the characteristics of the network in more detail. It is useful in some application scenarios, such as link fault detection. According to step b, we know $rank(C) \leq m$. When $rank(C) = m$, the equation has unique solution. When $rank(C) < m$, the equation has countless solutions which means that some single links' latency in the network can not by accurately calculated. We propose a method of link aggregation, which can merge several single links into an aggregated link to ensure all aggregated links' latency in network is accurate and unique. We construct augmented matrix $(C|O^{'})$ and transfer it into row canonical form matrix G. All non-zero columns in a row correspond to all single links in aggregated link and the last column represents the latency of the aggregated link. In our example, the matrix $(C|O^{'})$ and G are shown in Eq. 2. The latency of all aggregated links are

$o_{l_1} = 3.5, o_{l_2} = 4.5, o_{l_3} + o_{l_4} = 8.5, o_{l_5} = 6.5, o_{l_6} = 6, o_{l_7} = 5.5, o_{l_8} = 5$. l_3 and l_4 make up an aggregation link, which is reasonable for that they always transmit the data at the same time.

$$(C|O^{'}) = \begin{bmatrix} 2\,2\,0\,0\,0\,0\,0\,0\,16 \\ 2\,0\,2\,2\,2\,0\,0\,0\,37 \\ 2\,0\,2\,2\,0\,2\,0\,0\,36 \\ 2\,0\,0\,0\,0\,0\,2\,0\,18 \\ 2\,0\,0\,0\,0\,0\,0\,2\,17 \\ 0\,2\,0\,0\,0\,0\,2\,0\,20 \\ 0\,0\,0\,0\,2\,2\,0\,0\,25 \end{bmatrix}, G = \begin{bmatrix} 1\,0\,0\,0\,0\,0\,0\,0\,3.5 \\ 0\,1\,0\,0\,0\,0\,0\,0\,4.5 \\ 0\,0\,1\,1\,0\,0\,0\,0\,8.5 \\ 0\,0\,0\,0\,1\,0\,0\,0\,6.5 \\ 0\,0\,0\,0\,0\,1\,0\,0\;\;6 \\ 0\,0\,0\,0\,0\,0\,1\,0\,5.5 \\ 0\,0\,0\,0\,0\,0\,0\,1\;\;5 \end{bmatrix} \quad (2)$$

4 Validation and Analysis

4.1 Exprimental Settings

Since our method is based on rigorous mathematical process, the method is applicable to arbitrary indirect networks. Thus as a validation, we only evaluate the effectiveness of our method in synthesised fat-tree networks. We implement a source routing fat tree network simulator using the topology described in [9], to simulate fat-tree networks commonly used in data centers and supercomputers. $p-port\ q-tree$ InfiniBand network which contains $2{\times}(p/2)^q$ nodes and $2{\times}q{\times}(p/2)^q$ single links are simulated. To simulate typical fat-tree networks, we choose 7 different fat-tree configurations as shown in Table 2.

Table 2. Fat-tree configurations used in the experiments

Configuration	Number of terminal nodes	Number of links
$4-port2-tree$	8	16
$4-port3-tree$	16	48
$6-port3-tree$	54	162
$8-port3-tree$	128	384
$10-port3-tree$	250	750
$12-port3-tree$	432	1296
$16-port3-tree$	1024	3072

4.2 Accuracy of the Measurement

We first show our method can recover the link latency of the network. We design the following experiments: Firstly, We set every link in the network a random latency. Secondly, we compute a parallel measurement plan using our method. We carry out the measurement by simply aggregating the link latencies along the measuring path. Thirdly, we calculate the latency of all measuring paths and aggregated links in the network. Finally, we check those calculated link latency with the preset values. Our method finds the correct values for all the links. Table 3 shows that the calculated latency of all measuring paths is the same as the actual values in 4-port 2-tree network separately. In fact, we get the same conclusion as this example in the other 6 networks.

