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

Lecture 17: Distributed Shared Memory

Lecturer: Emery Berger *Scribe(s): Akash Mantry, Mohanish Panchal*

In this lecture, we first discussed the challenges with multi-processing and then proceeded to discussion of Distributed Shared Memory systems.

17.1 Challenges with Multiple Processing

Multiple processing includes loads of challenges. Due to different address spaces, concurrency becomes a problem. A message passing infrastructure is needed for communicating state. Passing messages is expensive and may result in non-deterministic delivery. If speed is one of the requirements, synchronous message passing becomes a bottleneck.

Debugging is also a challenge. The only way to debug is to do logging. There are multiple stacks, heaps, threads, etc. to keep track of and due to lack of a global view, it becomes hard to find and ascribe bugs.

Some of other requirements of multiple processing which are hard to implement include:

- Discovery mechanism (at the start)
- Dynamic partitioning of heap
- Load balancing
- Fault-tolerance (synchronous checkpoints)

17.2 Distributed Shared Memory

A distributed shared memory (DSM) is a form of memory architecture where physically separated memories can be addressed as one logically shared address space. Here, shared implies that the address space is shared.

17.2.1 Addressing in shared memory system

Distributed memory system works with machine local addressing. Suppose if a program asks for the value of a variable x (say), the system has to figure out if the variable is mapped to current machine or is somewhere else. In first case, it will be a simple memory read. In second, it has to be a network message that fetches the data from a remote machine.

But, there are some problems with this naive approach.

- Synchrony - This can be avoided by having many threads and thus we can hide latency by switching to other threads.
- Concurrency - Locks provide exclusive access to a shared memory but the lock itself is in shared memory. This forms a kind of recursive problem and can be avoided by using a separate locking service.

17.2.2 TreadMarks

TreadMarks is a distributed shared memory system whose design focuses on reducing the amount of communication necessary to maintain memory consistency. If a resource is not present locally, the node raises a segmentation fault and cues a signal handler to handle the network messages. Messages are also sent in batches to improve performance.

Lazy release consistency provides further improvement by sending updates only when the lock is released.

Some of the issues with TreadMark were:

- It only worked well with structures which had contiguous memory accesses like matrices but failed with graphs.
- Though, batching reduced network communication, it was still slow.

17.3 Grappa

17.3.1 Insights of the Grappa System

Grappa is a DSM system implemented in C++ which does not require spatial locality or data reuse to perform well. The idea is to have local memory and partitioned global memory. The code has free access to local memory but has to request for accessing global memory.

Some of the features of Grappa include:

- Grappa doesn't include fault tolerance. The reason being most of the jobs don't take long.
- InfiniBand is a an architecture and specification which features low latency high bandwidth It is widely used in high end servers.
- RDMA, Remote Direct Memory Access, is a direct memory access from the memory of one computer into that of another without through either one's kernel. It provides high-throughput and low-latency. It avoids context switches by having no system calls.

17.3.2 Delegation

This concept of completing work on the local machine rather than remotely doing the work. It is based on the principle that local memory access is way cheaper than remote memory access. It achieves this by sending the lambda (code) instead of the data.

17.3.3 Work Stealing

- Proactively go and ”steal” work from an overloaded fellow worker.
- It is used for load imbalance correction. There can be different strategies -
  - Randomly choosing a victim and then decide how much work to take.
  - ”Half Stealing” - steal half the work of the chosen victim.
  - ”The power of two choices” - between two choices, we pick the victim as the one with more work.
- Work Stealing is more fine grained when looking at degree of load balancing control.
- Only the processes with ”no work” are doing the load balancing work.