CMSC416: Introduction to Parallel Computing

Topic: Parallel Networks and Filesystems
Date: April 25, 2024

(Network) Congestion

- This occurs when multiple compute resources (ex. multiple processes) use the same physical hardware for communication
- How do you mitigate congestion? One way is to choose a routing algorithm that efficiently routes a packet between two switches.
    - Static routing: Routes are manually configured in a routing table
    - Dynamic routing: Route changes at runtime, but does not necessarily take into account the level of congestion in the network
    - Adaptive routing: Similar to dynamic routing in that routing can change, but this does take into account the network's congestion and can change routes to avoid congested routes.
- Network congestion can cause performance variability in the execution time of a program
    - Performance variability can be detected by splitting execution time into communication and computation time and studying the difference in variabilities of the times
    - Variability can occur if the network is congested during certain times of day, or if there are many ongoing tasks on nodes around the node you're running your job on, as there is more traffic flowing on the same networking hardware.

Topology-aware node allocation

- Another way to mitigate congestion is to use topology-aware node allocation to reduce the distance packets have to travel during the execution of a program
    - Ex. for a fat tree network, try to allocate nodes so that they are on the same switch, or if there are more nodes than ports on the switch, map them all to the same pod.
    - The general idea is to keep compute nodes as physically close as possible so there is a lower probability that traffic has to traverse many levels upwards, as more switches use the links in the higher levels.

Adaptive flow aware routing (AFAR)

- This is another routing method that dynamically re-routes traffic to avoid congested spots in the network.
- The heuristic it uses to reroute traffic is:
    - If the current load of any link in the system is greater than a threshold, re-route a flow going across that link to a link with a lower load compared to the network's average, which is effect is conducting load balancing.

I/O in parallel programs

- Parallel programs perform input/output operations when they read/write inputs/outs from files or write outputs to a designated process.
    - When all processes are communicating with one designated process, it is considered a non-parallel I/O. This cannot scale efficiently as the load on this process will be much higher than that on the rest of the processes.
- **Parallel Filesystem, also known as the IO sub-system**
    - The parallel filesystem contains home directories and scratch space. It is mounted onto the compute and login nodes, which allows data stored across multiple servers to be shared by multiple processes.
    - The file system is connected to compute nodes using network links. The file system and the compute cluster each have their own network topologies
        - The nodes of the leaf switches of the compute cluster are called LNET router nodes, and they have a link to the Object storage server (OSS) nodes on the I/O sub-system network, providing a connection between nodes and the file system.
    - **Benefits of a Parallel filesystem**
        - Parallel file systems optimize I/O bandwidth by dividing read/writes across multiple Object Storage Target (OST) (disks).
        - They also can stripe files across multiple OSS nodes, meaning chunks of the file are placed on different physical nodes.
            - This is beneficial when the request for a file is faster than the rate a single node can provide it at, because with multiple nodes transmitting a file's content, total throughput is increased.
    - Tape drives can be used by parallel filesystems for archiving data. Robotic arms are used to pick a certain tape drive from a rack of tapes to do read/writes.
    - Zaratan uses the BeeGFS as its filesystem, a community-supported parallel file system
    - **Burst buffer**
        - A burst buffer consists of fast storage devices like non-volatile memory devices (solid state disks, flash memory) used to increase read/write speeds between compute nodes and the parallel filesystem.
        - Data on burst buffers persist longer than data on on-node memory (DRAM), but there's less capacity available than the storage of the file system.
        - Burst buffers are useful for storing intermediate data while conducting large amounts of computation, as their fast read/write speeds allow nodes to quickly switch between writing data and running computations.
    - **I/O patterns**
        - Either one process is charged with reading/writing all data to storage devices or multiple processes can read/write.
    - **I/O profiling tools**

Commented [1]: https://en.wikipedia.org/wiki/Data_striping

- To determine which I/O pattern works best, need to conduct empirical analyses on their performance by using profiling tools like Darshan and Recorder.
    - Recorder is a tool developed by UIUC that can trace I/O function calls made across the stack.
    - GitHub repo for project: https://github.com/uiuc-hpc/Recorder