COMPSCI 590S Systems for Data Science **Fall 2016**

Lecture 18: Distributed Shared Memory

Lecturer: Emery Berger *Scribe(s): Udit Saxena,Freddie Sanchez*

18.1 Continuation of discussion on Grappa

We discussed the concept of 'Work Stealing' and the difference from Work Stealing and Work Sharing.

1. Work Stealing :
   (a) Proactively go and "steal" work from an overloaded fellow worker.
   (b) It is used for load imbalance correction. There can be different strategies -
      i. Randomly choosing a victim and then decide how much work to take.
      ii. "Half Stealing" - steal half the work of the chosen victim.
      iii. "The power of two choices" - between two choices, we pick the victim as the one with more work.
   (c) Work Stealing is more fine grained when looking at degree of load balancing control.
   (d) Only the processes with "no work" are doing the load balancing work.
   (e) Asking another worker if they have work, doesn?t create an interruption in the system.
2. Work Sharing :
   (a) Another load sharing paradigm where load is "shed" onto other workers. Different strategies include :
      i. Random picking
      ii. Round Robin
      iii. Min-first
   (b) Work Sharing is coarse grained when looking at degree of load balancing control.
   (c) Creates a bottleneck since it needs to check everyone loads. Everyone needs to be asked for their load.
   (d) Everyone is asking if the other workers need work.
   (e) When workers are asking/answering to a query for their load, they aren't doing meaningful work - this situation gets worse when everyone is asking everyone for their work.
   But to prevent the issue of stopping working when asking/being asked, we the concept of lock-free data structures.

18.2 Cooperative Scheduling

While on a single core, processes go to sleep and are context-switched out on an I/O call.
The process scheduling framework can be thought of as a scheduling - descheduling framework.

There are two kinds of scheduling:

1. : Preemptive Scheduling: Every process gets a time quantum (5ms). On the expiration of this time slice, the process is context-switched out, and another process is allowed to continue computation.
   This kind of scheduling is good for preventing malicious processes from taking an arbitrarily long computation time. Used in aggressive programming it can also help avoid infinite loops.
   The problem with this kind of scheduling is:
   (a) High scheduling overhead with lower granularity - bad for context switches for workloads which are large.
   (b) Lock Convoying
   (c) Warm up or collateral damage or frequent cache flushing - the TLB is flushed every time it is switched.
   (d) Can also kick out a program at awkward stages, such as when one has a lock.
2. Cooperative Scheduling: In Cooperative scheduling, a process is allowed to continue processing until it explicitly yields the CPU for another process to continue. The onus is on the process to hand over the CPU - if it doesn't, it can keep processing for as long as it wants. This is used in environments which are considered safe and have no malicious processes.

Some systems have EDF (earliest deadline first) scheduling between processes. The CPU has deadlines to schedule programs.

18.3 Page based DSM

Page based DSM refers to the Distributed Shared Memory at the level of a page which is the smallest unit of memory being shared across the distributed system. There are two kinds:

1. True Sharing : When the same location on the same page needs to be shared across a distributed system. When the page is both logically and physically shared.
2. False Sharing: When different locations on the same page need to be shared. Here the locations are logically unshared, but physically shared.

False sharing is overcome by using diff merges/updates - it is a cheaper mode of communication. Grappa doesn't do this - it batches updates to amortize update overhead.

18.4 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.