Optimization of Link Bandwidth for Parallel Communication Performance

Lydia Y. Chen
IBM Research Zurich Laboratory,
8803 Rüschlikon, Switzerland
Email: yic@zurich.ibm.com

Wolfgang Denzel
IBM Research Zurich Laboratory,
8803 Rüschlikon, Switzerland
Email: wde@zurich.ibm.com

Ronald Luijten
IBM Research Zurich Laboratory,
8803 Rüschlikon, Switzerland
Email: lui@zurich.ibm.com

***Abstract*—The efficiency of computer network has been regarded as a bottleneck in parallel computing paradigm. It is important to have efficient methodology to obtain network performance measures, especially for a large scale system, i.e. exa-scale system. Communication performance is often investigated by the static complexity analysis based on a given network topology or a detailed network simulation, which is often time consuming. To provide a dynamic and scalable communication performance measure, we first propose an aggregate multi-stage queueing network model to capture the application's communication load and derive the closed-form system performance, i.e. throughput and delay. Trace simulation results obtained from a sophisticated simulator, Venus, show that the proposed model is accurate, yet simple. Secondly, we develop a link bandwidth optimization framework, which optimally allocates/distributes link bandwidth across the network to maximize the system communication throughput. Specifically, we apply the derived optimal bandwidth allocation on dimensioning link bandwidth of an exploratory direct network and slimming fat-tree network. Our results show that the proposed methodology is cost-effective in providing system performance and design explorations for the existing and the next-generation network system.**

I. Introduction

The complexity and scale of today's supercomputer centers grow exponentially to host large-scale parallel applications, requiring a huge amount of computing and communication resources. The advancement of contemporary processor technology enables efficient parallel computing, whose communication commonly adopts the Message Parsing Interface (MPI). The communication efficiency of MPI on the interconnect network is regarded as an emerging bottleneck in modern parallel paradigm [23], [18], [9], especially for the large-scale system. To improve the parallel efficiency of a given application, it is important to have a scalable methodology to obtain accurate communication performance measures and further adjust the network resource provisions and applications.

A large number of complexity analysis [17] have been developed to study and analyze the performance of parallel algorithms on various network topologies. The derived speed-up is based on an assumed network message delay, which actually depends on the dynamics between MPI message loads and underlying network resources. To capture the dynamic communication performance, MPI tracing tools are commonly deployed in today's supercomputers to record the communication pattern among applications' MPI tasks, which are simulated on various network configurations. The parallel performances, i.e. actual message delay and speed-up, is essentially the projection of MPI tasks communication on the interconnect networks. Nevertheless, it's not a trivial endeavor to simulate the petascale or exascale network system across wide spectrums of architectures.

Among all network resources, the link bandwidth is the first order effect in communication performance and therefore the allocation of that is crucial to achieve a satisfactory parallel speed-up for applications. The modern multi-stage networks consisting of multi-level links are built, such as MareNostrum [2], to connect a large number of processing units, associated with specific MPI tasks. The link utilization and delay at each level depend on the communication pattern among MPI tasks. An under-provision or over-provision of link bandwidth at certain levels can lead to other level link bottlenecks and thus throttle the entire network throughput. For example, slimming fat-tree network to re-allocate link bandwidth is one of the challenges. To maximize the sustainable network performance, an efficient link bandwidth allocation of interconnect network needs to incorporate the recursive routings of different link levels, driven by applications' MPI communication.

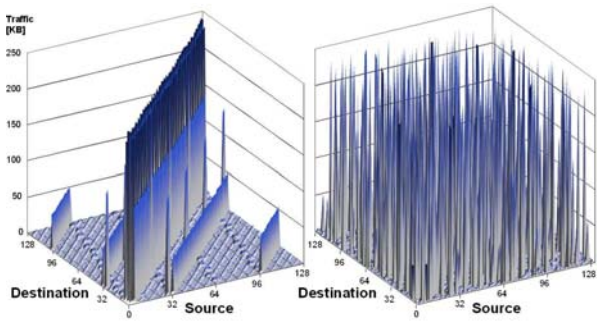
This study takes a network resources centric approach to derive and improve the parallel communication performance. We first propose to use a simple multi-stage queuing network to capture the first-order performance dynamics of MPI messages on various networks. To validate the proposed methodology, a sophisticated simulator, Venus [13], is used to simulate the detailed network system specifications, which are simplified in the proposed model. Secondly, we propose a link bandwidth optimization based on the projected MPI communication matrix to maximize the overall system throughput and improve the application delay. The proposed optimal link bandwidth allocation strategy can be applied to design an exploratory direct network and a slimming fat tree network. Experimental results from traces and workload benchmark show that our proposed approach can accurately predict communication performance, scale well, and practically improve the system throughput by optimally allocating level-wise link bandwidth.

The remainder of this paper is organized as follows. The discussion of projected network communication of MPI task is given in Section II. The proposed system modeling and bandwidth allocation are described in Section III. Section

IV contains the experimental results. The related work is presented in Section V and Section **??** concludes the work and future direction.

II. Projected MPI Communication on Network (PMCoN)

In this section, we illustrate the concept and characterization of the projected MPI communication on network, which is used as the building block to derive the communication performances in Section III. The PMCoN consists of two transition steps from given MPI task communication and network architectures. We first show the communication patterns resulting from mapping the MPI tasks to network nodes. Secondly, we present the projected communication loads on network links. Specifically, the MPI traces of LAMMPS and UMT2K [1] are applied to various fat-tree networks.



