Intelligent HPC Job Scheduling with Communication Pattern Awareness

Xu Yang*, John Jenkins†, Misbah Mubarak†, Robert B. Ross†, Zhiling Lan*
*Department of Computer Science, Illinois Institute of Technology, Chicago, Illinois, USA 60616
xyang56@hawk.iit.edu, lan@iit.edu
†Mathematics and Computer Science Division, Argonne National Laboratory, Argonne, IL, USA 60439
{jenkins,rross}@mcs.anl.gov, mmubarak@anl.gov

***Abstract*—It has been widely recognized that network contension between concurrently running jobs in HPC system is one of the primary causes of performance variability. Optimizing job allocation and avoiding network sharing have been proved to be effective for alleviating network contension and performance degradation. In this work, we propose a new job communication pattern aware allocation strategy which serve as a critical module in our future HPC batch scheduler. The novelty of our new strategy is that it makes allocation decision with not only system topology and resource availability information, but also with job's communication pattern information. The information scheduler gets about job communication pattern can help it to decide whether the jobs performance relies heavily on the network or not, and what the potential inter/intra-job interference would be. Thus, the scheduler can pick the preferable resource allocation based on specific job's communication pattern and preserve the locality of allocation in a better way. Using traces collected from three typical parallel applications of DOE, we validate the effectiveness of our new desgin. The idea presented in this paper is widely applicable: HPC parallel application has specific dominant communication pattern, allocating resource with communication pattern awareness can preserve locality in a bettwer way on systems with differnt network topologies.**

I. Introduction

The scale of supercomputer keeps growing from petascale to exascale, to accomodate the the demand of insatiable computing power from scientific research areas. Production systems contain hundreds of thousands of processors serve as irreplaceable research vehicle for scientific problems with increasing size and complexity. Supercomputers employed in a shared way to accomodate many jobs running concurrently1 , which will also improve system utilization. These jobs share the system infrastructure such as network, I/O bandwidth, which will inevitably cause contention over the shared resource. As supercomputer continues to evolve, these shared resources are tend to become the bottleneck for performance.

One of the most prominant problem comes into researcher's attention is the network contention between concurrently running jobs. The network sharing between concurrently running jobs cause communication variability, which reslut in jobs running slower and longer waiting time for results. The delay of currently running jobs will increase the queueing time of the following submitted jobs, thus leads to low system throughput, low utilization and high energy cost. The network sharing can even cause interference between neighbouring jobs, abort the running job unexpectedly, or even worse, wrong science results.

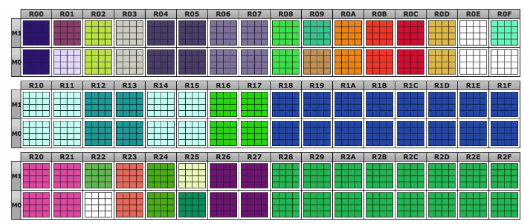

Fig. 1 The activity of Mira, IBM BlueGene/Q system at Argonne National Lab. Many jobs runninng concurrently on the system. Different jobs represented by specific colors.

This adverse effect of network sharing can be mitigated by providing the job with better node allocation. The batch scheduler is responsible for assigning job with the amount of node it required upon submission. Currently two scheduling strategies are commonly used on torus-connected systems. One is so called partition based systems, where the scheduler assigns each user job a compact and contiguous set of computing nodes. IBM Blue Gene series systems fall into this category. This strategy is in favor of applications performance by providing it with isolated partition and exclusive network connections, thus reducing network contention caused by concurrently running jobs sharing network bandwidth. However, this strategy can cause internal fragmentation (when more nodes are allocated to a job than it requests) and external fragmentation(when sufficient nodes are available for a request, but they can not be allocated contiguously), therefore leading to poor system performance (e.g., low system utilization and long waiting time for job). The other is non-contiguous allocation system, where free nodes are assigned to user job no matter whether they are contiguous or not. Cray XT/XE series systems fall into this category. Non-contiguous allocation eliminates internal and external fragmentation as seen in partition-based systems, thereby leading to high system utilization. Nevertheless, it introduces other problems such as scattering application processes all over the system and causing both inter-job contention and intra-job contention. The non-contiguous node allocation can make inter-process communication less efficient and cause network contention among concurrently running jobs, thereby resulting in poor job performance especially for those communication-intensive jobs.

Partition-based allocation achieves good job performance by sacrificing system performance (e.g., poor system utilization), whereas non-contiguous allocation can result in better system performance but could severely degrade performance of user jobs (e.g, prolonged wait-time and run-time). Supercomputers keep growing in size, its network become more sophisticated then ever before and applications also increase its complexity. How to provide application with better allocation that can balance job performance with system performance become a prominant problem need a solution before HPC enters era of exasacle computing. In this work, we propose a guideline for future batch scheduling system, a smart way to make allocation with the awareness of job's communication pattern.

Most existing batch schedulers take so few initiative because it been hold under the obscurantism policy. Whether it use partition-based or non-contigous policy, means it chooses the benefit of either system or job over the other. Some propose compromised policies like buddy system[**?**] still make the allocation for job blindly without knowing job's communication information.

