CMSC416: Introduction to Parallel Computing

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

Routing Algorithm:

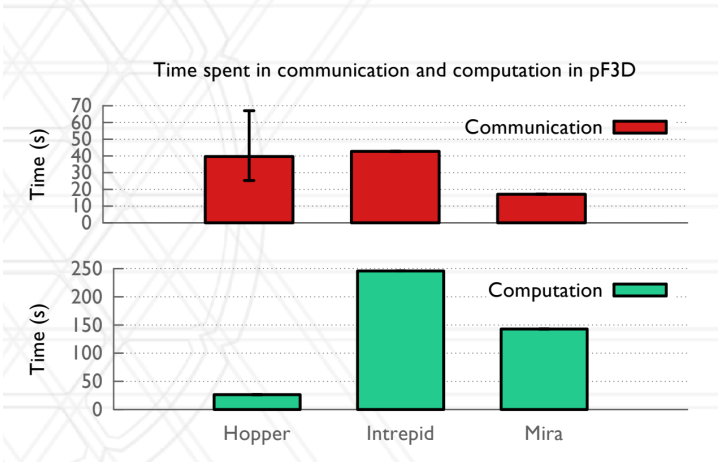
One way to mitigate congestion is through a routing algorithm. The routing algorithm essentially decides how a package is going to get routed between source and destination switch. There are two types of routing: Static and Dynamic

1. **Static routing** - (seen in fat-tree network) where each router is pre-programmed with a routing table. This occurs at boot time.

2. **Dynamic routing** - the router can be changed on runtime. Ex: when a machine is on, a user can change the routes that are taken for a pair of routers.

However dynamic routing doesn't necessarily need to take network congestion into account, it can change the routing without having to worry about network congestion. But if dynamic routing does take network congestion into account and adapts to the network congestion, that is called **Adaptive routing**.

Performance Variability:



Performance variability in computing systems stems from communication bottlenecks rather than computational capabilities. The chart is highlighting two key factors that contribute to the variability: placement of jobs across the system and contention for

shared network resources. By optimizing job distribution and mitigating network congestion, users can hopefully reduce the observed discrepancy in performance.

Mitigating Congestion:

There are three ways to mitigate congestion:

1. **Network topology aware node allocation** - on a torus machine, you can assign convex prisms to every job. By allocating nodes while considering the underlying network topology, jobs can be placed that minimizes potential congestion.

2. **Adaptive routing** - move traffic around the hotspot so you do not use the links that have the hotspot.

3. **Within a job:** The user tries to figure out if theft can map MP processes in a better way than the default. On a cray machine you allocate certain sets of nodes, which are not contiguous, you can still try arranging the MP processes within your own partition such that processes that communicate more frequently are near on the network, not sending traffic far away which hopefully leads to less congestion.

Topology-aware node allocation:

It's a network strategy tool used for distributed computing and networking to optimize performance of tasks when allocating resources. Resources are allocated based on their position within the network of nodes, the connectivity between nodes in regards to edges, and the communication paths available. This allocation method also helps to distribute workloads more evenly and reduce network congestion, leading to greater efficiency. If you have a job that has less than 20 ports, you want to assign it within a single switch. This ensures the traffic for each of those jobs does not cross the switch fountain. The idea is to allocate nodes in a manner, to prevent sharing of links by multiple jobs as much as possible. Ultimately jobs in this network strategy do not interfere by overloading the same contiguous network links which could potentially cause network congestion and slow down the network.

AFAR:

Adaptive flow aware routing is a technique that adapts the paths taken by network flows dynamically based on recorded congestion levels across various links in the network. It

takes into account the requirements of individual flows and monitors the condition of the network to direct traffic away from the most congested areas. This congestion-aware routing method for paths enables AFAR to overall improve network performance, serving as a key advantage of this network strategy. On average every link has 5 network flows through them.

Topology Aware Mapping:

Requires the user to know what the machine topology is. In a topology-aware mapping, processes are allocated to nodes within a job to optimize performance, and this mapping takes into account the physical and logical layout of the system. In regards to the input of this mapping, we receive an application communication graph, which is a representation of all the communication that occurs between the different processes of an application. Hence, the nodes represent processes while the edges represent communication links. The machine topology is also provided as input, which refers to the structure of the computing system in regard to interconnected devices. However, this mapping is a graph embedding problem, which is commonly known as an NP-hard problem, meaning it is computationally difficult to find a solution in polynomial time. Heuristics can be used to come up with a solution that may be close to accurate here though in polynomial time. This mapping can also be potentially integrated into a load-balancing strategy, which aims to distribute the workload evenly across nodes.

Input and Output:

When you are reading a file/datasets or writing outputs/checkpoints you are performing I/O. The most basic I/O is no parallel I/O. An example of this is dedicating a single process that is doing the I/O to and from the system. This is not scalable.

Parallel Filesystem:

A parallel filesystem allows multiple clients to access files simultaneously, which differs from a serialized file system. Home directories in this file system are personal storage areas for users to store files and personal settings. These directories are accessible by any node in the cluster. The scratch space is storage for temporary data during computation processes. This filesystem is mounted on all login and compute nodes, meaning it is accessible from the various nodes that users can interact with. This is also referred to as an I/O sub-system, including the data transfer to and from storage devices. There are different types of parallel filesystems including Lustre, BeeGFS, GPFS, and PVFS. These parallel filesystems can improve I/O bandwidth by spanning read and write operations across multiple OSTs (Object Storage Targets. This

distribution of operations enhances the overall efficiency and speed of data handling. Files can also be striped within and across multiple I/O servers, and this enables the distribution of contents of files across multiple physical devices. Each compute node runs I/O daemon, which is responsible for interacting with the parallel filesystem mounted on the compute node. The MDS (Metadata Server) servers file metadata such as ownership, and permissions, and provide directory updates. These servers are important, as they manage information about the data, which allows for efficient access and manipulation of the files in the filesystem.

Tape Drive:

Tape drives are for longer term storage and physically they are longer than physical tapes. These are separate entities. They read and write to the parallel filesystem. Typically used for archive data. Tape drives are cheaper but are slower because it takes longer to get data from it.

Burst Buffer:

Used as cache for the compute node and the filesystem. It is a fast and intermediate storage. There are two types of design, Node local burst buffer and Remote burst buffer. The best use cases for burst buffers are for storing checkpoints data, prefetching input data and visualizing simulations. It’s more expensive.

I/O Libraries, Patterns, Tools:

There are three levels of libraries: High, Middle, and Low.

1. High-level: HDF5, NetCDF
2. Middle-level: MPI-IO
3. Low-level: POSIX IO

There are various differing I/O patterns that are used in parallel file systems. The first is the use of single process I/O where a serial approach with one process possesses control over I/O operations. Multiple processes with a shared file is another methodology, which requires more synchronization to manage multiple reads and writes associated with a shared file. Additionally, multiple processes can access separate files, which does not require the same degree of synchronization as the previous strategy. The performance of these I/O patterns is dependent upon several factors, including the number of readers and writers, the size of the files being processed, as well as the type of filesystem in use. There are tools for I/O profiling including Darshan and Recorder.