(a) Sequential placement (b) Random placement

Fig. 1. Mapping/placing MPI tasks on processing nodes.

MPI tasks are defined by parallel algorithms and programming paradigm of applications. To obtain actual communication performance, e.g. message delay, from a given MPI trace on a given network configuration, the MPI tasks need to be first mapped to process nodes. We consider two mapping schemes, sequential placement and random placement. Note that we interchangeably use task mapping and placement throughout this paper. Fig. 1 depicts an example of the projected communication pattern of mapping LAMMPS tasks to 128 nodes with sequential and random placement. One can observe that in principle the collected MPI task communication pattern is often near neighboring and thus results in a very regular projected communication pattern under sequential task placement. In contrast, a very irregular and even long-haul pattern is observed in the random placement of MPI tasks. Nevertheless, the different network configurations can further affect message communication performance under various task mapping/placement schemes.

MPI tasks on processing nodes communicate through multi-level interconnect network, which can be either direct network [12], such as hypercube, or indirect network, such as fat-tree. The MPI messages traverse from the source to destination nodes through levels of links according to the routing schemes, network architecture, and task placement scheme. We call the resulting MPI traffic on each level of network links, the

TABLE I
Average packet delay ($\mu s.$) of UMT2K on fat-tree network.

	4-ary-4-tree	8-ary-3-tree	16-ary-2-tree
Sequential task placement	1.30	1.18	1.15
Random task placement	8.59	6.54	4.24

Projected MPI Communication on Network (PMCoN). We show the statistical characterization of PMCoN of the UMT2K under various fat-tree networks and task placement.

We consider a total number of 256 processing nodes, in each of which a single MPI task is placed. The nodes are connected by networks of 8-ary 4-tree, 16-ary 3-tree and 32-ary 2-tree. An example of three-level fat-tree network is shown in Fig. 2 and higher level links, i.e. third level, are used to further communicate destination nodes. The percentages of the highest fat-tree level the messages traverse through are summarized in Fig. 3. We can see that from a given MPI trace very different projected communications on network links result. Thus, the average message packet delay listed in Table I differs. We believe such as those statistics shown in Fig. 3 can be obtained through two approaches: (a) MPI trace simulation ; (b) approximated aftermath calculation based on the given MPI task communication pattern, the network architecture, and task placement. As MPI task communication of several applications is well studied [22], [6], [10], [23], we assume the projected communication of MPI on network can be estimated as a given parameter in the proposed analytical model in the following sections.

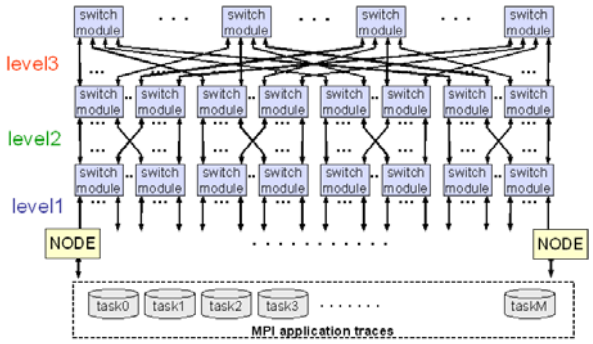


Fig. 2. Three-level fat-tree network.

III. Performance Model of Link Bandwidth

We propose a general multi-stage model to obtain the MPI communication performance measures on a multi-level network, which can be either a direct or an indirect type. The model has a total of K stages, which correspond to the total link levels of the network. The i^{th} stage consists of a single buffer and single link server, whose capacity, μ_i, equals the aggregate link bandwidth at the i^{th} level of the network. When an i^{th} level of a network has n_i number of links with the bandwidth of b_i GB per second, aggregate link bandwidth is $\mu_i = n_i b_i$. We assume the buffer space is sufficiently large that the starvation and blocking between levels is negligible.

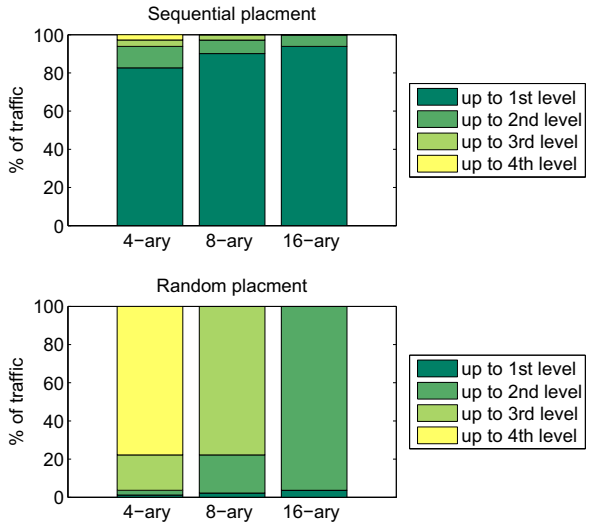


Fig. 3. Traffic statistics on fat-tree topology of 4-ary 4-tree, 8-ary 3-tree, and 16-ary 2-tree with sequential tasks placement (top) and random tasks placement (bottom).

Fig. 4 illustrates a three-level link server model, which is applicable to any k-ary 3-tree networks or three-level direct network, which is detailed in Section IVA.

The MPI message dynamics in the proposed model is as follows: Messages are generated by processors at a total rate λ per second and sent/routed through the link server of the network to destination processors. If there is not sufficient capacity at the network system, the throughput of message packets will be throttled because of the back pressure or embedded flow control mechanism at rate d. The effective arriving messages of a rate $\lambda^e = \lambda - d$ traverse link servers with probabilities, p_{ij} from i^{th} to j^{th} link server.