Although HPC applications complexity continue to grow, people could get more knowledge about the applications communication information with the help from various parallel application profiling tools, such as TAU[12], MPIP[11] and DUMPI[34]. Job's communication trace and performance analysis can be get by using such tool at runtime.

The off-line study of job's communication traces could be helpful to develop better allocation policies for batch scheduler to make on-line scheduling/allocating decisions. This is because the communication pattern of parallel applications on HPC system can be summarized into a couple of categories[29]. The deep understanding of one application can be helpful to all other applications that conforms to the same communication pattern. Job's communication traces study would be extreme helpful to leadership computing facility too. The set of applications running on those leadership computing facilities is usually stable, and the workloads from those HPC systems is usually has great repetitiveness. We analyzed the workload on Mira, an IBM Blue/Gene Q system at Argonne Leadership Computing Facility, found that most jobs require 2k nodes and about 14 particular jobs consist of the majority its workload**??**. In other word, Mira accommodates mostly some particular 14 jobs and most jobs have the same size.

In this work, we present a guideline for future batch scheduler to make allocation with awareness about job's communication pattern. Rather than choose blindly between partition-base or non-contigous allocation, the new allocation mechanism make allocation based on job's communication pattern. With detail analysis about application's communication pattern, we can get a clear view of its communication topology graph, data transfer between processes, and decide which processes are tightly connected to which and conduct intensive communication. The "local community" consists of such tightly coupled processes shuld get compact alloction. Then, the allocation mechanism should make allocations that only needs to guarantee the locality of the subset of the nodes where those "local community" should reside on.

We first pick three signature applications from DOE Design Forward Project[14] as examples to conduct deep study about their communication patterns. We identify the "local community" in these applications and assign them allocation accrodingly. The advantage of making allocation is obvious. First, compared with partition-based policy, our communication pattern aware policy is more flexible, it doesn't require the system to provide a big partition that can accomodate the whole application, just small set of compact nodes enough for all the "local community" in the application. On the other hand, our communication pattern aware allocation policy won't destroy the locality of application's communication pattern. On the contrary, it provides small compact node sest for the the "local communities" in the application will preserve the communication locality in the maximum extent.

We use a sophisticated simulation toolkit named CODES, Co-Design of Multi-layer Exascale Storage Architecture, from Argonne National Lab as research vehicle to evaluate the performance of our communication pattern aware allocation policy on torus network. CODES enables the exploration of simulating different HPC networks with high fidelity[31]. We extend CODES with a Job Mapping API that support different allocation policies on torus network.

We also conduct a case study with the results we got from communication pattern aware allocation policy evaluation with real trace from production system.Not Sure About this part!!!

The rest of this paper orgainzed as follows. SectionII gives a detailed study of the three applications from DOE Desgin Forward Project. SectionIII talks about the simulation platform and Section

II. Application Study

A parallel application usuall conforms to a combination of several basic communication patterns. At its different execution phases, the application's communication behavior may follow different basic patterns repectively. When we look into the data flow during the application execution, most parallel application start with broadcast operation to distribute the data from root process to other processes, followed by a series of computation and communication that comforms to certain pattern, which is usually the dominant part of application's execution. And before the application come to completion, all the working process will return their results to the root dirctly or hierarchically. In this work, we focus on the dominant communication pattern because that's where the application spend most of its running time. We believe making optimized allocation regarding application's dominant communication pattern would greatly benefit both the application and the system. In this section, we analysis two applications whose dominant communication patterns are most prevalent in scientific computation.

There are many profiling tools available to capture the communication behavior from parallel applications, such as Tuning and Analysis Utilities(TAU)[12], mpiP[11]. They can help analysis parallel application, providing information like the percentage of different MPI operations of the applications, communication topology, the amount data transferred between processes.

In this work, we provide detailed analysis about the communication pattern of two DOE MiniApps. MiniApps are reduced proxy applications that encapsulate the salient performance of larger full size applications[15]. Since we only focus on the dominant communication pattern of parallel applications, these MiniApps are perfect candidates for analysis the application's performance on different allocations or on new network architectures.

A. *AMG*

AMG is the MiniApp of Algebraic Multigrid Solver, which is a parallel algebraic multigrid solver for linear systems arising from problems on unstructured mesh physics packages. It has been derived directly from the BoomerAMG solver that is being developed in the Center for Applied Scientific Computing (CASC) at LLNL[16]. The dominant communication pattern is this regional communication with decreasing message size for different parts of the multigrid v-cycle.

