International Journal of Scientific Research in Computer Science, Engineering and Information Technology
© 2017 IJSRCSEIT | Volume 2 | Issue 5 | ISSN : 2456-3307

Splitting and Grouping of Jobs in Map Reduction for Various Multicore Processors

M. Navya[1], N. Padmaja [2]

[1]M.Tech Student, Department of Computer Science and Engineering, Padmavathi Mahila Visvavidyalayam, Tirupati, India

[2]Assistant Professor, Department of Computer Science and Engineering, Padmavathi Mahila Visvavidyalayam, Tirupati, India

ABSTRACT

The functionality of modern multi-core processors is often driven by a given power budget that requires designers to evaluate different decision trade-offs, e.g., to choose between many slow, power-efficient cores, or fewer faster, power-hungry cores, or a combination of them. Here, we prototype and evaluate a new Hadoop scheduler, called DyScale, that exploits capabilities offered by heterogeneous cores within a single multi-core processor for achieving a variety of performance objectives. A typical Map Reduce workload contains jobs with different performance goals: large, batch jobs that are throughput oriented, and smaller interactive jobs that are response time sensitive. Heterogeneous multi-core Processors enable creating virtual resource pools based on "slow" and "fast" cores for multi-class priority scheduling. Since the same data can be accessed with either "slow" or "fast" slots, spare resources (slots) can be shared between different resource pools. Using measurements on an actual experimental setting and via simulation, we argue in favor of heterogeneous multi-core processors as they achieve "faster" (up to 40%) processing of small, interactive Map Reduce jobs, while offering improved throughput (up to 40%) for large, batch jobs. We evaluate the performance benefits of DyScale versus the FIFO and Capacity job schedulers that are broadly used in the Hadoop community.

Keywords: Map Reduce, Hadoop scheduler, Dyscale, throughput, Heterogeneous processors

I. INTRODUCTION

In the existing system we have implemented the study to reduce network traffic cost for a Map Reduce job by designing a novel intermediate data partition scheme. Furthermore, we jointly consider the aggregator placement problem, where each aggregator can reduce merged traffic from multiple map tasks. A decomposition-based distributed algorithm is proposed to deal with the large-scale optimization problem for big data application and an online algorithm is also designed to adjust data partition and aggregation in a dynamic manner. Finally, extensive simulation results demonstrate that our proposals can significantly reduce network traffic cost under both offline and online cases. Map Reduce and its open source implementation Hadoop offer a scalable and fault-tolerant framework for processing large data sets. Map Reduce jobs are automatically parallelized, distributed, and executed on a large cluster of commodity machines. Hadoop was originally designed for batch-oriented processing of large production jobs. These applications belong to a class of so-called scale-out applications, i.e., their completion time can be improved by using a larger amount of resources. in the proposed system Here, we design and evaluate DyScale, a new Hadoop scheduler that exploits capabilities offered by heterogeneous cores for achieving a variety of performance objectives. These heterogeneous cores are used for creating different virtual resource pools, each based on a distinct core type. These virtual pools consist of resources of distinct virtual Hadoop clusters that operate over the same datasets and that can share their resources if needed. Resource pools can be exploited for multiclass job scheduling. We describe new mechanisms for enabling "slow" slots (running on slow cores) and "fast" slots (running on fast cores) in Hadoop and creating the corresponding virtual clusters. Extensive simulation

experiments demonstrate the efficiency and robustness of the proposed framework. Within the same power budget, DyScale operating on heterogeneous multi-core processors provides significant performance improvement for small, interactive jobs comparing to using homogeneous processors with (many) slow cores. DyScale can reduce the average completion time of time-sensitive interactive jobs by more than 40%. At the same time, DyScale maintains good performance for large batch jobs compared to using a homogeneous fast core design (with fewer cores). The considered heterogeneous configurations can reduce completion time of batch jobs up to 40%. There is a list of interesting opportunities for improving Map Reduce processing offered by heterogeneous processor design. First of all, both fast and slow Hadoop slots have the same access to the underlying HDFS data.

programmers. This paper focus on improve the MapReduce performance through a heterogeneity-aware data placement strategy: faster nodes store larger amount of input data. In this way, more tasks can be executed by faster nodes without a data transfer for the map execution. It addresses the problem of how to place data across nodes in a way that each node has a balanced data processing load. Given a data intensive application running on a Hadoop MapReduce cluster, our data placement scheme adaptively balances the amount of data stored in each node to achieve improved data-processing performance.