The 0^{th} level denotes the source or destination processing level, which is considered external to the network system. Specifically, p_{0i} represents the probability of the i^{th} level server link being the first hop of the effective arriving messages, and p_{i0} represents the probability effective messages use the i^{th} level server link as the last hop to reach destination processors. In our experimental prototype system [13], messages from processing units can be directly injected into and at every link level, whereas in a fat-tree network messages can only enter and leave the network through the first level link. Therefore, $p_{0i} = 0$ and $p_{i0} = 0, i > 1$ for the fat-tree network.

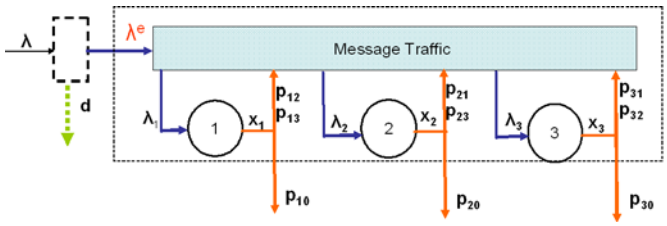


Fig. 4. The three-level link server model.

An i^{th} stage has arrival rate, λ_i, and throughput, x_i, as depicted in Fig.5. Only p_{i0} percentage of the throughput of subsystem i, $x_i p_{i0}$, is effectively directed to the destination processors and contributes to the total system throughput. The remainder of subsystem throughput, $\sum_{j\neq i} p_{ij}x_j$, will be re-directed into other link servers to form the so-called internal arrivals for other link levels. As a result, the arrival rate of i^{th} subsystem, λ_i, is the sum of external arrival, $\lambda^e p_{0i}$, and the internal arrivals, $\sum_{j\neq i} p_{ji}x_j$, routed from the other levels. The stability of i^{th} system is

$$\lambda_i = \lambda^e p_{0i} + \sum_{j\neq i} p_{ji}x_j \leq \mu_i.$$

The stability of the entire system is $\lambda_i < \mu_i \forall i$. The total system throughput, Θ, is the sum of the effective throughput of all link servers,

$$\Theta = \sum_i p_{i0}x_i.$$

Note that the subsystem throughput is equal to its total of arriving messages, meaning

$$\lambda_i = x_i,$$

under the scenarios of no head-of-line blocking in switches and no starvation among levels. Those conditions are well supported by today's switching technology and sufficiently large buffer space, and therefore are implicitly assumed in this study.

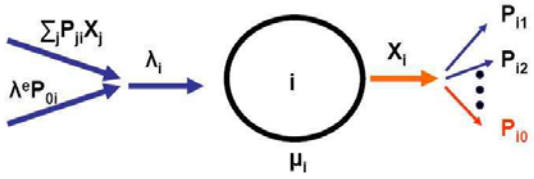


Fig. 5. Routing probability of link server i.

Derivation of p_{ij}: In general, p_{ij} can be estimated by the number of messages using j^{th} level links immediately after using i^{th} level links divided by the total number of messages using i^{th} level links. One can extrapolate such information by the estimated average distance of node communication driven by MPI task communication and the corresponding link hops required. The derivation of p_{ij} essentially depends on the projected MPI communication on network. In our experience, it is more complicated to analytically derive p_{ij} in the direct network. It's more straightforward to derive p_{ij} for the fat-tree network. Lets take the instance of the sequential placement of UMT2K on 8-ary 3-tree shown in Fig 9. One can deduce the probability hopping to the third level link after using the second level as $p_{23} = p_{32} = \frac{1}{2}\cdot$(percentage of traveling up to the 3^{rd} level)/(percentage of traveling up to the 2^{nd} and 3^{rd} level).

A. Optimal Throughput

We construct an optimization problem $(T1)$ to derive the optimal system throughput, Θ^*, the throttling rate, d^*, and

TABLE II
NOTATION

p_{ij}	: the probability of using i^{th} level link to j^{th} level link
λ	: total arrival rate
x_i	: throughput of level i.
λ^e	: total effective arrival rate
λ_i	: arrival rate of level i.
μ_i	: aggregate capacity of level i.
d	: messages throttling rate
A	projected MPI communication matrix

the subsystem throughput, x_i^* for given p_{ij} and aggregate bandwidth, μ_i. $T1$ has the objective of maximizing the system throughput ($\sum_i^K p_{i0}x_i$) and the system stability related constraints, which have been explained above.

$$\begin{aligned}(T1)\max \quad &= \quad \sum_{i=1}^{K} p_{i0}x_i \\ st \quad & \quad \lambda = \lambda^e + d \\ & \quad x_i = p_{0i}\lambda + \sum_j p_{ji}x_j,\ i \geq 1 \\ & \quad \mu_i \geq x_i,\ i \geq 1 \\ & \quad d \geq 0.\end{aligned}$$

Prior to deriving the optimal throughput, we first show a lemma to conclude the system stability condition by a **projected MPI communication matrix**, **A**. The definition of A is very effective to present the recursive hops among link servers of given message routings, especially in the direct network, and ease the derivation of the optimal throughput.