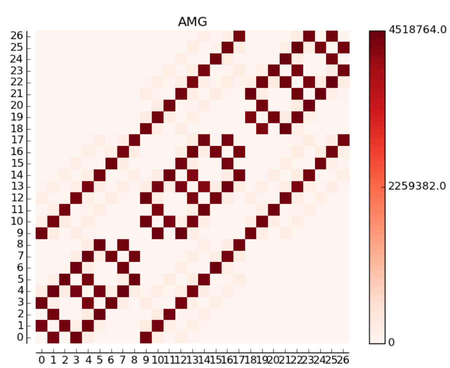

Fig. 2 Communication Topology of AMG with 27 MPI Ranks

Figure 2 shows the communication topology of a small scale AMG with 27 MPI Ranks. Here we prefer to show the communication topology with small scale version of the MiniApp because the dominant communication pattern of the application doesn't change with its scale. We can make the observation from the upper-left of the figure that each rank in AMG has intense communication between 3 other ranks, such as rank 0 between rank 1, 3 and 9, rank 1 between 2, 4 and 10, etc. And this relation is also symmetric along the main diagnal. The dominant communication pattern is quite obvious when we identify this relation between ranks.

Figure 3 shows the amount of data each rank transferred in AMG. The blue line shows that total data amount transferred from each rank, while the red shows the amount of received data and green shows the amount of send data. The first thing we can spot from the figure is that for each rank, the amount of data sent and received are basically the same, which also well explain the symmetry of the application's dominant communication pattern. And the second thing we can find about AMG is that the data transfer amount also has a symmetric pattern. Rank 0 has the same amout of data transfer as Rank 26, Rank 1 as of Rank 25, etc.

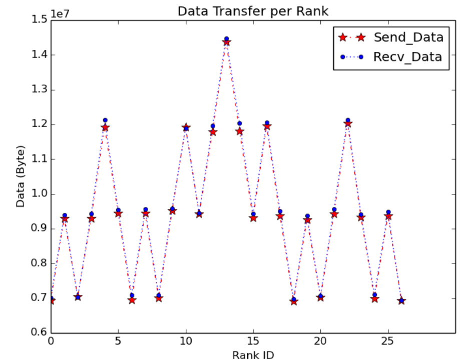

Fig. 3 The amount of data each rank transferred in AMG.

B. *Crystal Router*

The second MiniApp we have is Crystal Router, which is the extracted communication kernel of the full application Nek5000. Nek5000[17] is a spectral element CFD application developed at Argonne National Laboratory. It features spectral element multigrid solvers coupled with a highly scalable, parallel coarse-grid solver that widely used for projects including ocean current modeling, thermal hydraulics of reactor cores, and spatiotemporal chaos. The MiniApp of Nek5000, Crystal Router demonstrates the "many-to-many" communication pattern through scalable multi-stage communication process.

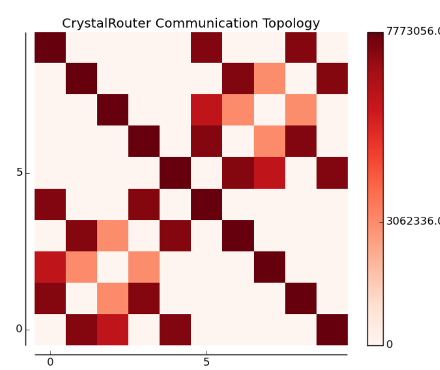

Fig. 4 Communication Topology of Crystal Router with 10 MPI Ranks

Here, we provide the communication topology graph of Crystal Router with 10-rank and 100-rank respectively. As we can see from figure 4, every 5-rank is a compact community, where data transfer is very intensive. Beside the "5-

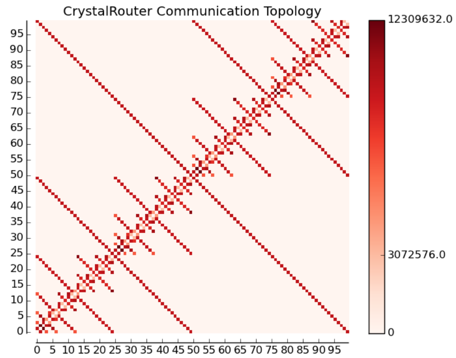

Fig. 5 Communication Topology of Crystal Router with 100 MPI Ranks

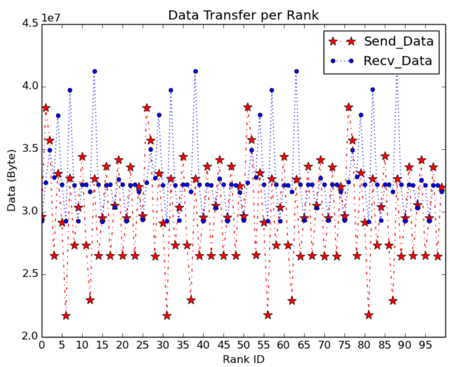

Fig. 6 The amount of data each rank transferred in Crystal Router.

rank community", there is "pair-wise" global data transfer in Crystal Router. Obviously, there is data transfer alone the diagnal. When the scale of Crystal Router increase to 100, the "5-rank community" is still the dominant pattern in a multi-stage manner. The same pattern can be observed from the global data transfer. Our observation about the dominant communication pattern of Crystal Router is the multi-stage "5-rank community" and "pair-wise" global data transfer happen recursively as the scale of the application increase.

