論文 / 著書情報
Article / Book Information

題目(和文)	HPCシステムのデータ移送による資源競合
Title(English)	Resource Contention due to Data Movement on HPC Systems
著者(和文)	BROWNKEVIN
Author(English)	Kevin A. Brown
出典(和文)	学位:博士(学術), 学位授与機関:東京工業大学, 報告番号:甲第11009号, 授与年月日:2018年9月20日, 学位の種別:課程博士, 審査員:松岡 聡,増原 英彦,遠藤 敏夫,脇田 建,額田 彰
Citation(English)	Degree:Doctor (Academic), Conferring organization: Tokyo Institute of Technology, Report number:甲第11009号, Conferred date:2018/9/20, Degree Type:Course doctor, Examiner:,,,,
学位種別(和文)	博士論文
Category(English)	Doctoral Thesis
種別(和文)	論文要旨
Type(English)	Summary

（博士課程）
Doctoral Program

論 文 要 旨

THESIS SUMMARY

専攻： Department of	数理・計算科学　専攻	申請学位（専攻分野）： Academic Degree Requested	博士 Doctor of (Philosophy)
学生氏名： Student's Name	Kevin Antoney BROWN	指導教員（主）： Academic Supervisor(main)	特任教授　松岡　聡
		指導教員（副）： Academic Supervisor(sub)	

要旨（英文 800 語程度）
Thesis Summary (approx.800 English Words)

The trend in the field of high-performance computing is that supercomputers increase in performance capabilities with each successive generation. Top systems have had their compute performance increase by two orders of magnitude each generations, while their network bandwidths have only increased by one order of magnitude over three generations. The fact that the (i) size of HPC systems are increasing, (ii) workloads sizes are becoming larger due to the HPC-Big Data convergence et al., and (iii) that compute capabilities are improving at a much higher rate than communication capabilities, various applications are becoming more transfer-bound than before.

Another factor that affects communication performance is the sharing of network resources among processes in the same application as well as sharing among different applications. Different workloads (namely MPI and I/O) will have different traffic patterns because their data exchange requirements differ. The degree by which the resulting interference will affect communication performance requires analysis of application communication patterns and their interference patterns.

The scale and complexity of network architectures, such as InfiniBand-based fat-tree networks, and the abstractions of communication libraries, such as MPI, pose challenges to conducting efficient performance analysis. In the absence of faster network hardware, effective measurement and analysis of communication performance is crucial for overcoming these limitations. Non-portable data extraction methods and unwieldy tool-chains are typically needed in order to understand application performance bottleneck over network links, the point of intra- and inter-application resource contention.

This work introduces an approach to conducting performance analysis of MPI applications running on large-scale networks that efficiently and portably exposes performance metrics from within MPI the layer. The collection and analysis of these metrics provide unique insights into possible bottlenecks occurring in the hardware layer during MPI operations, including collectives. The results demonstrate that the existing Peruse interface in Open MPI can be extended to expose metrics from within the MPI layer in a portable manner. This portability is achieved by the fact that the Peruse interface is available in all Open MPI implementations. The creation of the **ibprof** profiler further showed that the collection of low level metrics can be achieved in a very non-intrusive manner and with minimal overhead. The average communication overhead incurred was 2.06% for the MPI_Alltoall microbenchmark, 7.63% for the MPI_Bcast microbenchmark, and 0.02% for the NPB FT application benchmark. Additionally, the time taken to write the communication profile to OTF files was less than 1 second for each benchmark.

To achieve meaningful presentation of the communication profiles reported by **ibprof**, a hardware-centric performance visualization for fat-tree networks was developed in Boxfish. This generalized visualization module can be used to graphically represent any 2-dimensional network topology. The visualization method leverages the features of Boxfish to provide user-friendly, flexible, extensible, and scalable presentation of performance data from multiple domains, such as the hardware, the communication, and the application domains. Case studies demonstrate how this analysis approach can identify communication anomalies in applications as well as communication libraries and guide performance optimization strategies.

Another important contribution of this work is a characterization of the interference between the MPI and I/O traffic on fat-tree networks. The characterization study was done using the packet-level accurate network simulations and highlights the impact of varying the message size, communication interval, and job size. The results show that MPI traffic is more sensitive to interference than I/O traffic in all cases. Light I/O traffic exhibited a 1.9X slowdown due to heavy MPI interference, while light MPI traffic exhibited a 7.6X slowdown due to heavy I/O traffic. Furthermore, a significant feature of I/O traffic patterns was identified and labeled as the *I/O-congestion threshold*. This threshold is the I/O interval, or frequency, of sending I/O requests where the self-congestion from I/O is more detrimental than the contention due to MPI interference. This threshold varies with the I/O request size and the job size.

Finally, an evaluation of the efficacy of three communication optimization strategies is presented. These strategies are targeted at mitigating the I/O-MPI interference on fat-tree networks by taking advantage of network topology-specific features and the characteristics of the interference. The node allocation strategy and the I/O server placement strategy are shown to, independently, completely avoid I/O-MPI interference. That is, careful placement of jobs on the network and the physical location of I/O servers can isolate I/O traffic from MPI traffic on the network. The third mitigation strategy, throttling, is shown to reduce slowdown of MPI performance by approximately 200% while incurring only an 18% performance penalty. This strategy is guided by the I/O-congestion threshold of the I/O workload.