Lemma 3.1: Let d^* be the optimal solution in $T1$. The stability condition of the system:

$$\mu_i \geq (\lambda^e)A_i^{-1}b = (\lambda - d^*)A_i^{-1}b,\ \forall i \geq 1,$$

where A_i is the i^{th} row of

$$A = \begin{bmatrix} -1 & p_{21} & \dots & p_{K1} \\ p_{12} & -1 & \dots & p_{K2} \\ \vdots & \vdots & \ddots & \vdots \\ p_{1K} & p_{2K} & \dots & -1 \end{bmatrix} \text{ and } b = \begin{bmatrix} -p_{01} \\ -p_{02} \\ \vdots \\ -p_{0N} \end{bmatrix}.$$

Proof: As d^* is the optimal solution of $T1$, $\lambda^e = \lambda - d^*$. Let a vector $X = (x_1 \dots x_K)'$. After algebraic operations, the constraints can be re-written as $AX = (\lambda - d)b$. The optimal throughput can thus be expressed by d^* and A as follows:

$$X^* = (\lambda - d^*)A^{-1}b.$$

Therefore, $x_i = (\lambda - d^*)A_i^{-1}b$, $i \geq 1$ and the stability condition follows from the afore-mentioned system stability, $\mu_i > \lambda_i = x_i$. ■

We can re-write $T1$ by using $x_i = (\lambda - d)A^{-1}b$, $i \geq 1$, so that $T1$ becomes an optimization problem, $T1'$, with d as a variable and given parameters of A and b. The closed form of optimal throughput can be obtained from $T1'$.

$$\begin{aligned}(T1')\max \quad &= \quad \sum_{i=1}^{K} p_{i0}(\lambda - d)A_i^{-1}b \\ st \quad & \quad \lambda = \lambda^e + d \\ & \quad \mu_i > (\lambda - d)A_i^{-1}b\, i \geq 1 \\ & \quad d \geq 0.\end{aligned}$$

Theorem 3.2: The optimal solution of d^* and X^*

1) $d^* = 0, X^* = \lambda A^{-1}b$, if $\mu_i \geq \lambda A_i^{-1}b\, i \geq 1$.
2) Otherwise, $X^* = (\lambda - d^*)A^{-1}b$, and $d^* = \min_i\{\lambda - \frac{\mu_i}{A_i^{-1}b}\}$.

Proof: The objective in $T1'$ is a decreasing function of d. If $\mu_i \geq \lambda A_i^{-1}b \forall i$, the $d* = 0$ because of maximizing the objective, and thus $X^* = \lambda A^{-1}b$. Otherwise, d^* needs to be chosen such that the stability constraint is met in ($T1'$). These constraints can be re-organized and as the lower bound for d:

$$d \geq \lambda_i - \frac{\mu_i}{A_i^{-1}b}\, i \geq 1.$$

To maximize throughput, d^* should be minimized. Therefore, when $\mu_i < \lambda A_i^{-1}b\, i \geq 1$, and $d^* = \min_{i\geq 1}\{\lambda_i - \frac{\mu_i}{A_i^{-1}b}\}$. ■

1) The average packet delay: The average packet delay can be obtained by the product of the average hops of each level, N_i, and the delay of a single hop of each level, D_i, which is the function of the link utilization($\frac{\lambda_i}{\mu_i}$). The delay can be estimated by $M/M/1$ model [8], $D_i = \frac{1}{\mu_i - \lambda_i}$, and therefore,

$$D = \sum_i N_i D_i = N_i \frac{1}{\mu_i - \lambda_i}.$$

Note that in reality the buffer is limited, and that a larger buffer dimension can reduce the issue of blocking and starvation across levels. To maximize the system throughput and minimize the delay, the system needs to deploy a well-dimensioned buffer and employ the flow control mechanism to throttle packets at rate d^*.

B. Link bandwidth Allocation

The system throughput can be derived through the proposed $T1'$, where d and X are the only variables and the aggregate link bandwidth of all levels are just given parameters. The lower system throughput and higher throttling rate can result from inefficient link allocation. Nevertheless, the system throughput can be improved if the link bandwidth allocation is more efficient. We therefore propose another total throughput optimization with variables of the subsystem throughput x_i, and also aggregate link bandwidth, μ_i. We assume there is an upper bound in total link, bandwidth allocation, denoted by U. An additional constraint,

$$\sum_{i=1}^{K} \mu_i \leq U,$$

is added in $T1'$.

Theorem 3.3: Optimal aggregate link bandwidth allocation to maximize the throughput is

$$u_i^* = \frac{A_i^{-1}b}{\sum_{i=1}^{K} A_i^{-1}b} U.$$

Proof: We first show the total throughput is maximized only when $\sum_{i=1}^{K} \mu_i = U$ and then provide the optimal allocation of U for all levels. As the objective function is a decreasing function of d and $d > \{\lambda - \frac{\mu_i}{A_i^{-1}b}, i \geq 1\}$, the objective function is an increasing function of μ_i and $\sum_i \mu_i$ as well. Therefore, when $\sum_i \mu_i = U$, the throughput can be maximized.

From constraint $\frac{\mu_i}{A_i^{-1}b} > (\lambda - d)\, \forall i$, we know $(\lambda - d) \leq \min_i \frac{\mu_i}{A_i^{-1}b}$. The objective function is an increasing function of $(\lambda - d)$; maximizing the upper bound can lead to the optimal solution. Therefore,

$$\mu_i^* \in \{\max \min_{u_i} \frac{\mu_i}{A_i^{-1}b}, st \sum_i \mu_i = U\}.$$

It is straightforward to obtain the optimal service allocation, $u_i^* = \frac{A_i^{-1}bU}{\sum_i A_i^{-1}bU} i \geq 1$.