Figure 8 shows the amount of data each rank transferred in Crystal Router. The locality pattern of "5-rank community" can still be spotted in this figure. The spiky shape occurs in the frequency of every 5 rank. Unlike AMG, the there is variance between the amount of data send and received for each rank. This is due to the specific feature of Crystal Router's communication pattern.

C. MultiGrid

Differe from AMG, MultiGrid is geometric multigrid v-cycle from production elliptic solver BoxLib, a software framework for massively parallel block-structured adaptive mesh refinement (AMR) codes. MultiGrid conforms to 3D lattice communication pattern with decreasing message size and collectives for different parts of the multigrid v-cycle. It is widely used for structured grid physics packages.

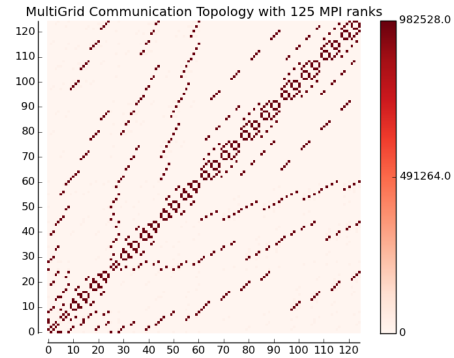

Fig. 7 Communication Topology of MultiGrid with 125 MPI Ranks

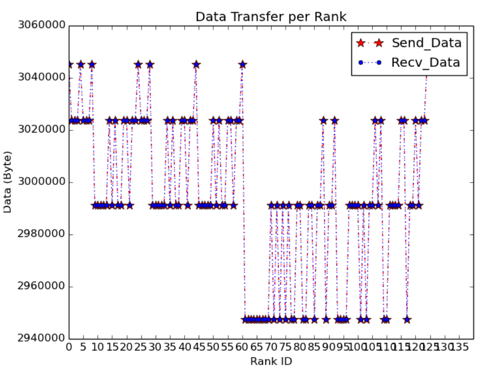

Fig. 8 The amount of data each rank transferred in MultiGrid.

III. Simulation Platform

A simulation toolkit named CODES (Co-Design of Multi-layer Exascale Storage Architecture) from Argonne National Lab enables the exploration of simulating different HPC networks with high fidelity[31][33]. CODES is built on top of Rensselaer Optimistic Simulation System (ROSS) parallel discrete-event simulator, which is capable of processing billions of events per second on leadership-class supercomputers[32]. CODES support both torus and dragonfly network with high fidelity flit-level simulation. In this work, we only use torus network to show the impact of node allocation to job with different dominant communication patterns. CODES has this network workload component that capable of conducting trace-driven simulation. It can take real MPI application trace generated by SST DUMPI[34] to drive CODES network models.

Torus networks have been extensively used in the current generation of supercomputers because of their linear scaling on per-node cost and competitive communication performance. The topology of torus network is k-ary n-cube, with k^n nodes in total arranged in an n-dimensional grid having k nodes in each dimension. Each node has 2× n direct linked neighbor nodes.

For a job requires n nodes submitted to non-partition based 3D torus-connected system, it gets an allocation of various possible shapes, which could be 3D balanced-cube, 3D unbalanced-cube, 2D mesh, or even 1D list. Figure 9 shows some possible allocations for a 64-node job. 3D balanced-cube allocation (in Figure 9 as red) is $\{4,4,4\}$, 3D unbalanced-cube allocation is $\{8,4,2\}$(in Figure 9 as green), 2D mesh allocation is $\{8,8,1\}$ (in Figure 9 as blue).

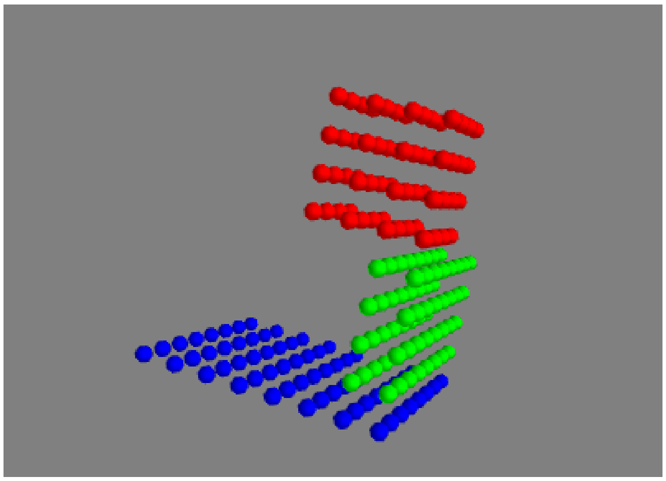

Fig. 9 64-node job with different allocation shapes. Red, Green and Blue represents 3D balanced, 3D unbalanced and 2D allocation respectively.

We implemented a Job Mapping API for CODES, which can provide flexible job allocation strategies. The new API can help us to explore the impact of different allocation strategies to job with specific communication pattern. We have choosen two publicly available HPC network traces provided by the Design Forward Program[14]. The detailed analysis of these two traces are present in SectionII-A and II-B.