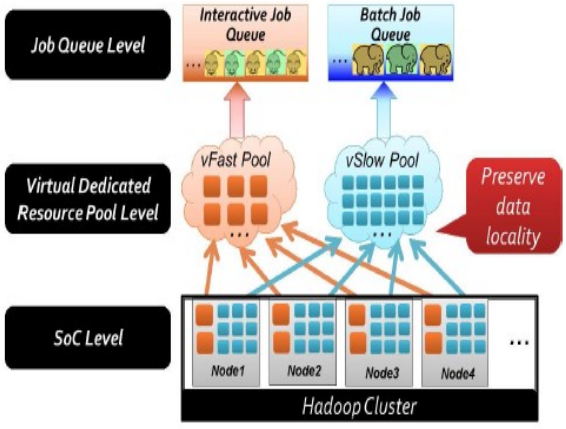
G. Lee,G. Chun, and R. H. Katz, conducted an experiment on "Heterogeneity-aware resource allocation and scheduling in the cloud [5]," Data analytics are key applications running in the cloud computing environment. To improve performance and cost-effectiveness of a data analytics cluster in the cloud, the data analytics system should account for heterogeneity of the environment and workloads. In addition, it also needs to provide fairness among jobs when multiple jobs share the cluster. In this work it mainly focus on resource allocation and job

II. DYSCALE FRAMEWORK

We propose a new Hadoop scheduling framework, called DyScale, for efficient job scheduling on the heterogeneous multi-core processors. First, we describe the DyScale scheduler that enables creating statically configured, dedicated virtual resource pools based on different types of available cores. Then, we present the enhanced version of DyScale that allows the shared use of spare resources among existing virtual resource pools. The number of fast and slow cores is SoC design specific and workload dependent. Here, we focus on a given heterogeneous multi-core processor in each server node, and the problem of taking advantage of these heterogeneous capabilities, especially compared to using homogenous multi-core processors with the same power budget. Our goal is twofold: 1) design a framework for creating virtual Hadoop clusters with different processing capabilities (i.e., clusters with fast and slow slots); and 2) offer a new scheduler to support jobs with different performance objectives for utilizing the created virtual clusters and sharing their spare resources.

Dedicated Virtual Resource Pools for Different Job Queues:

DyScale offers the ability to schedule jobs based on performance objectives and resource preferences. For example, a user can submit small, time-sensitive jobs to the Interactive Job Queue to be executed by fast cores and large, throughput-oriented jobs to the Batch Job Queue for processing by (many) slow cores. It is also possible for the scheduler to automatically recognize the job type and schedule the job on the proper queue. For example, small and large jobs can be categorized based on the number of tasks. A job can be also classified based on the application information or by adding a job type feature in job profile.



The attractive part of such virtual resource pool arrangement is that it preserves data locality because both fast and slow slots have the same data access to the datasets stored in the underlying HDFS. Therefore, any dataset can be processed by either fast or slow virtual resource pools, or their combination. To support

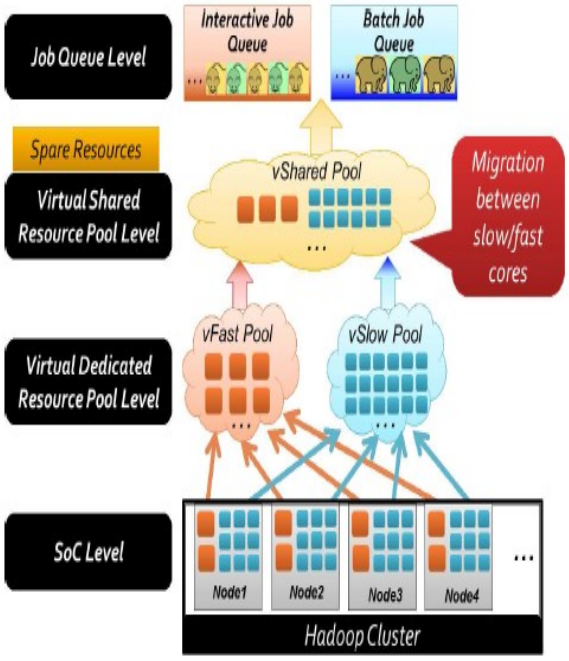
a virtual resource pool design, the Task Tracker needs additional mechanisms for the following functionalities:

- The ability to start a task on a specific core, i.e., to run a slot on a specific core and assign a task to it;
- To maintain the mapping information between a task and the assigned slot type.