■

Theorem 3.3 states that the optimal link server capacity (the aggregate link bandwidth) of each level is proportional to the percentage of traffic loads on its own level over the entire system loads. Because of $\mu_1 : \cdots : \mu_K = A_1^{-1}b : \cdots : A_K^{-1}b$, we then know the utilization of each level is the same in the case of optimal link bandwidth allocation, meaning $\frac{(\lambda^e)A_1^{-1}b}{\mu_1} = \cdots = \frac{(\lambda^e)A_K^{-1}b}{\mu_K}$.

IV. Experiments

We adopt a large-scale end-to-end HPC simulator, Venus [13], not only to validate the proposed model, but also to demonstrate how the proposed methodology can be applied to design and improve computer networks. Venus is built using OMENST network environment, which can create any arbitrary interconnect network topology, with flexible choices of link bandwidth, switch sizes, routing algorithms, and flow control mechanisms. The message traffic is generated at the generic source processor and then routed to the destination processors in the interconnect network. Venus could use either the collected MPI traces or a synthetic traffic benchmark. As the number and size of the traces are limited, we focus on the GUPs (Giga Updates Per Second) to obtain scalability of exploratory direct networks under various bandwidth allocation schemes. Furthermore we use a MPI trace, UMT2K, to investigate the slimming of fat-tree network.

A. GUPS on Direct Network

We use GUPs (Giga Updates per Second) on a three-level exploratory direct network [13] shown in Fig. 6 to validate our proposed analytical model. GUPS packets of size 24 bytes are uniformly generated at source processors and sent to random destination processors, and routed according to Dijkstra routing algorithm [11], which allows, at most, one single hop on the highest level (3^{rd}) link. The entire system is constructed as follows: eight processing nodes form a second level node, called board; 16 second-level nodes form a third-level node, called rack. A fully connected system has 512 third-level nodes. The number of links at each level scales with the number of nodes at that level to maintain the full connectivity, so does the total bandwidth. The link bandwidth is set at 5, 10, and 10GB for the first-, second-, and third-level link, respectively. Therefore, the resulting aggregate link server capacity, μ_i has the normalized ratio of $\mu_1 : \mu_2 : \mu_3 = 3.82 : 1.02 : 1.30$. The projected MPI communication matrix, A, can be derived from the known topology, tasks dependency (random) and the Dijkstra routing schemes as explained in Section III A. Therefore, we can obtain analytical throughput from the proposed model.

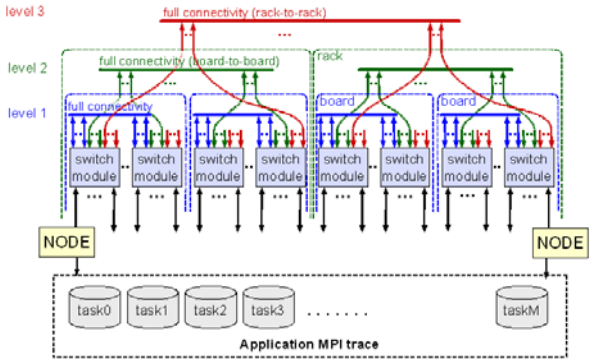
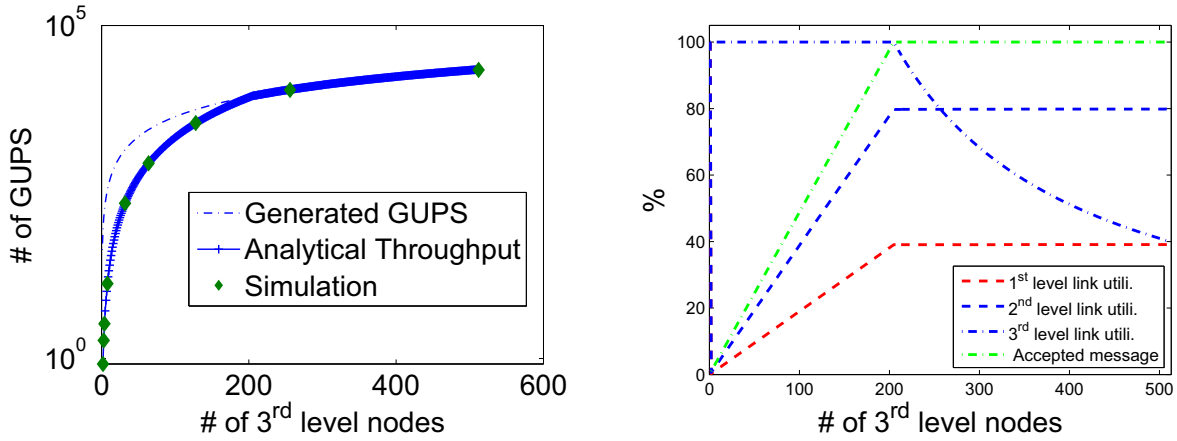


Fig. 6. Exploratory three-level direct network.

Fig. 7(a) shows that the throughput obtained from analytical model and simulation increases with the number of third-level nodes and the aggregate third-level link bandwidth. It takes time to simulate systems of such a scale [13]; significantly longer than the proposed analytical approach. As sufficient buffer dimensioning prevents the throughput degradation due to head-of-line blocking, the analytical model matches well with the simulated system. The link utilizations of each level obtained from the analytical model is illustrated Fig. 7(b). The gap between the generated GUPSs and the throughput is the throttling effect of flow control and the value corresponds to d in the proposed model. For the uniformly distributed GUPs, the first-level link bandwidth is over-provisioned and the lowest link utilization results. On the other hand, such a conservative allocation might be fairly effective for applications with mainly neighboring communication. The second-level link utilization is fairly high and could be the bottleneck link, especially in systems with a larger number of third-level nodes. Moreover, the utilization values obtained from the analytical model can be used as the threshold for monitoring the hotspot in the network.