A. Allocation Simulation

In this section, we analysis the impact of different allocation strategies on two real applications, AMG and Grystal Router. The torus network performance is determined by its dimensionality and link bandwidth. We evaluate the data transfer time of AMG and Crystal Router on torus network with different dimensionality and link bandwidth configurations in section III-B. Then, we will show applications performance on 3D torus network with different allocation strategies in section IV-B.

B. Torus Dimensionality

As the increase of the dimensionality of torus netowrk, so does the number of links connected with each node. The increased aggregated bandwidth of each node will definitely reduce the data transfer time of each rank in the application. Figure xxx shows the performance of the AMG and CrystalRouter on a 2K node torus network model with a 3D torus (16x16x8), a 5D torus (8x4x4x4x4), and a 7D torus (4x4x4x4x2x2x2). The bandwidth between nodes is 2GiB/s in one direction, thus, the aggregated bandwidth is 12GiB/s per node in 3D torus, 20GiB/s per node in 5D torus, and 28GiB/s per node in 7D torus. As we can see from the figure, communication time of both applications decrease as the dimensionality of the network increase. The aggregated bandwidth of each node can accelerate the transfer of data. This can justify the fidelity and consistency of the torus network model.

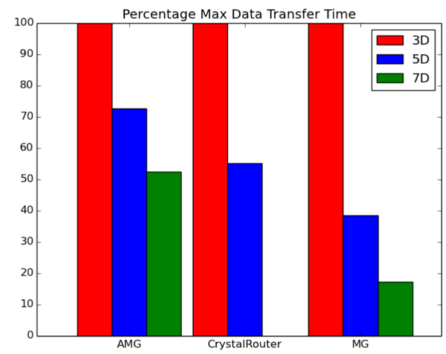

Fig. 10 The Percentage of Maximum Time spent to send data over 3D, 5D and 7D torus networks by AMG1728, CrystalRouter1000, MultiGrid1000.

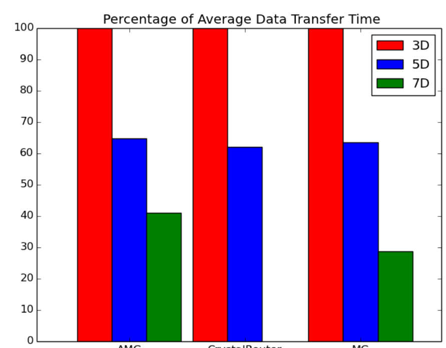

Fig. 11 The Percentage of Average Time spent to send data over 3D, 5D and 7D torus networks by AMG1728, CrystalRouter1000, MultiGrid1000.

cr1000 on 7D result is missing, will be added soon. We can also provide Per-Rank time spent for data transfer. not sure if necessary.

IV. Communication Pattern Aware Allocation Policy

In this section, we will explore the performance of three applications on different allocations in a 3D torus network. First, we analysis the impact from different allocation shape, then we will present the results by using our communication aware allocation policy.

A. Different Allocation Shape

As we shown in Fig9, there could be three differet allocation shape on 3D torus network, namely 3D balanced, 3D unbalanced, and 2D. In this section, we will show the performance of three applications on these three different shape allocations.

There are lots of research work try to design fancy allocation algorithms to provide application with most compact allocation, like using space filling curve (SFC) to index high dimentional cube into one dimensional list. Allocation by cutting chunks of nodes from this one dimensional list will guarantee cubical allocation. However, try to provide cubical allocation without considering application's communication pattern could cause backfire.

Detail explanation will be add for Fig12-Fig16.

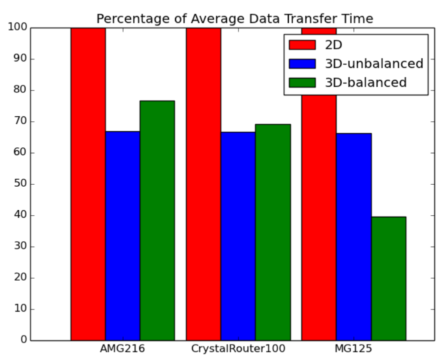

Fig. 12 The Percentage of Average Time spent to send data by AMG216, CrystalRouter100, MultiGrid125 with 2D, 3D-unbalanced, 3D-balanced allocation.

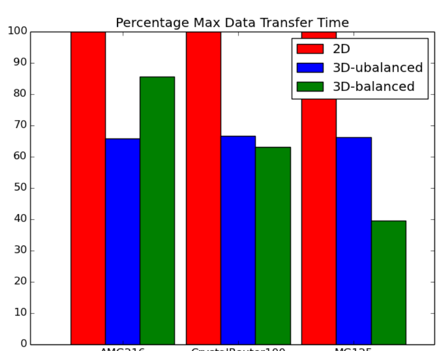

Fig. 13 The Percentage of Max Time spent to send data by AMG216, CrystalRouter100, MultiGrid125 with 2D, 3D-unbalanced, 3D-balanced allocation.

