<table>
<thead>
<tr>
<th>Title</th>
<th>Co-locating Graph Analytics and HPC Applications</th>
</tr>
</thead>
<tbody>
<tr>
<td>Authors</td>
<td>Kevin Brown, Satoshi Matsuoka</td>
</tr>
<tr>
<td>Citation</td>
<td>2017 IEEE International Conference on Cluster Computing (CLUSTER), pp. 659-660</td>
</tr>
<tr>
<td>Pub. date</td>
<td>2017, 9</td>
</tr>
<tr>
<td>Copyright</td>
<td>© 2017 IEEE. Personal use of this material is permitted. Permission from IEEE must be obtained for all other uses, in any current or future media, including reprinting/republishing this material for advertising or promotional purposes, creating new collective works, for resale or redistribution to servers or lists, or reuse of any copyrighted component of this work in other works.</td>
</tr>
<tr>
<td>DOI</td>
<td><a href="http://dx.doi.org/10.1109/CLUSTER.2017.111">http://dx.doi.org/10.1109/CLUSTER.2017.111</a></td>
</tr>
<tr>
<td>Note</td>
<td>This file is author (final) version.</td>
</tr>
</tbody>
</table>

Powered by T2R2 (Tokyo Institute Research Repository)
Co-locating Graph Analytics and HPC Applications

Kevin Brown
Department of Mathematical and Computing Sciences
Tokyo Institute of Technology
2-12-1 Oo-okayama, Meguro-ku, Tokyo 152-8550, Japan
Email: brown.k.aa@m.titech.ac.jp

Satoshi Matsuoka
Global Scientific Information and Computing Center
Tokyo Institute of Technology
2-12-1 Oo-okayama, Meguro-ku, Tokyo 152-8550, Japan
Email: matsu@is.titech.ac.jp

Abstract—We evaluate the on-node interference caused when co-locating traditional high-performance computing applications with a big-data application. Using kernel benchmarks from the NPB suite and a state-of-art graph analytics code, we explore different process placements and effects they have on application performance. Our results show that the most memory intensive HPC application (MG) experienced the highest performance variation during co-location.

I. INTRODUCTION

The high-performance computing (HPC) systems at large research centers typically serve users from a variety of disciplines with varying resource requirements. In systems that employ conventional node-exclusive resource allocation schemes, nodes are over-provisioned with resources to meet the varying needs of all users. However, this generalized node design results in inefficient resource utilizations; users do not use all of the resources on their assigned nodes, leading to system-wide resource fragmentation and wastage [1]. Resource sharing at the node level is therefore required to ensure good occupancy and achieve high system-wide efficiency [1], [2].

Co-locating multiple applications on the same system can result in significant degrees of performance variation. For example, the potential for inter-application contention over network resources is increased with application node-sharing since communication endpoints are now being shared. Furthermore, cache, memory, local storage devices, and other on-node resources become points of inter-application interference. These on-node resources are not optimized for parallel access and are therefore susceptible to causing major performance bottlenecks.

Researchers have started investigating the effects of on-node interference due to node-sharing on HPC systems [3]. However, recent trends in the workload on HPC systems are not reflected in these studies. These studies have centered on traditional HPC applications without much focus on Big Data workloads, which have an increasing presence on HPC systems [4].

Graph coloring for streaming graphs is one of the important Big Data problems that is needed in areas such as resource allocation and independence testing [5]. It should also be noted that many real-world problems are modeled as streaming graphs, including mapping neurons of the human brain and social network analytics. Hence, studies of future systems must include this important class of applications in order to truly understand the resulting performance of these systems.

We empirically quantify the effective interference caused by a Big Data workload on HPC applications during co-location. We use graph coloring as the representative Big Data application in this study and the FT, IS, CG, and MG kernel benchmarks from the NAS Parallel Benchmark (NPB) suite as the representative HPC benchmarks.

Using three different process-to-core mappings, we confirmed socket-sharing will yield the best performance for HPC applications. The memory-intensive MG benchmark experienced 4% slowdown when it ran in socket-exclusive mode while getting a performance boost in socket-sharing mode. Furthermore, the graph coloring benchmark experienced no notable performance variations due to the presence of HPC applications on the same node.

II. METHODOLOGY

A. System Setup

Experiments were conducted on TSUBAME-KFC, a 44-node supercomputer with two (2) interconnected InfiniBand FDR switches. Each node runs CentOS Linux release 7.3.1611 and has two Intel Xeon E5-2620 v2 processors with 64 GB of main memory. Red Hat’s GCC v4.8.5 and Open MPI v2.1.1 were used to compile the benchmarks.

The effect of socket-sharing and socket-exclusive process placement on application performance was explored. The recommended shared-stripe mapping [3] was compared against two socket-exclusive mappings shown in Figure 1. All processes were pinned to their target cores using Open MPI’s rankfile.

The system’s SLURM scheduler policy did not support node-sharing, hence 32 nodes were reserved using the scheduler and then application co-location was achieved by executing the jobs interactively.
II. RESULTS

Figure 2 shows the performance variation due to co-location for the different process mappings. For the NPB benchmarks, the shared-stripe mapping appeared to yield speedups for all benchmarks while the socket-exclusive mappings generally resulted in slowdowns. The standard deviation for CG, IS, and MG measurements were 1% or lower, while FT’s standard deviation increased up to 8.4%, even for the baseline runs. Therefore, FT’s performance variation cannot be attributed to its co-location with GC.

MG experienced the most significant slowdown for socket-exclusive allocations. The intra-application contention for last-level cache is very high under compact socket-exclusive mappings [3], hence, the presence of GC likely increases contention for access to DRAM and degrades MG’s performance.

The results in the lower plot of Figure 2 indicate that the GC benchmark has a very different interference profile from the HPC benchmarks. While the exclusive4bd mapping seemed to produce the best results for co-locating GC, the individual baseline and inference measurements varied non-trivially. The standard deviation for most cases ranged from 1.4% to 5.4%. The load-balancing and locality optimizations of this implementation introduces performance variations across baseline runs. Hence, a more in-depth study of these optimizations in the context of co-location is required.

IV. SUMMARY

Using kernel benchmarks from the NPB suite and a state-of-the-art graph coloring implementation, we have begun to quantify the effects of co-locating HPC and graph analytics applications. The memory-intensive MG benchmark experienced the largest performance variation due to co-location. We also confirmed that using socket-sharing, striped process placement actually improves the performance of HPC applications when co-located with graph analytics programs.

ACKNOWLEDGMENT

This research was supported by JST, CREST (Research Area: Advanced Core Technologies for Big Data Integration).

REFERENCES