We proved that optimal link allocation policy results in a balanced utilization of all levels in Section III, an unbalanced link utilization of a fully connected system of 512 third-level nodes has yet to reach optimal throughput as shown in Fig. 7



(a)Throughput (b) Link utilization

Fig. 7. Analytical and simulation results of GUPS on a three-level exploratory direct network.

(b). To improve the maximal system throughput, we adopt three aggregate link allocations, (1) $\mu_1 = \mu_2 = \mu_3$, (2) optimal, (3) $\mu_1 < \mu_2 < \mu_3$, given that the sum of μ_i is constant for all three cases. The optimal allocation policy follows Theorem 3.3. The optimal bandwidth allocation should keep the normalized bandwidth ratio to the normalized arrival rate ratio, meaning $\mu_1 : \mu_2 : \mu_3 = A_1^{-1}b : A_2^{-1}b : A_3^{-1}b = 3.5 : 2 : 1$. The highest system throughput and link utilizations can be achieved in the optimal case. Policy 3 provides most of the bandwidth to third-level links, which actually have the lowest workload. The third- and second-level links are always under utilized, whereas the first-level links reach 100% utilization even with the light GUPs traffic. Note that a single specific link bandwidth at the i^{th} level can be calculated by dividing μ_i with the total number of links at the i^{th} level, which depends on the network topology. The proposed performance model and the bandwidth allocation can provide not only a sound prediction of the system throughput and average link utilization, but also a design guideline to build the next-generation computer networks.