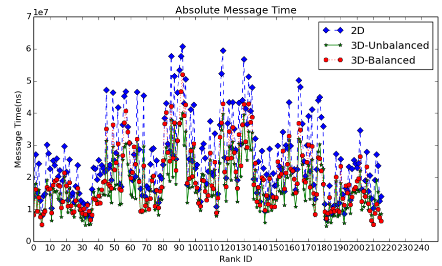

Fig. 14 (AMG216) Time spent by each rank to send data on 2D, 3D-unbalanced, 3D-balanced allocation.

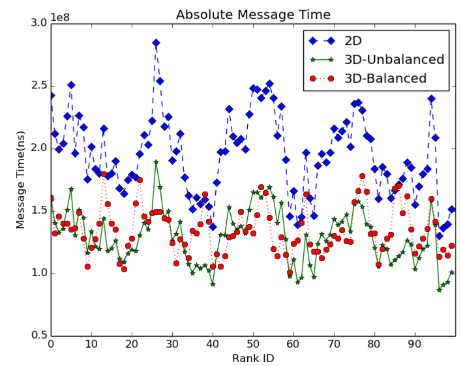

Fig. 15 (CrystalRouter100) Time spent by each rank to send data on on 2D, 3D-unbalanced, 3D-balanced allocation.

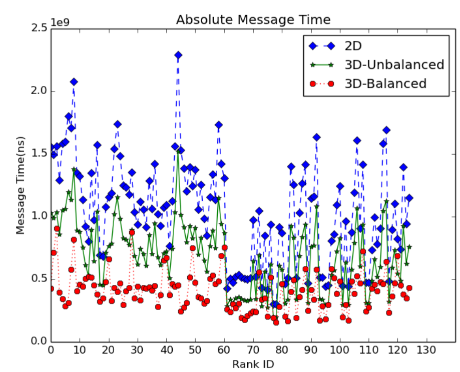

Fig. 16 (MultiGrid125) Time spent by each rank to send data on 2D, 3D-unbalanced, 3D-balanced allocation.

B. Allocation Strategies Study

We proposed two allocation strategies, Compact and Chunk to study the impact of different allocation the application's performance. The Compact allocation always provides application with a cubical allocation on 3D torus, regardless application's communication pattern. This cubical allocation will keep the locality of the application, all the processes reside on it will have the shortest pair-wise distance.

The Chunk allocation simulation the scenario of non-contiguous allocation strategy employed in production system like Cray XT machines. The traditional non-contiguous allocation simply picks available nodes in spite of their locations and assign them to application. The motivation of non-contiguous allocation act like this is to maximize system utilization, while being blind to application's communication pattern would greatly prolong application's communication time. Although Chunk allocation provides application with non-contigous allocation too, it assign the chunk of nodes in consideration of application's communication pattern. For example, we know the dominant communication pattern of CrystalRouter in section II-B, every 5 rank form a community, where intensive communication happens between them. With such knowledge knowledge about application's communication pattern, the Chunk allocation will provide CrystalRouter with allocation that consists of small chunks, the size of which will be at least 5 nodes.

Figure xxx shows the performance of AMG and CrystalRouter on 3D torus with Compact and Chunk allocation.

V. Experiment Results

VI. Related Work

Batch scheduling on HPC system has been a hot topic getting constant discussion since the 1990's. Feitelson et al proposed the most widely used job scheduling policy which is First Come, First Serve (FCFS) combined with EASY backfilling[1]. FCFS/EASY backfilling policy greatly improves HPC system's utilization and cuts the wait time of first job in the waiting queue. Since then, many studies seek to refine this classic scheduling paradigm. Tang et al. tried to improve the accuracy of user's estimated job runtime and walltime so as to enhance the efficiency of backfilling and minimize the system fragmentation[2] [4] [3].

Only with the information about job size and estimated run time seems to be insufficient for batch scheduler do make the optimal scheduling/allocaiton decision. Some work proposed that the batch scheduler should be aware of some information about system network information in order to fully utilize system resources.Leung et al. presented allocation strategies based on space filling curves and one dimensional packing [8]. The problem with using space filling curve is that it can only be applied to system with the scale of 2^n nodes in each dimension, which is a ideal case that not existing in morden HPC systems. Albing et al. conducted study about the allocation strategies that the Cray Application Level Placement Scheduler (ALPS) used [9]. The job allocation in Cray Linux Environment (CLE) operating system is managed by ALPS, which simply works off an ordered node list, however that is ordered.

Based on their work, Yang et al. presented a window-based locality-aware job scheduling design for torus-connected system [10]. In this new scheduling framework, several jobs are taken into consideration at the same time for job prioritizing and resource allocation. A list of slots is maintained to preserve node contiguity information for resource allocation. Each job been assigned into one of these slot so that the locality can be mostly preserved.

Rosenthal et al. found there is limited benefit for many applications by increasing network bandwidth [18]. They employed a subset of the CORAL mini-applications that represent U.S. Department of Energy workload and leverage multirail networkings to evaluate the improvement of these applications caused by increased network bandwidth. The applications that send mostly small messages or larger messages asynchronously are not bandwidth bounded, hence, benefit only slightly or not at all from increased bandwidth.

