CMPSCI 590S Systems for Data Science **Fall 2017**

Lecture 26

Lecturer: Emery Berger *Scribe: Tao Chen, Zhaoduo Wen*

26.1 Review of Last Lecture - Effect of Hardwares and Softwares on Performance

Apparently, both hardwares and softwares have influence on the performance of system. In terms of hardware, CPU, GPU, network, disk, memory and RAM/cache can affect the performance of system. With regard to software, algorithm, programming language/runtime system, OS, architecture (multi threads, multi processes) are factors to the performance. If using multi processes, data share is expensive, while concurrency is one of the problems in multi threads.

26.2 Paper1: Making Sense of Performance in Data Analytics Framework

This paper proposes blocked time analysis for quantifying performance bottlenecks in distributed computation frameworks. It tries to understand the bottlenecks to scale up a distributed computation framework, like CPU bound, I/O bound (disk bound, network bound), etc.. Although the paper finds out CPU (and not I/O) is often the bottleneck, it only carries out experiments on Spark environment, whose key idea is in memory computation.

Acutally, I/O is generally the bottleneck for distributed systems. In the condition that CPUs use shared memory generally takes 100 cycles, but using network I/O generally takes $2 \sim 3$ order of magintude slower than shared memory, i.e. 100,000 cycles, because network communication requires copying data through the network.

Remote Direct Memory Access(RDMA) is proposed to solve the problem, super fast especially for small messages. RDMA is a direct memory access (a pointer) from the memory of the source computer into that of destination computer without involving either one's operating system. RDMA requires no copying, so it is generally 1 order of magnitude faster than network I/O, which is 10,000 cycles.

26.3 Spark vs Hadoop

The advantage of Spark is in-memeory computation using RDD. Everything is a transformation from one RDD to another RDD. RDD allows it to efficiently provide fault tolerance by logging the transformations used to build a dataset (its lineage) rather than the actual data.

Question: Is Spark always faster than Hadoop? Answer: No. Spark is limited by the following two conditions.

- Data is too large to fit in RAM

- No iterations in the map-reduce job (Spark optimizes iteration)

In a nutshell, perfomance of the two models is workload dependent, which is always true.

26.4 What leads to straggles

As pointed out in the paper, GC usually leads to the stragglers. However, it can be solved in a "terrible" way. First, use PL without GC, like C++; Second, don't allocate objects.

The pros and cons of GC are as follows:

Pros

- safety (use after free)
- no memory leaks. If memory is freed too soon, program might crash; If memory is freed too late, program consumes too much memory.

Cons

- Pause time: GC takes time which would stop the program temorarily.
- Space overhead: The assigned heap size is always larger than the actual size the program really needs. Generally, heap size is $3 \sim 5$ times bigger than the actual size.

Question: Is there any better GC? Answer: We have real-time GCs that never pause, but those things are relatively slow. There is always a trade-off that either you get small multiple pauses or a very long pause during GC.

26.5 Paper2: Scalability! But at what COST?

COST stands for Configuration Outperfoms a Single Thread. When we try to measure the scalibility of distributed system, it is important to pick a good baseline. Scaling can be easy if you start at very slow baseline. The rule of thumb is the baseline should be the fastest, state-of-the-art version on one node.

Problems of COST

1. Data is too big to fit in memory (RAM).
2. Data is too big to fit in local disk.
3. Gustafsons Law. The number of CPUs should should scales with the size of problem.
4. The algorithm on a single core/node may be different from that on multi-cores or multiple nodes.
5. Fault tolerance problem. Program running on single thread may suffer from single point failure.
6. The cost of moving data from multiple data sources to a single machine is relatively high.

7. Graph analysiscs perspective. This paper creates a domain specific programming language, which does not mean the general purpose. (For example, Java, C++, Python are general purpose language. SQL is a domain specific programming language for database. Shell and MapReduce are also the domain specific language.)