Parallel file system (PFS)

45 participants

Participants were shown the following background information on PFS, and were then asked to indicate the importance, for their research, of having PFS capability in the HPC clusters they used. Participants who chose either *Very important* or *Moderately important* were asked to provide reasons.

Parallel file system (PFS)

Background

- The HPC cluster model uses hundreds of compute nodes, each containing several CPUs, each of which contains several processors (cores) to perform calculations. The two NJIT HPC clusters, Kong (general-access) and Stheno (Dept. Mathematical Sciences only), between them contain 3,448 cores. Jobs running on these cores write/read temporary files to/from disk as part of the computational processes.
- When the temporary files exceed a few gigabytes, writing to and reading from temporary files on disk consumes the most computation time- i.e., disk I/O is the bottleneck. In addition, parallel jobs handling temporary files of any size will encounter this bottleneck.
- The problem is exacerbated by the very large increase in the computational capacity and number of cores in Kong in summer 2015, which has resulted in a very large increase in the amount of large temporary data that the compute nodes are attempting to write to and read from disk.

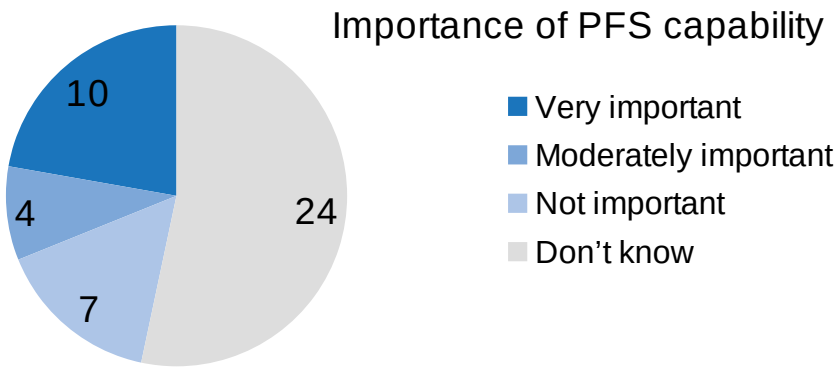
Implication

- Researchers at NJIT using HPC clusters are dealing with increasingly large sets of data, with the concomitant need for much higher I/O capacity for temporary space.

PFS appliance

- A PFS is a file system that distributes file data across multiple servers and provides for concurrent access by multiple tasks.
    - A PFS can be used by both serial and parallel processes running on an HPC cluster.
    - A PFS appliance can be connected to multiple clusters.
    - PFS examples: IBM General Parallel File System (GPFS), Lustre

Participants' indications of the importance of PFS capability are shown in the graph below.



Reasons why 10 participants considered PFS capability *Very important*:

- Recently, we have been trying to move the simulation results out of AFS and it is so difficult. We need to share our simulations output data with the community and we can't do that without moving it out of the cluster.

- We run massive hydrodynamic computation requiring tens of million of nodes. We just published a paper in the prestigious journal Geophysical Research Letters on the nature of oil flow from the Gulf spill of 2010. We could not have done it without the HPC resources at NJIT.

- needed for higher level calculations and reasonable time for product

- Because we often have to share data across multiple users on different servers

- It would partially reduce the I/O bottleneck and would allow to execute my simulations across systems without having to transfer data across.

- We have found that I/O takes up the most time in our experiments. To reduce this time we use the /scratch directories on the nodes but there is still a large overhead when writing to disk. A PFS is important to our work because it would reduce the time spent in reading and writing to disk which all of our jobs need to do.

- Execution time matter during code profiling and can be set off if proper file access is not maintained

- Several of the tools I use for genome evolution is done in parallel; it would be infeasible to do serial runs.

- Speed up my training.

- fast

Reasons why 4 participants considered PFS capability *Moderately important*:

- for faster IO

- large set of data, and we would like to share with certain colleagues

- We had a use case where spawning multiple independent threads that write to a single file would have decreased the overall processing time exponentially, Limited IO bandwidth is a huge bottle neck, which limits the amount of processing power you can tap into on kong.

- I have limited knowledge about the technical terms involved. Based on the information provided here "When the temporary files exceed a few gigabytes, writing to and reading from temporary files on disk consumes the most computation time - i.e., disk I/O is the bottleneck". I deal with large datasets - sometimes up to 500 gigabytes. I imagine that this would improved the computing time (ignore it if I am wrong).