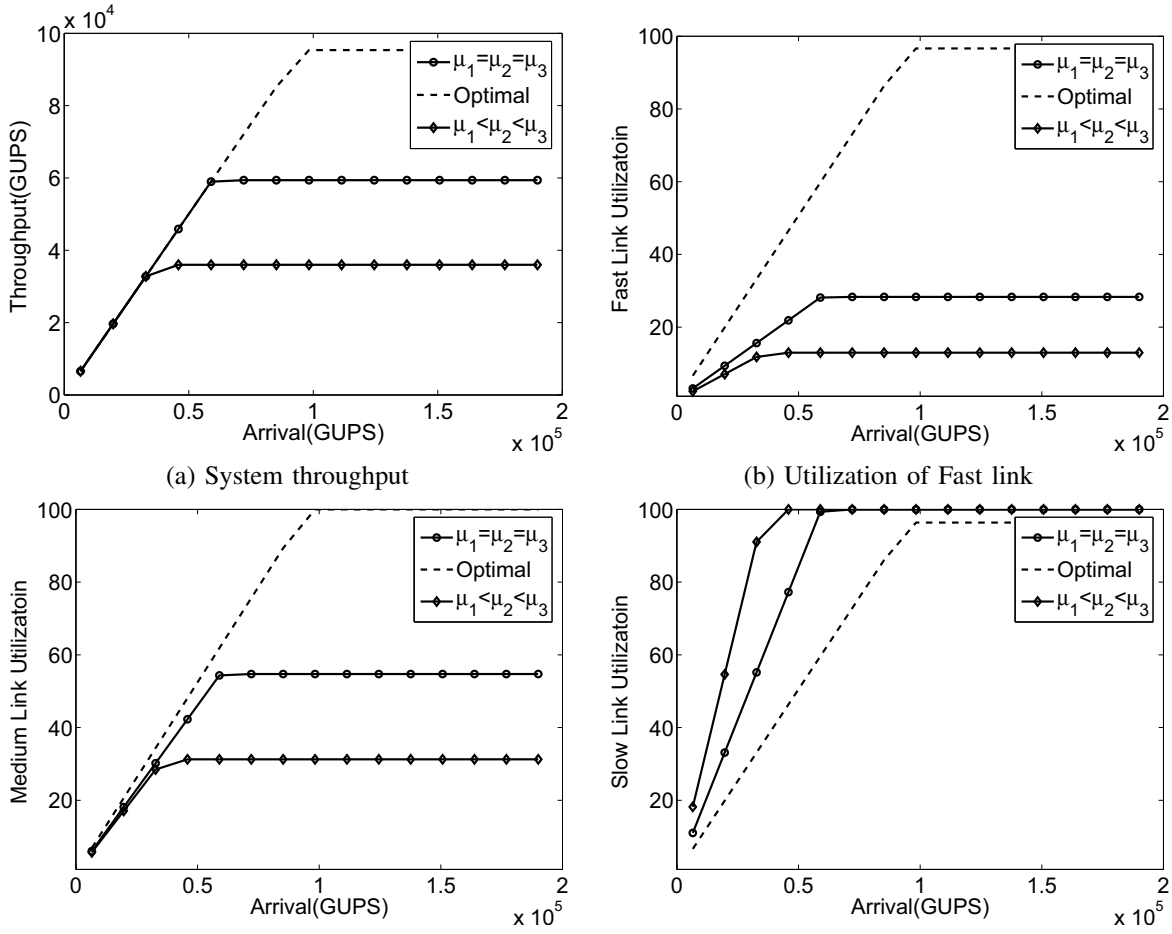
B. UMT2K on Fat-tree

We apply the proposed methodology to evaluate the performance of slimming fat-tree network, given the projected link communication matrix. In Section II, we obtained the projection MPI communication of UMT2K on various fat-tree networks. The sequential task placement of UMT2K on 4-ary 4-tree and 16-ary 2-tree has the normalized aggregate link arrivals of each level, $A_i^{-1}b$, as $17 : 1$ and $29.5 : 4.0 : 1.2 : 1$, respectively. As both result in a high volume of neighborhood communication, higher-level links have significantly low arriving loads. Meanwhile, the standard fat-tree network keeps the aggregate link bandwidth constant at each level ($\mu_1 = \mu_2 \cdots = \mu_K$). The higher-level link therefore results in a lower utilization, especially for the sequential task placement, whereas the lower-level links have much higher utilizations. According to the proposed optimal link bandwidth allocation, we can slim the tree significantly without degrading UM2K performance.

As a result, we reduce the bandwidth of each fat-tree level by 25%, 50% and 75% relative to its preceding level in Fig. 9. Note that we try to slim the fat-tree as close as to the suggested optimal allocation conservatively, because the transient load changes of applications can not be fully reflected in the proposed aggregate analysis. For the sequential placement, only a slight packet delay occurred even with 75% bandwidth reduction in a 16-ary 2-tree, as the resulted aggregate bandwidth is still more than the aggregate arrivals. In a 4-ary 4-tree, 50% reduction of bandwidth results in a visible packet delay, because the normalized aggregate link bandwidth ratio ($\mu_1 : \mu_2 : \mu_3 : \mu_4 = 0.50 : 0.25 : 0.125 : 0.62$) deviates from the normalized arrival loads $(5 : 0.6 : 0.2 : 0.1)$. The proposed aggregate analysis is more accurate for a broader and shorter fat-tree, where the workload dynamics more stable due to the restricted routing scheme. The tall fat-tree is more sensitive to bandwidth slimming. In random task placement, slimming the fat-tree increased packet delay much more, because a major projected traffic traverses higher-level links especially in the case of 75% reduction. The optimal slimming strategy for random task placement should actually try to reduce the lower-link bandwidth more than that of higher-level links.

V. Related Work

A large number of studies in the scientific community [3], [18], [14] address the parallel performance by improving MPI efficiency of applications. Specific algorithms are explicitly designed to incorporate MPI into computation routines [3], [18], [20] for better parallel efficiency. Another focus of MPI researches [22], [6], [10], [23] is to characterize the MPI communication pattern of the large-scale scientific applications traces, such as molecular dynamics and weather forecast. Their MPI task communication patterns can thus be well predicted. Many performance models and tools [5], [4], [15], [19] are also built for traffic monitoring and performance control; however the architectural complexity of interconnect networks is often



(a) System throughput

(b) Utilization of Fast link

(c) Utilization of medium link

(d) Utilization of slow link

Fig. 8. System throughput and links utilization under various bandwidth allocations (1) $\mu_1 = \mu_2 = \mu_3$ (2) the proposed optimal allocation (3) $\mu_1 < \mu_2 < \mu_3$).

over simplified or the overheads incurred by analytical and traffic monitoring tools are substantial.

Pervious research has also addressed the efficiency of the parallel communication from the architectures hardware perspectives. Bogdanov et al. [7] proposed a parallel hardware architecture for fast Gaussian elimination. Iancu and Strohmaier [16] proposed a model based bandwidth configuration and tested on infiniband and Elan4 network nodes with synthetic traffic. Utrera and et al. [21] demonstrate that allocating MPI processing nodes by the job type is robust to the workload variation. TO our best knowledge, the architecture and hardware prepared for improving MPI performance are specific to computation routines or interconnect topology, rather than to general methodology proposed by this study.

VI. Conclusions and future direction

We have proposed an analytical performance model to allocate the link bandwidth of supercomputer networks, based on the MPI projected communication on network, which is determined by the network topology, MPI task placement, and routing scheme. An optimization technique is used to derive the maximum system throughput, link utilization, and the optimal link bandwidth allocation, with a good scalability with system size. The simulator, Venus, has been adopted to characterize the projected network communication and validate the proposed methodology. The proposed optimal aggregate link allocation policy is shown to optimize the system throughput and improve link utilization across all levels of the network. Our experiments demonstrate that the proposed optimal bandwidth allocation can effectively obtain the optimal link bandwidth design points for the exploratory direct network and the slim fat-tree network.

References

[1] https://asc.llnl.gov.

[2] http://www.bsc.es.

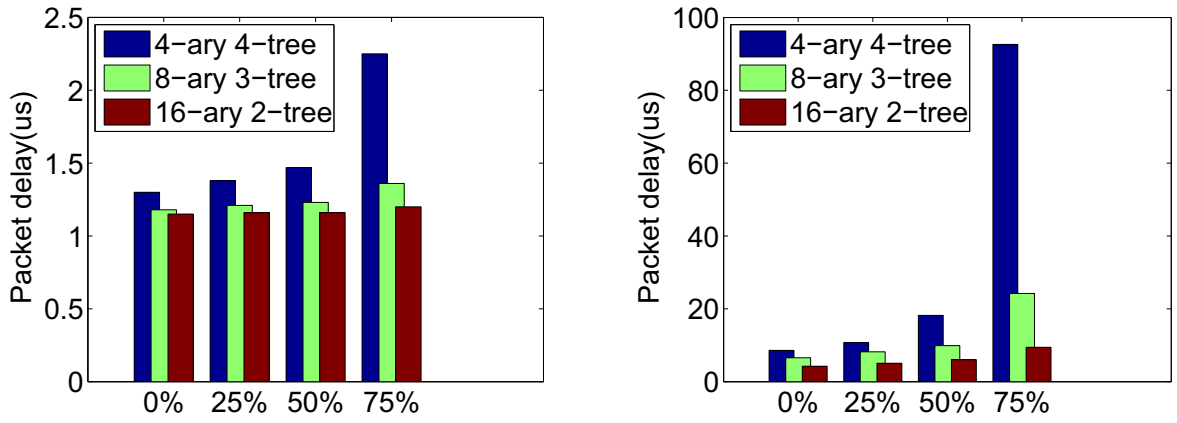
[3] P. Amestoy, I. Duff, J-E L'Excellent, and J. Koster. A Fully Asynchronous Multifrontal Solver Using Distributed Dynamic Scheduling. *SIAM Journal on Matrix Analysis and Applications*, 23(1):15–41, 2002.