Hoefler et al. propose to use performance modeling techniques to analysis factors that impact the performance of parallel scientific applications [19]. However, as the scale of HPC continue grows, the interference of concurrently running jobs is getting worse, which is hard to be quantified by performance profiling tools.

Bogdan et al provide a set of guidelines how to configure a network with Dragonfly topology for workload with Nearest Neighbor communication pattern[28]. They first derived a theoretical model of Nearest Neighbor communication performance on Dragonfly network that can predict network bottleneck. Then, they use simulation to validate the correctness of their theoretical model.

Dong et al describe IBM Blue Gene/Q's 5D torus interconnect network[30]. They developed simple benchmarks that conforms to four different communication patterns, namely ping-pong, nearest neighbor, broadcast and allredcue, to demonstrate the effectiveness of this highly parallel 5D torus network.

VII. Conclusions

Acknowledgment

The work at Illinois Institute of Technology is supported in part by US National Science Foundation grant CNS-1320125. The work at Argonne is supported in part by the U.S. Department of Energy (DOE), Office of Science, under Contract DE-AC02-06CH11357.

References

[1] D. Feitelson and A. Weil. Utilization and predictability in scheduling the IBM SP2 with backfilling. In *International Parallel and Distributed Processing Symposium*, 1998.

[2] W. Tang, N. Desai, D. Buettner, and Z. Lan. Analyzing and adjusting user runtime estimates to improve job scheduling on the Blue Gene/P. In *2010 IEEE International Symposium on Parallel Distributed Processing*, 2010.

[3] W. Tang, Z. Lan, N. Desai, D. Buettner, and Y. Yu. Reducing fragmentation on torus-connected supercomputers. In *2011 IEEE International Symposium on Parallel Distributed Processing Symposium*, 2011.

[4] W. Tang, N. Desai, D. Buettner, and Z. Lan. Job Scheduling With Adjusted Runtime Estimates on Production Supercomputers. Journal of Parallel and Distributed Computing (JPDC), 73(7):926-938, 2013.

[5] Z. Zhou, Z. Lan, W. Tang, and N. Desai. Reducing energy costs for IBM Blue Gene/P via Power-Aware job scheduling. In *17th Workshop on Job Scheduling Strategies for Parallel Processing*, 2013.

[6] X. Yang, Z. Zhou, S. Wallace Z. Lan, W. Tang, S. Coghlan and Mike. Papka. Integrating Dynamic Pricing of Electricity into Energy Aware Scheduling for HPC Systems. In Proceedings of the International Conference on High Performance Computing, Networking, Storage and Analysis(SC), 2013 International Conference for SC '13, 2013, Salt Lake City, Utah

[7] Cqsim: An event-driven simulator. http://bluesky.cs.iit.edu/cqsim

[8] Leung, V.J.; Arkin, E.M.; Bender, M.A.; Bunde, D.; Johnston, J.; Alok Lal; Mitchell, J.S.B.; Phillips, C.; Seiden, S.S., Processor allocation on Cplant: achieving general processor locality using one-dimensional allocation strategies In Proceedings of 2002 IEEE International Conference on Cluster Computing, 2002. pages 296–304, 2002

[9] Carl Albing and Mark Baker ALPS,Topology, and Performance: A Comparison of Linear Orderings for Application Placement in a 3D torus. Presented at the CUG 2010, Edinburgh, Scotland, UK, 2010.

[10] Xu Yang and Zhou Zhou and Wei Tang and Xingwu Zheng and Jia Wang and Zhiling Lan. Cluster Computing (CLUSTER), 2014 IEEE International Conference on, Balancing job performance with system performance via locality-aware scheduling on torus-connected systems Sept, 2014, pp.140-148

[11] mpiP: Lightweight, Scalable MPI Profiling. http://mpip.sourceforge.net, 2013.

[12] S. Shende and A. Malony. The TAU parallel performance system. International Journal of High Performance Computing Applications, 20(2):287311, 2006.

[13] X. Wu, F. Mueller, S. Pakin, "Automatic Generation of Executable Communication Specifications from Parallel Applications", In Proceedings of the International Conference on High Performance Computing, Networking, Storage and Analysis(SC), 2011 International Conference for SC '11, 2011,pages 12-21.

[14] Department of Energy, Characterization of the DOE Mini-apps. [Online]. Available: http://portal.nersc.gov/project/CAL/trace.htm (Accessed on: Sept 16th. 2015)

[15] Bronson Messer "Using a Developing MiniApp to Compare Platform Characteristics on Cray Systems" In Proceedings of Cray User Group 2012

[16] V. E. Henson and U. M. Yang, "BoomerAMG: A Parallel Algebraic Multigrid Solver and Preconditioner", Appl. Num. Math. 41 (2002), pp. 155-177. UCRL-JC-141495.

[17] P. Fischer, J. Lottes, D. Pointer, and A. Siegel, Petascale algorithms for reactor hydrodynamics, Journal of Physics: Conference Series, vol. 125, no. 1, p. 012076, 2008.

