Scheduling Data-intensive Many-task Computing Applications in the Cloud

Ke Wang
Department of Computer Science
Illinois Institute of Technology
Chicago, IL, USA
kwang22@hawk.iit.edu

Ioan Raicu
Department of Computer Science
Illinois Institute of Technology
Chicago, IL, USA
iraicu@cs.iit.edu

ABSTRACT

Scientific applications are ushering in the era of big data that has expedited the evolution of paradigm shifting from compute-centric model to data driven one. The Many-task computing (MTC) paradigm comes from a data driven model and aims to address the challenges of scheduling data-intensive workloads through over-decomposition. MATRIX is a distributed scheduler for fine-grained data-intensive MTC applications. We have evaluated MATRIX on the BG/P machine up to 4K cores, and on the Kodiak cluster up to 200 cores. We propose to integrate the Swift workflow engine with MATRIX to enable MATRIX running many more scientific applications in the Cloud. We also plan to replace the centralized schedulers of the Hadoop clusters, such as YARN and Mesos, by MATRIX to support MATRIX running the data-intensive applications in the data centers and Cloud domains. We believe that the two recently funded Cloud testbeds, namely the Chameleon and CloudLab, would offer great platforms for our experiments.

1. INTRODUCTION

Large-scale scientific applications are ushering in the era of big data such that task execution involves consuming and producing large volumes of input and output data with data dependencies among tasks. These applications are referred to data-intensive applications that cover a wide range of disciplines, including astronomy, astrophysics, bioinformatics, data analytics, data mining, and MPI ensembles [1]. The big data phenomenon [2] has expedited the evolution of paradigm shifting from compute-centric model to data driven one [3].

As systems are growing exponentially in parallelism approaching billion-way concurrency at exascale [4], we argue that the data driven programming models will likely employ over-decomposition that would generate even many more fined-grained tasks than available parallelism. While over-decomposition has been shown to improve utilization at extreme scales as well to make fault tolerance more efficient [5], it poses significant challenges on scheduling system to make fast scheduling decisions (e.g. millions/sec), in order to achieve the highest throughput and utilization. This requirement is far beyond the capability of today's centralized scheduling systems.

The Many-task computing (MTC) [6] paradigm comes from the data driven model, and aims to define and address the challenges of scheduling fine-grained data-intensive workloads. MTC applies over-decomposition to structure applications as Direct Acyclic Graphs (DAG), in which the vertices are small discrete tasks and the directed edges represent the data flows from one task to another. The tasks have fine granularity in both size (e.g. per-core) and durations (e.g. sub-seconds to a few seconds), and do not require strict coordination of processes at job launch as the traditional HPC workloads.

We justify that the task scheduling system for MTC will need to be fully distributed to achieve high scalability, efficiency, and availability. The architecture is shown in Figure 1.

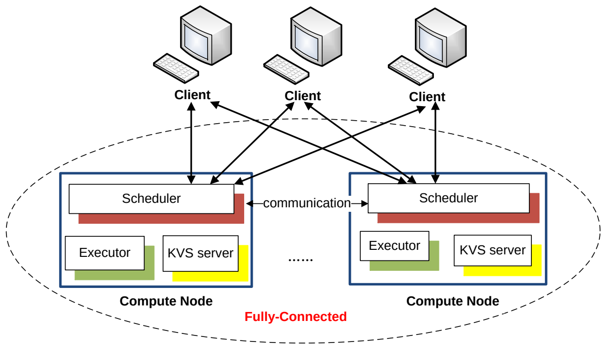

Figure 1: Fully-distributed scheduling architecture

Each compute node would run one scheduler and one or more executors/workers. As a compute node would have thousands of cores at exascale [4], and given the fact that the MTC data-intensive workloads have extremely large amount of fine-grained jobs/tasks with each task requiring one core for a short amount of time (e.g. millisecond), there would need a dedicated scheduler on one "fat" compute node forming 1:1 mapping. All the schedulers are fully connected, and receive workloads to schedule tasks to local executors. Therefore, ideally, the throughput would gain near-optimal linear speedup as the system scales. More importantly, the failures would only affect the tasks that are run on the failed compute nodes.

2. MATRIX Scheduling System

MATRIX [9] is a fully distributed task scheduling system for fine-grained data-intensive MTC applications with the architecture shown in Figure 1. MATRIX applies work stealing to achieve distributed load balancing, and supports the data-aware work stealing with the help of four distributed local queues (task waiting queue, task ready queue, task running queue, and task completion queue) and the ZHT distributed key-value stores (KVS) [7][8]. ZHT stores the task metadata including task dependency information, data flow information, data-locality information, and task execution progress in a transparent and scalable way.

MATRIX is developed in C++. MATRIX has about 3K lines of codes, with 8K lines of ZHT codebase. We have evaluated MATRIX using micro-benchmarks on an IBM Blue Gene/P supercomputer up to 4K-core scale. MATRIX maintains throughput as high as 13K tasks/sec, and 85%+ efficiency with fine-grained sub-second tasks (64ms) [9]. We also applied MATRIX to schedule two data-intensive applications, namely the image stacking application from Astronomy and the all-pairs applications from Bio-informatics up to 200 cores [10].

3. Proposed Work

We propose to integrate the Swift workflow engine with MATRIX to enable MATRIX running many more scientific applications in Cloud. We also plan to replace the centralized schedulers of the Hadoop clusters, such as YARN and Mesos, by MATRIX to support MATRIX running the data-intensive applications in the data centers and Cloud domains.