Table 3. Actual latency and calculated latency of all measuring paths in $4 - port$ $2 - tree$ network

(a) Actual latency of all measuring paths

Node	1	2	3	4	5	6	7	8
1	0	25	54	51	67	49	58	54
2	25	0	55	52	68	50	59	55
3	54	49	0	25	61	53	52	58
4	57	52	25	0	64	56	55	61
5	67	55	61	61	0	31	65	64
6	62	50	56	56	31	0	60	59
7	58	53	52	59	65	57	0	30
8	60	55	54	61	67	59	30	0

(b) Calculated latency of all measuring paths

Node	1	2	3	4	5	6	7	8
1	0	25	54	51	67	49	58	54
2	25	0	55	52	68	50	59	55
3	54	49	0	25	61	53	52	58
4	57	52	25	0	64	56	55	61
5	67	55	61	61	0	31	65	64
6	62	50	56	56	31	0	60	59
7	58	53	52	59	65	57	0	30
8	60	55	54	61	67	59	30	0

4.3 Measurement Reduction

We then show that our method can greatly reduce the number of measurements in full-network point-to-point latency measurements. We compute the measurement plan for 6 different network configurations, and compute the round of measurements required. Each round of measurements involves a collection of measurements can be done concurrently. We assume one measurement takes T seconds, and compare the total measurement execution time in Fig. 4. We compare our method with the brute-force one-by-one measurement of all node pairs. In the brute-force method, it takes us $(n \times (n-1)/2)T$ seconds to measure the latency of all paths serially. In our measurement method, it takes about m T seconds to serially measure the latency of all paths in *MMset*. In the network with 3-tree, the total measurement time can be further reduced by 33.3% compared with the serial measurement. With parallel measuring the latency of paths in the same *MMset*, only n T seconds are needed. We can conclude that the proposed methods can reduce the overhead of large-scale fat-tree networks containing thousands of nodes by three orders of magnitude.

4.4 Complexity Analysis of the PMM Method

Although the proposed method reduces the time costed in measuring the latency, it brings additional computing overhead. We analyze the complexity of the extra computing here. We choose the time during which CPU completes an arithmetic operation or access a variable in memory as the unit.

The first part of the computing overhead comes from generating the measurement scheme. We use *Gaussian elimination* to transfer matrix A into row echelon form for getting all maximal linear independent subsets of S, during which about m eliminations are required. In each elimination, we need to look up an main row from $n(n{-}1)/2$ rows firstly, and then carry out $n(n{-}1)/2$ elementary transformations. Thus the average time overhead of *Gaussian elimination* is T_1.

$$T_1 = m(mn(n-1)/2 + mn(n-1)/2) = m^2 n(n-1). \tag{3}$$

The second part of the computing overhead comes from deriving $MPG{<}V,E{>}$ to get parallel measurement scheme. We use Welch Powell algorithm to get an optimized solution of the NP-Hard Graph Dying problem in large-scale network. The time complexity of the algorithm is $O(m^3)$.

The third part of the computing overhead comes from calculating the latency of all paths and links. Our method use *Gaussian elimination* to solve m linear equations for getting the latency of all aggregated links, and then calculate the latency of all paths. The average time overhead is T_2

$$T_2 = 2m^3 + n(n-1)/2 \tag{4}$$

For $p - port\ q - tree$ network, $n < m < n(n-1)/2$. As a result, a loose time complexity of our method is $O(n^2{\cdot}m^2)$.

We further investigate reducing the computing overhead by parallel computing. We substitute the Gaussian elimination with a MPI based implementation and run the computing of a $12 - port\ 3 - tree$ with 432 nodes and 1296 links on Tianhe-2 super computer. The timing results are shown in Fig. 5 and it shows than we can compute the measurement plan in less than 30 s with 116 MPI processes, which is pretty acceptable in HPC environments.

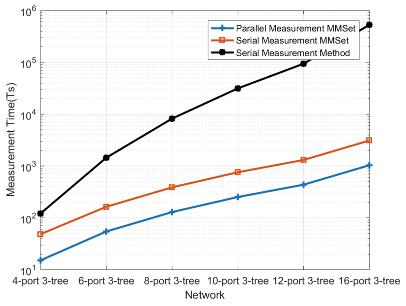

Fig. 4. The measurement time of two methods. Each measurement takes T seconds.

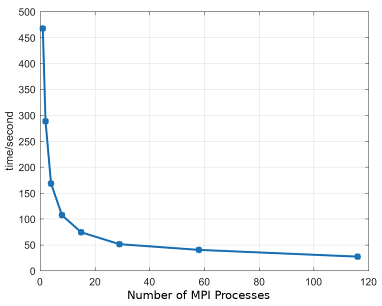

Fig. 5. The computing overhead of generating measurement plan and calculating the latency of all paths and links in $12 - port\ 3 - tree$ network in parallel settings

5 Applications

Being a low level method, our *PMM* method can be used in many application scenarios where full point-to-point latency is required. We discuss some of these applications in this section.

