MATRIX TASK EXECUTION FRAMEWORK

CS554 – Data Intensive Computing CWID – A20402686

Summary of the paper

The authors of the paper developed MATRIX, a real distributed task execution framework, which implements the work stealing technique to achieve distributed load balancing, and employs a distributed key-value store to manage task metadata. MATRIX is deployed on an IBM Blue Gene/P supercomputer and Amazon cloud. MATRIX is compared with Falkon on BG/P up to 4K cores. Throughput of MATRIX is 13K tasks/sec and efficiency is 90%+ for 1-sec tasks while Falkon only achieves 20% efficiency. In shared memory environment, work stealing is efficient load balancing technique at the thread level. It this pull-based method, the idle processors randomly steal tasks from the overloaded ones. In idle condition, scheduler randomly chooses some candidate neighbors and it goes through all neighbors sequentially to check the number of ready tasks, and steals tasks from the most heavily loaded neighbor. Every scheduler has a global membership list and is aware of all others in fully distributed architecture. Hence, idle scheduler (thief) steals tasks statically or dynamically from neighbor candidate. There is a lightweight discrete event simulator, called SimMatrix, which simulates MTC task execution framework of millions of nodes and billions of cores/tasks. In this, consecutive integer numbers range from 0 to the number of nodes N-1.

Each compute node runs a scheduler, an executor and a ZHT server in MATRIX. All the schedulers are fully connected with each one knowing all of others. It initiates work stealing to steal tasks from neighbors if a scheduler has no more waiting tasks. The tasks being stolen would be migrated from the victim to the thief. To keep the system and task metadata in a distributed, scalable, and fault tolerant way, ZHT is used. A client can submit tasks to any arbitrary scheduler in MATRIX. To improve the per client throughput, the tasks are submitted in batches. The client submits tasks by hashing each task to corresponding scheduler using hashing function. Clients can be of as many numbers as schedulers, which will evenly divide the total tasks among the clients. Each client would submit the tasks to a corresponding scheduler. Load is perfectly balanced, and there is no need to do work stealing if all the tasks are same. Tasks are in the wait queue initially. There are two fields in task: a counter, and a list of child tasks. The key is the task id, the two fields are part of the value. The counter represents the number of parent tasks. The task is then moved to the complete queue after finishing the execution of a task.

MATRIX has a monitoring program on the client side that can poll the task execution progress. Monitoring program query the status of a specific task by sending query message to ZHT servers. It records logs about the system state, such as the number of total, busy and free execution threads, the lengths of the three queues for each scheduler. The input file (configuration file) consist of parameters, such as the number of tasks, the batch size task dependency type, task length, number of executing cores, work stealing initial poll interval, work stealing poll interval upper bound.

How is this work different than the related work?

When MATRIX is compared with YARN in scheduling Hadoop workloads, MATRIX performed well on YARN by 1.27X for typical workloads, and by 2.04X for the real application. There are other state-of-the-art fine-grained task scheduling systems that are targeting loosely coupled parallel applications such as Sparrow and CloudKon. Sparrow is like MATRIX except in each scheduler is aware of all the compute daemons, this design can cause many resource contentions when the number of tasks are large. Performance of MATRIX is 9 times greater than that of Sparrow. CloudKon also has similar architecture as MATRIX, except for CloudKon, which focuses on the Cloud environment, and relies on the Cloud services, SQS to do distributed load balancing. Falkon is a centralized task scheduler with the support of hierarchical scheduling. CloudKon has a similar scheduling architecture but, the workers of CloudKon need to pull every task from SQS leading to significant overheads for NOOP tasks, on the other hand, MATRIX migrates tasks in batches through the work stealing technique that introduces much less communication overheads.

Identify the top 3 technical things this paper does well.

- Work stealing technique to achieve distributed load balancing.
- Efficiently execution of fine-grained tasks with work stealing algorithm.
- Efficiency of MATRIX (92% to 97%) when compared with Falkon.

Identify 3 things the paper could do better.

- As ZHT is used in MATRIX, this paper could implement method to increase the efficiency of ZHT when number of node increases.
- MATRIX could be in C for even better speedup.
- Hardware complexity is higher in MATRIX.

If you were to be an author of a follow up paper to this paper, what extensions would you make to improve on this paper?

If I was an author of this paper, I would have included the method to decrease the hardware complexity of MATRIX.