The Task Tracker always starts a new JVM for each task instance (if the JVM reuse feature in Hadoop is disabled). It is done such that a JVM failure does not impact other tasks or does not take down the Task Tracker. Running a task on a specific core can be achieved by binding the JVM to that core. We use the CPU affinity to implement this feature. By setting the CPU affinity, a process can be bound to one or a set of cores. The Task Tracker calls spawn New JVM class to spawn a JVM in a new thread. The CPU affinity can be specified during spawn to force the JVM to run on the desired fast or slow core. An additional advantage of using the CPU affinity is that it can be changed during runtime. If the JVM reuse feature is enabled in the Hadoop configuration (note, that the JVM reuse can be enabled only for the tasks of the same job), the task can be placed on a desired core by changing the CPU affinity of the JVM.

Managing Spare Cluster Resources

Static resource partitioning and allocation may be inefficient if a resource pool has spare resources (slots) but the corresponding Job Queue is empty, while other Job Queue(s) have jobs that are waiting for resources. For example, if there are jobs in the Interactive Job Queue and they do not have enough fast slots, then these jobs should be able to use the available (spare) slow slots. We use the Virtual Shared (vShare) Resource pool to utilize spare resources; the spare slots are put into the vShare pool. Slots in the vShare resource pool can be used by any job queue.



The efficiency of the described resource sharing could be further improved by introducing the Task Migration mechanism. For example, the jobs from the Interactive-Job Queue can use spare slow slots until the future fast slots become available. These tasks are migrated to the newly released fast slots so that the jobs from the Interactive Job Queue always use optimal resources. Similarly, the migration mechanism allows the batch job to use temporarily spare fast slots if the Interactive Job Queue is empty. These resources are returned by migrating the batch job from the fast slots to the released slow slots when a new interactive job arrives. DyScale allows specifying different policies for handling spare resources. The migration mechanism is implemented by changing the JVM's CPU affinity within the same SoC. By adding the MIGRATE TASK action in the Task Tracker Action list in heartbeat Response, the Job Tracker can inform the Task Tracker to migrate the designated task between slow and fast slots.

III. CONCLUSION

Here we exploit the new opportunities and performance benefits of using servers with heterogeneous multi-core processors for Map Reduce processing. We present a new scheduling framework, called DyScale that is implemented on top of Hadoop. DyScale enables creating different virtual resource pools based on the core-types for multi-class job scheduling. This new Framework aims at taking advantage of capabilities of heterogeneous cores for achieving a variety of

performance objectives. DyScale is easy to use because the created virtual clusters have access to the same data stored in the underlying distributed file system, and therefore, any job and any dataset can be processed by either fast or slow virtual resource pools, or their combination. Map Reduce jobs can be submitted into different queues, where they operate over different virtual resource pools for achieving better completion time (e.g., small jobs) or better throughput (e.g., large jobs). It is easy to incorporate the DyScale scheduler into the latest Hadoop implementation with YARN [30], as YARN has a pluggable job scheduler as one of its components.

IV. REFERENCES

[1]. T. White, Hadoop: The Definitive Guide. Yahoo Press.

[2]. F. Ahmad et al., "Tarazu: Optimizing Map Reduce on Heterogeneous Clusters," in Proceedings of ASPLOS, 2012.

[3]. J. Dean and S. Ghemawat, "Map Reduce: Simplified data processing on large clusters," Communications of the ACM, vol. 51, no. 1, 2008.

[4]. M. Zaharia et al., "Delay scheduling: A simple technique for Achieving locality and fairness in cluster scheduling," in Proceedings of EuroSys, 2010.

[5]. Apache, "Capacity Scheduler Guide," 2010. Online]. Available: http://hadoop.apache.org/common/docs/r0.20.1/ capacity scheduler.html

[6]. Z. Zhang, L. Cherkasova, and B. T. Loo, "Benchmarking approach for designing a map reduce performance model," in ICPE, 2013, pp. 253–258.

[7]. S. Rao et al., "Sailfish: A Framework For Large Scale Data Processing," in Proceedings of SOCC, 2012.

[8]. A. Gates, O. Natkovich, S. Chopra, P. Kamath, S. Narayanam, C. Olston, B. Reed, S. Srinivasan, and U. Srivastava, "Building a high-level dataflow system on top of map reduce: The pig experience," PVLDB, vol. 2, no. 2, pp. 1414–1425, 2009.

[9]. A. Verma, L. Cherkasova, and R. H. Campbell, "ARIA: Automatic Resource Inference and Allocation for MapReduce Environments," in Proc. of ICAC, 2011.

[10]. "Play It Again, SimMR!" in Proceedings of Intl. IEEE Cluster' 2011.

[11]. S. Ren, Y. He, S. Elnikety, and S. McKinley, "Exploiting Processor Heterogeneity in Interactive Services," in Proceedings of ICAC, 2013.