3.1 Swift Integration with MATRIX

The plan to enable MATRIX to run large-scale MTC applications in Cloud is to integrate with Swift [12]. Swift is a parallel programming system and workflow engine for MTC applications that cover a variety of disciplines, ranging from Biology, Earth Systems, Physical Chemistry, Astronomy, to Neuroscience, Economics, and Social Learning and so on. Swift will serve as the high-level data-flow parallel programming language between the applications and MATRIX. Swift would essentially output many parallel and/or loosely coupled distributed jobs/tasks with the necessary task dependency information, and submit them to MATRIX. Instead of having Swift manage the DAG, the DAG would be managed in a distributed way by MATRIX through the ZHT distributed KVS. Swift has been scaled to distribute up to 612 million dynamically load balanced tasks per second at scales of up to 262,144 cores [12]. This extreme scalability would absolutely advance the progress of enabling MATRIX supporting large-scale scientific applications at extreme-scales.

3.2 MATRIX in Data Centers and Clouds

Another direction is to push MATRIX in the data center and Cloud environment to run the data-intensive workloads from the Internet domain. The current MapReduce framework has a centralized task dispatcher to dispatch the data-intensive workloads to mappers that have data for specific tasks. We will extend MATRIX to support the MapReduce framework to enable distributed scheduling for the MapReduce data-intensive workloads. We will utilize distributed file systems, such as FusionFS [11], to help MATRIX implement data locality and the data-aware scheduling with the help of ZHT KVS. MATRIX + FusionFS will be the combination of distributed version of MapReduce framework, as opposed to the current centralized YARN + HDFS or the Mesos + Hive combinations. We will also compare MATRIX with YARN [13] and Mesos [14] schedulers.

4. Expected Experiments and Platforms

For Swift integration, the expected experiments are decomposing data-intensive applications from Biology, Earth Systems, Physical Chemistry, Astronomy, Neuroscience, Economics, and Social Learning as different DAGs. Then, MATRIX would be deployed in the Cloud environment to run all these applications at large scales. We also plan to extend our work through simulations to study the scalability of MATRIX up to exascale levels with millions of nodes. These experiments aim to explore the performance gains of scheduling large-scale scientific applications in the Cloud platforms through the MTC over-decomposition principle and the data-aware work stealing technique.

For applying MATRIX to run data-intensive applications in the data centers and cloud environments, we expect to deploy MATRIX on the Cloud platforms to run typical Hadoop applications, such as Word Count, Sort, Grep, Inverted Index, at the first stage. Then we will try to compare MATRIX with the YARN and Mesos schedulers by running the data-intensive applications that were run by YARN (the Yahoo! data-intensive workloads) and by Mesos (the Facebook data-intensive workloads), respectively. We expect that MATRIX will be more scalable and have the ability to make much faster scheduling decisions for these data-intensive workloads due to the distributed data-aware scheduling architecture and technique.

All these experiments are expected to be up to the scales of thousands of cores. The newly funded Cloud testbeds are perfect platforms for our expected experiments, because they are designed to run loosely coupled data-intensive applications. What's more, there shouldn't be problems of deploying and running MATRIX in both platforms, as MATRIX is developed in C++, which has great supports in different Linux distributions.

5. REFERENCES

[1] I. Raicu, et al. "Many-Task Computing for Grids and Supercomputers", Invited Paper, IEEE MTAGS 2008.

[2] Boyd, Danah and Kate Crawford. "Critical Questions for Big Data: Provocations for a Cultural, Technological, and Scholarly Phenomenon." Information, Communication, & Society 15:5, p. 662-679, 2012.

[3] Willcock, J., Hoefler, T., et al. "Active Pebbles: Parallel Programming for Data-Driven Applications." In: Proc. of ACM ICS 2011.

[4] V. Sarkar, et al. "ExaScale Software Study: Software Challenges in Extreme Scale Systems", ExaScale Computing Study, DARPA IPTO, 2009.

[5] X. Besseron and T. Gautier. "Impact of Over-Decomposition on Coordinated Checkpoint/Rollback Protocol", Euro-Par 2011: Parallel Processing Workshops, Lecture Notes in Computer Science Volume 7156, 2012, pp 322-332.

[6] I. Raicu. "Many-Task Computing: Bridging the Gap between High Throughput Computing and High Performance Computing", ISBN: 978-3-639-15614-0, VDM Verlag Dr. Muller Publisher, 2009.

[7] T. Li, et al. "ZHT: A Light-weight Reliable Persistent Dynamic Scalable Zero-hop Distributed Hash Table", IEEE IPDPS 2013.

[8] K. Wang, et al. "Using Simulation to Explore Distributed Key-Value Stores for Extreme-Scale Systems Services," IEEE/ACM Supercomputing/SC 2013.

[9] K. Wang, et al. "Paving the Road to Exascale with Many-Task Computing", Doctoral Showcase, IEEE/ACM Supercomputing/SC 2012.

[10] K. Wang, et al. "Optimizing Load Balancing and Data-Locality with Data-aware Scheduling", IEEE BigData, 2014.

[11] D. Zhao, et al. "FusionFS: Towards Supporting Data-Intensive Scientific Applications on Extreme-Scale High-Performance Computing Systems", IEEE BigData, 2014.

[12] T. Armstrong, et al. "Compiler techniques for massively scalable implicit task parallelism", IEEE/ACM SC 2014.

[13] V. Vavilapalli, et al. "Apache Hadoop YARN: yet another resource negotiator", SOCC '13.

[14] B. Hindman, et al. "Mesos: a platform for fine-grained resource sharing in the data center", NSDI'11.