[18] E. Rosenthal and E.A. Leon Characterizing Application Sensitivity to Network Performance. In Proceedings of SC14: International Conference for High Performance Computing, Networking, Storage and Analysis, 2014

[19] T. Hoefler and W. Gropp and M. Snir and W. Kramer Performance Modeling for Systematic Performance Tuning, 2011 Nov. In Proceedings of International Conference for High Performance Computing, Networking, Storage and Analysis (SC'11), SotP Session,

[20] P. Roth and J. Meredith and J. Vetter, "Automated Characterization of Parallel Application Communication Patterns", In Proceedings of the 25rd International Symposium on High-performance Parallel and Distributed Computing HPDC '15, 2015,

[21] Pješivac-Grbović, Jelena and Angskun, Thara and Bosilca, George and Fagg, Graham E. and Gabriel, Edgar and Dongarra, Jack J. Performance Analysis of MPI Collective Operations, Cluster Computing, June 2007, volume 10, numpages 17,

[22] Thakur, Rajeev and Gropp, William D. Improving the performance of collective operations in MPICH, Recent Advances in Parallel Virtual Machine and Message Passing Interface, year 2003, Springer

[23] Abhinav Bhatele, Andrew R. Titus, Jayaraman J. Thiagarajan, Nikhil Jain, Todd Gamblin, Peer-Timo Bremer, Martin Schulz and Laxmikant V. Kale, "Identifying the Culprits behind Network Congestion", In *2015 IEEE International Symposium on Parallel Distributed Processing Symposium*, 2015.

[24] S. Langer, A. Bhatele, and C. H. Still, pF3D simulations of laser-plasma interactions in National Ignition Facility experiments, Computing in Science and Engineering, vol. 99, Aug. 2014, lLNL-JRNL-648736. [Online]. Available: http://doi.ieeecomputersociety.org/10.1109/MCSE.2014.79

[25] Prisacari, Bogdan and Rodriguez, German and Heidelberger, Philip and Chen, Dong and Minkenberg, Cyriel and Hoefler, Torsten Efficient Task Placement and Routing of Nearest Neighbor Exchanges in Dragonfly Networks In Proceedings of the 23rd International Symposium on High-performance Parallel and Distributed Computing HPDC '14, 2014, Vancouver, BC, Canada

[26] Wu, Jingjin and Lan, Zhiling and Xiong, Xuanxing and Gnedin, Nickolay Y. and Kravtsov, Andrey V. Hierarchical Task Mapping of Cell-based AMR Cosmology Simulations Proceedings of the International Conference on High Performance Computing, Networking, Storage and Analysis SC '12, 2012, Salt Lake City, Utah

[27] Z. Zhou, X. Yang, Z. Lan, P. Rich, W. Tang, V. Morozov, and N. Desai. Improving Batch Scheduling on Blue Gene/Q by Relaxing 5D Torus Network Allocation Constraints, Proceedings of 29th IEEE International Parallel & Distributed Processing Symposium, IPDPS'15, 2015. Hyderabad, INDIA

[28] Prisacari, Bogdan and Rodriguez, German and Heidelberger, Philip and Chen, Dong and Minkenberg, Cyriel and Hoefler, Torsten Efficient Task Placement and Routing of Nearest Neighbor Exchanges in Dragonfly Networks, Proceedings of the 23rd International Symposium on High-performance Parallel and Distributed Computing, HPDC '14, 2014, Vancouver, BC, Canada,

[29] Philip. C.R, Jeremy S. M, etc "Automated Characterization of Parallel Application Communication Patterns" HPDC'15

[30] Dong Chen; Eisley, N.A.; Heidelberger, P.; Senger, R.M.; Sugawara, Y.; Kumar, S.; Salapura, V.; Satterfield, D.L.; Steinmacher-Burow, B.; Parker, J.J., "The IBM Blue Gene/Q interconnection network and message unit," High Performance Computing, Networking, Storage and Analysis (SC), 2011 International Conference, pp.1,10, 12-18 Nov. 2011

[31] Jason Cope, Ning Liu, Sam Lang, Phil Carns, Chris Carothers, Robert Ross. Proceedings of the Workshop on Emerging Supercomputing Technologies 2011, 2011

[32] P. D. Barnes, C. D. Carothers, D. R. Jefferson, and J. M. LaPre, Warp speed: executing time warp on 1,966,080 cores, in Proc. of the 2013 ACM SIGSIM Conf. on Principles of Advanced Discrete Simulation (PADS), May 2013, pp. 327336.

[33] Mubarak, M.; Carothers, C.D.; Ross, R.; Carns, P., "Modeling a Million-Node Dragonfly Network Using Massively Parallel Discrete-Event Simulation," High Performance Computing, Networking, Storage and Analysis (SCC), 2012 SC Companion: pp.366,376, 10-16 Nov. 2012

[34] Sandia National Labs, SST DUMPI trace library. [Online]. Available: http://sst.sandia.gov/using dumpi.html (Accessed on: Sept 3rd, 2015)