A position paper

File system and runtime system for big data

Osamu Tatebe
University of Tsukuba
tatebe@cs.tsukuba.ac.jp

Key need

New style of scientific method that analyses big data to make a great finding is widespread. Big data is often generated by experiment devices such as accelerators, telescopes, and DNA sequencers. Data analysis typically requires to read all data, which requires I/O bandwidth ideally proportionate to the data size. At least, it is desirable to achieve I/O bandwidth proportionate to the CPU core count. Moreover, when the CPU core count increases, the required number of parallel I/O operations per second (IOPS) increases. Thus, IOPS is also required to be proportionate to the CPU count.

Proposed architecture

According to the history, memory architecture in parallel machines shifted to distributed memory from shared memory to achieve scalable memory bandwidth to the CPU node count. Storage is same, and should be distributed to achieve scalable I/O bandwidth to the CPU node count. Or, at least an I/O node needs to be deployed for each, for example, ten CPU nodes. This means we utilize local disk in a compute node, or an I/O node per fixed number of compute nodes, as a distributed storage.

In case of distributed memory, there is no global addressing support in hardware. Applications manage it typically by MPI. On the other hand, storage is different from memory. The data in storage assumes to be persistent, stable and reliable. And, global addressing, more precisely global pathname access is assumed by applications since the data in storage will be accessed by the POSIX API transparently from all processes in applications. On the other hand, the I/O access performance is not uniform in this case since it depends on the latency and bandwidth to the storage; the storage may be a local disk, or a remote disk. These three problems are major issues for the distributed storage, or non-uniform storage access system.

Problem and solution

Regarding persistency, stability and reliability, traditional solution is to provide multiple paths to storage. If we utilize a local disk on a compute node, providing multiple paths from different nodes is not straightforward. Another solution may include multiple file replicas among compute nodes, cluster-wide RAID and cluster-wide erasure coding.

To provide global pathname access, file system metadata servers (MDS) are often used to manage file system namespace that points to file locations. The performance, scalability, and reliability of MDS are major concern. Regarding performance, there are some researches including GIGA+ [1] and SkyFS [2]. GIGA+ showed relatively scalable performance and achieves 98K file creates per second using 32 servers. Our preliminary research shows 250K file creates per second using 15 servers, and suggests the possibility to design scalable and reliable MDS.

Regarding non-uniform access performance, key issue is to utilize locality of storage. The maximum storage performance can be achieved by accessing local disk on all compute nodes. To maximize locality, local disk should be utilized as much as possible. To improve the local disk access rate when reading files, the process scheduling is a key. Our earlier work proposed a scheduling method to improve local access rate when executing large scale workflow [3].

Reference

[1] Swapnil Patil, Garth Gibson, "Scale and Concurrency of GIGA+: File System Directories with Millions of Files", Proceeding of USENIX Conference on File and Storage Technologies (FAST), 2011

[2] Jing Xing, Jin Xiong, Ninghui Sun, Jie Ma, "Adaptive and scalable metadata management to support a trillion files", Proceedings of IEEE/ACM International Conference for High Performance Computing, Networking, Storage and Analysis (SC), 2009

[3] Masahiro Tanaka, Osamu Tatebe, "Workflow Scheduling to Minimize Data Movement using Multi-constraint Graph Partitioning", Proceedings of IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing (CCGrid), pp.65-72, doi: 10.1109/CCGrid.2012.134, 2012