[4] R. Badia, J. Labarta, J. Gimenez, and F. Escale. DIMEMAS: Predicting MPI Applications Behavior in Grid Environment. In *Workshop on Grid Applications and Programming Tools (GGF8)*, 2003.

[5] D. H. Bailey and A. Snavely. Performance Modeling: Understanding the Present and Predicting the Future. In *Proceedings of EuroPar*, 2005.

[6] M. W. Berry. Scientific Workload Characterization by Loop-based Analyses. *SIGMETRICS Perform. Eval. Rev.*, 19(3):17–29, 1992.

[7] A. Bogdanov and M. C. Mertens. A Parallel Hardware Architecture for Fast Gaussian Elimination over GF(2). In *Proceedings of the 14th Annual IEEE Symposium on Field-Programmable Custom Computing Machines (FCCM'06)*, pages 237–248, 2006.



(a) Sequential task placement

(b) Random task placement

Fig. 9. Packet delay of UMT2K on various slim fat-tree Networks, having 0%, 25%, 50%, and 75% bandwidth reduction from the proceeding link level.

[8] G. Bolch, S. Greiner, H. Meer, and K. Trivedi. *Queueing Networks and Markov Chains: Modeling and Performance Evaluation With Computer Science Applications*. Wiley, 2006.

[9] R. D. Chamberlain, M. A. Franklin, and C. S. Baw. Gemini: An Optical Interconnection Network for Parallel Processing. *IEEE Trans. Parallel Distrib. Syst.*, 13(10):1038–1055, 2002.

[10] R. Cheveresan, M. Ramsay, C. Feuchtm, and I. Sharapov. Characteristics of Workloads Used in High Performance and Technical Computing. In *Proceedings of the 21st Annual International Conference on Supercomputing*, pages 73–82, 2007.

[11] T. Cormen, C. Leiserson, R. Rivest, and C. Stein. *Introduction to Algorithms*. MIT Press and McGraw-Hill, 2001.

[12] W. Dally and B. Towles. *Interconnection Networks*.

[13] W. Denzel, J. Li, P. Walker, and Y. Jin. A Framework for End-to-end Simulation of High-performance Computing Systems. In *SimulTools*, 2008.

[14] A. Faraj, P. Patarasuk, and X. Yuan. A Study of Process Arrival Patterns for MPI Collective Operations. In *Proceedings of the 21st Annual International Conference on Supercomputing*, pages 168–179, 2007.

[15] E. Grobelny, D. Bueno, I. Troxel, A. George, and J. Vetter. FASE: A Framework for Scalable Performance Prediction of HPC Systems and Applications. *Simulation*, 83(10):721–745, 2007.

[16] C. Iancu and E. Strohmaier. Optimizing Communication Overlap for High-speed Networks. In *Proceedings of the 12th ACM SIGPLAN Symposium on Principles and Practice of Parallel Programming*, pages 35–45, 2007.

[17] V. Kumar, A. Grama, A. Gupta, and G. Karypis. *Introduction to Parallel Computing*. The Benjamin/Cummings Publishing Company, Inc., 1994.

[18] X. Li and J. Demmel. SuperLU-DIST: A Scalable Distributed-memory Sparse Direct Solver for Unsymmetric Linear Systems. *ACM Trans. Math. Softw.*, 29(2):110–140, 2003.

[19] I. Sharapov, R. Kroeger, G. Delamarter, R. Cheveresan, and M. Ramsay. A Case Study in Top-down Performance Estimation for a Large-scale Parallel Application. In *Proceedings of the 11th ACM SIGPLAN Symposium on Principles and Practice of Parallel Programming*, pages 81–89, 2006.

[20] Kai Shen. Parallel Sparse LU Factorization on Second-class Message Passing Platforms. In *Proceedings of the 19th Annual International Conference on Supercomputing*, pages 351–360, 2005.

[21] G. Utrera, J. Corbalan, and J. Labarta. Implementing Malleability on MPI Jobs. In *Proceedings of the 13th International Conference on Parallel Architectures and Compilation Techniques (PACTs)*, pages 215–224, 2004.

[22] J. Vetter. Performance Analysis of Distributed Applications Using Automatic Classification of Communication Inefficiencies. In *Proceedings of the 14th International Conference on Supercomputing*, pages 245–254, 2000.

[23] J. Vetter. Dynamic Statistical Profiling of Communication Activity in Distributed Applications. In *Proceedings of the 2002 ACM SIGMETRICS International Conference on Measurement and Modeling of Computer Systems*, pages 240–250, 2002.