[12]. H. Esmaeilzadeh, T. Cao, X. Yang, S. M. Blackburn, and K. S. McKinley, "Looking back and looking forward: power, performance, and upheaval," Commun. ACM, vol. 55, no. 7, 2012.

[13]. C. Bienia, S. Kumar, J. Singh, and K. Li, "The PARSEC benchmark suite: Characterization and architectural implications." in Technical Report TR-811-08, Princeton University, 2008.

[14]. "Pass Mark Software. CPU Benchmarks," 2013. Online]. Available: http://www.cpubenchmark.net/cpu.php?cpu=Intel + Xeon+E3-1240+%40+3.30GHz

[15]. F. Yan, L. Cherkasova, Z. Zhang, and E. Smirni, "Optimizing power and performance trade-offs of map reduce job processing with heterogeneous multi-core processors," in Proc. of the IEEE 7th International Conference on Cloud Computing (Cloud'2014), June, 2014.

[16]. A. Verma et al., "Deadline-based workload management for map reduce environments: Pieces of the performance puzzle," in Proc. of IEEE/IFIP NOMS, 2012.

[17]. R. Kumar, D. M. Tullsen, P. Ranganathan, N. P. Jouppi, and K. I. Farkas, "Single-is a heterogeneous multi-core architectures for multithreaded workload performance," in ACM SIGARCH Computer Architecture News, vol. 32, no. 2, 2004.

[18]. K. Van Craeynest, A. Jaleel, L. Eeckhout, P. Narvaez, and J. Emer, "Scheduling heterogeneous multi-cores through performance impact estimation (pie)," in Proceedings of the 39th International Symposium on Computer Architecture, 2012.

[19]. M. Becchi and P. Crowley, "Dynamic thread assignment on heterogeneous multiprocessor architectures," in Proceedings of the 3rd conference on Computing frontiers, 2006.

[20]. D. Shelepov and A. Fedorova, "Scheduling on heterogeneous multi core processors using architectural signatures," in Proceedings of the Workshop on the Interaction between Operating Systems and Computer Architecture, 2008.

[21]. K. Van Craeynest and L. Eeckhout, "Understanding fundamental design choices in single-is a heterogeneous multicore architectures," ACM Transactions on Architecture and Code Optimization (TACO), vol. 9, no. 4, p. 32, 2013.
[22]. M. Zaharia et al., "Improving map reduce performance in heterogeneous environments," in Proceedings of OSDI, 2008.
[23]. Q. Chen, D. Zhang, M. Guo, Q. Deng, and S. Guo, "Samr: A self-adaptive map reduce scheduling algorithm in heterogeneous environment," in IEEE 10th International Conference on Computer and Information Technology (CIT), 2010.
[24]. R. Gandhi, D. Xie, and Y. C. Hu, "Pikachu: How to rebalance load in optimizing map reduce on heterogeneous clusters," in Proceedings of 2013 USENIX Annual Technical Conference. USENIX Association, 2013.
[25]. J. Xie et al., "Improving map reduce performance through data placement in heterogeneous hadoop clusters," in Proceedings of the IPDPS Workshops: Heterogeneity in Computing, 2010.
[26]. G. Gupta, C. Fritz, B. Price, R. Hoover, J. DeKleer, and C. Witteveen, "Throughput Scheduler: Learning to Schedule on Heterogeneous Hadoop Clusters," in Proc. of ICAC, 2013.
[27]. G. Lee, B.-G. Chun, and R. H. Katz, "Heterogeneity-aware resource allocation and scheduling in the cloud," in Proceedings of the 3rd USENIX Workshop on Hot Topics in Cloud Computing, Hot Cloud, 2011.
[28]. J. Polo et al., "Performance management of accelerated map reduce workloads in heterogeneous clusters," in Proceedings of the 41st Intl. Conf. on Parallel Processing, 2010.
[29]. W. Jiang and G. Agrawal, "Mate-cg: A map reduce-like framework for accelerating data-intensive computations on heterogeneous clusters," in Parallel Distributed Processing Symposium (IPDPS), 2012 IEEE 26th International, May 2012, pp. 644–655.
[30]. Apache, "Apache Hadoop Yarn," 2013. Online]. Available: http://hadoop.apache.org/docs/current/hadoop-yarn/ hadoop-yarn-site/YARN.html
[31]. A. Verma, L. Cherkasova, and R. H. Campbell, "Resource Provisioning Framework for Map Reduce Jobs with Performance Go als," Proc. of the 12th ACM/IFIP/USENIX Middleware Conference, 2011.