Manycore Partitioning for Big Data Processing: Does Core Affinity Matter?

Joong-Yeon Cho   Hyun-Wook Jin

Department of Computer Science and Engineering, Konkuk University, Seoul 143-701, Korea

{dynamicj, jinh}@konkuk.ac.kr

Abstract

In this study, we aim to investigate the impact of core affinity on big data processing and discuss the potential for manycore partitioning that decides the core affinity based on the characteristics of threads, some of which are I/O intensive, some are computation intensive.

Categories and Subject Descriptors D.4.1 [*Operating Systems*]: Process Management

Keywords Core affinity, Manycore, MapReduce, Hadoop

1. Manycore partitioning and core affinity

The MapReduce programming model has been introduced to efficiently analyze large data sets. In this model, it generally happens that I/O-intensive threads and computation-intensive threads run on the same node simultaneously. For example, in the Hadoop framework, the threads of Hadoop Distributed File System (HDFS) are network and disk I/O intensive, while the MapReduce threads are commonly computation/memory intensive. Thus, we need to understand resource requirements of these threads and schedule them efficiently. It is well known that the core affinity significantly affects the performance of applications or efficiency of resource utilization. In this study, we aim to investigate the impact of core affinity on big data processing and discuss the potential for partitioning the cores based on the different characteristics of threads. Though there have been researches regarding the scheduling and core assignment on manycore systems [1, 2], the core affinity has not been considered in the context of big data processing.

Figure 1 shows the overall design of manycore partitioning for the Hadoop framework. Our previous work has revealed that the threads created by HDFS and MapReduce module prefer different core affinities [3]. Thus, the affinity manager endeavors to assign the cores that promise better I/O performance to the HDFS threads, while other cores to

SoCC '14, 3-5 Nov. 2014, Seattle, Washington, USA.

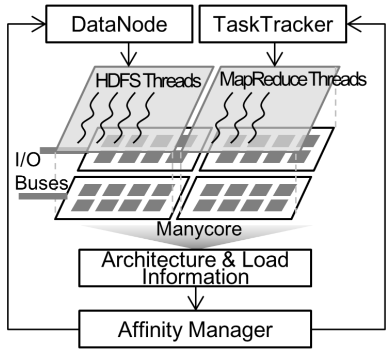

Figure 1. Manycore partitioning for Hadoop

MapReduce threads. The affinity manager considers cache layout, memory layout, distance to I/O devices, etc., and assigns different weights to each core. A core has two weight values for HDFS and MapReduce threads, respectively. We do not partition the cores neither statically nor rigidly; that is, a core can run both HDFS and MapReduce threads according to the current state. The experiment results showed that our preliminary implementation considerably improved the performance of a real application that ranks websites. We believe that these results emphasize the importance of the core affinity in big data processing. We are currently extending the affinity manager to consider the resource contention between threads and investigating various applications.

Acknowledgments

This research was supported by MSIP, Korea, under the Research Project on High Performance and Scalable Manycore Operating System (#14-824-09-011).

References

[1] H. Sasaki, T. Tanimoto, K. Inoue, and H. Nakamura. Scalability-based manycore partitioning. In Proc. of PACT, 2012.

[2] M. Bhadauria and S. McKee. An approach to resource-aware co-scheduling for CMPs. In Proc. of ICS, 2010.

[3] J.-Y. Cho, H.-W. Jin, M. Lee, and K. Schwan. Dynamic core affinity for high-performance file upload on Hadoop Distributed File System. To appear in Parallel Computing, DOI: 10.1016/j.parco.2014.07.005.