5.1 Communication Performance Modeling and Prediction

In many cases we want to model the communication network, so as to predicate the application performance on given supercomputers, to inspect the communication bottlenecks of parallel applications, and to compare design alternatives of network parameters. For example, when we optimize the application communication performance, we can use trace simulators such as LogGOPSim [10] to simulate the communication and find the bottlenecks. The LogGOPSim relies on point-to-point latency to make an accurate predication for small messages, which often require one to measure the full-network point-to-point latency of a given super-computer. Our methods can greatly reduce the number of measurements and thus improve the model accuracy by being able to incorporate the difference of per node pair latencies.

5.2 Transitional Link Failure Detection

Transitional link failures happens a lot on large scale high performance computer networks, which often results in downgraded communication performance, and gradual system failures. Extra hardware can be built into the network to moniter each link to detect these problematic states, but this is not practical on many networks. Our method provides a software-based alternative. One can generate a measurement plan for any suspecting subnet and measure the point-to-point latency quickly to obtain per-link latency, and flag links with larger latency than expected as problematic for further investigation.

5.3 Parallel Communication Optimization

Automatic optimization of communication performance often requires knowing the inter-node message latency of the running nodes, which can only be measured online. For example, in topology-aware process mapping algorithms, one often needs to model the per-note message latency, and accurate online modeling of these latency is essential for real-world parallel applications. Our method can help by generating the measurement plan and measure the point-to-point latency on the fly quickly, thus make the optimization applicable to any indirect networks.

6 Conclusion

In this paper, we propose an efficient method, namely *PMM*, to generate full-network point-to-point latency measurement plans for arbitrary indirect HPC networks. Our method reduces the measurements required from $O(n^2)$ to $O(n)$ for modern high performance computer networks such as fat-tree based infiniband networks, and can be extremely useful in communication performance modeling, transitional link failure detection, and parallel communication optimization.

Although being effective, there are still aspects to improve in our methods. We go through some or all *MMsets* to find out an optimized one in our method, which is ineffective. We also consider find out heuristics to locate measurement plans with the maximal parallelism. We can also make the measurement additive to allow for continuously monitoring link latencies.

Acknowledgement. This work in this paper is supported by the National Key R&D Program of China under Grant NO. 2018YFB0203901, Science Challenge Project, NO. TZ2016002, and the National Natural Science Foundation of China under Grant No. 61772053. The authors would like to thank the reviewers for their valuable comments.

References

1. Alexandrov, A., Ionescu, M.F., Schauser, K.E., Scheiman, C.: LogGP: incorporating long messages into the LogP model. J. Parallel Distrib. Comput. **44**, 71–79 (1995)
2. Ino, F., Fujimoto, N., Hagihara, K.: LogGPS: a parallel computational model for synchronization analysis. ACM SIGPLAN Not. **36**, 133–142 (2001)
3. Bhanot, G., Gara, A., Heidelberger, P., Lawless, E., Sexton, J.C., Walkup, R.: Optimizing task layout on the Blue Gene/L supercomputer. IBM J. Res. Dev. **49**, 489–500 (2005)
4. Szymaniak, M., Presotto, D., Pierre, G., Steen, M.V.: Practical large-scale latency estimation. Comput. Netw. **52**, 1343–1364 (2008)
5. Sen, S., Wang, J.: Analyzing peer-to-peer traffic across large networks. In: Proceedings of the 2nd ACM SIGCOMM Workshop on Internet measurement, no. 2, pp. 137–150 (2002)
6. Liu, J., Zhang, X., Li, B., Zhang, Q., Zhu, W.: Distributed distance measurement for large-scale networks. Comput. Netw. **41**, 177–192 (2003)
7. Guo, C., et al.: Pingmesh: a large-scale system for data center network latency measurement and analysis. In: ACM SIGCOMM Computer Communication Review, vol. 45, pp. 139–152 (2012)
8. Shavitt, Y., Sun, X., Wool, A., Yener, B.: Computing the unmeasured: an algebraic approach to Internet mapping. IEEE J. Sel. Areas Commun. **22**, 67–78 (2004)
9. Lin, X,Y., Chung, Y,C., Huang, T,Y.: A multiple LID routing scheme for fat-tree-based InfiniBand networks. In: Parallel and Distributed Processing Symposium, 18, p. 11 (2004)
10. Hoefler, T., Schneider, T., Lumsdaine, A.: LogGOPSim: simulating large-scale applications in the LogGOPS model. In: Proceedings of ACM International Symposium on High Performance Distributed Computing, 19, pp. 597–604 (2010)