Can Clustered File Systems Support Data Intensive Applications?

Rajagopal Ananthanarayanan, Karan Gupta, Prashant Pandey, Himabindu Pucha
Prasenjit Sarkar, Mansi Shah, Renu Tewari
IBM Research

This WIP attempts to address the question—*Can cluster file systems match specialized file systems such as Google's GFS for data-intensive applications?*

With the explosive growth of information and applications exploiting that information, large-scale data processing has emerged as an important challenge. Example applications include web search, indexing and mining, discovering biological functions from genomic sequences, astronomical phenomena from telescope imagery, and brain-scale networks for cognitive systems. These data-intensive applications demand a scalable, yet cost-effective storage layer. The storage layer for these applications needs to scale to thousands of nodes so that highly parallel applications process petabytes of data in hours rather than days. At the same time, the infrastructure needs to be built on commodity components to minimize cost while tolerating the failures that are typical for these components. Given the large volumes of data being processed, another key requirement for this storage layer is to enable shipping compute to data rather than the other way around.

Recently, enterprises faced with these critical needs of data-intensive applications have proposed *specialized file systems*, built with the unique requirements of the layer in mind. For example, Google developed the GFS file system that is optimized for large sequential and small random reads on a small number of large files residing on a commodity cluster. Companies such as Yahoo and Kosmix followed this trend by emulating the GFS architecture in Hadoop DFS and KFS respectively. For the scope of this work, we choose the open source Hadoop DFS (HDFS) as a representative specialized file system.

This work argues that cluster file systems can also rise to the challenges posed by these data-intensive applications. Moreover, there are inherent advantages to using cluster file systems in this paradigm: (1) These file systems can provide well-known traditional file APIs to these new class of applications. (2) Given that these file systems have been around for a while, they are enabled with a rich set of management tools such as automated backup and disaster recovery etc. (3) These file systems can simultaneously support legacy applications that rely on traditional file APIs obviating the need to maintain different storage layers for different applications. (4) Finally, an interesting trend further motivates this study: enterprises are increasingly incorporating data analytics in their workflows resulting in a mix of legacy and the new class of data-intensive applications accessing a common storage layer.

There is ample evidence that existing cluster file systems such as Lustre, PVFS, OCFS and GPFS meet the scalability requirement of the data-intensive applications; modern cluster file systems scale to thousands of nodes while providing high performance and availability. Furthermore, these file systems can be configured using commodity processors, disks and networks without the need for specialized SANs or enterprise-class storage. Thus, to make our case that cluster file systems can indeed support data-intensive applications, work is in progress to demonstrate the following:

Cluster file systems can support compute shipping to data nodes. We choose IBM's GPFS as a representative cluster file system and Map-Reduce as the data-intensive application framework to study the challenges in compute shipping. To enable this functionality, we first exposed block location information to the Map-Reduce framework through a new GPFS `ioctl`. The framework uses this information to ship the compute task to where the data resides. Further, we found that Map-Reduce performs computation on large HDFS blocks (e.g. 64 MB at a time) since it helps HDFS utilize the sequential bandwidth of disks. So our next challenge was to enable large block sizes in GPFS. While a first-cut solution of increasing GPFS block size helped Map-reduce performance, the large block size was harmful for the cache efficiency and the prefetching performance of GPFS, thereby impacting the performance of legacy applications. To accommodate both workloads we introduced the concept of *metablocks*. We used small block sizes in GPFS (512 KB–2 MB), but changed the block allocation policy so that contiguous blocks are placed on the same node (in groups of 32-128 blocks). This allows GPFS to manage prefetching and caching in terms of small blocks, but enables Map-Reduce to schedule jobs to work on a group of blocks (exposed to Map-Reduce as a single "metablock").

Our preliminary results suggest that our enhanced version of GPFS shows negligible performance degradation for legacy applications while matching the performance of the specialized file system, HDFS, for data-intensive applications. In addition, GPFS exceeds HDFS performance for meta-data intensive and for cache-friendly applications.

Cluster file systems can efficiently provide fault tolerance in a commodity hardware environment. HDFS replicates data to deal with node/disk failures. Our initial experiments with GPFS configured with a data replication factor of two show that the write performance of GPFS is comparable to that of HDFS. Encouraged by this finding, our ongoing work explores the design of an efficient n-way replication strategy for cluster file systems to achieve fault tolerance. We are also investigating GPFS' response to failures.