To what extent the network affects DFSs?

Gustavo Bervian Brand, Adrien Lèbre
École des Mines de Nantes / ASCOLA Research Group - Nantes, France
Email: {Gustavo.Bervian-Brand, Adrien.Lebre}@mines-nantes.fr

I. Introduction

Providing an infinite capacity of computing power to efficiently process a huge amount of data is an old objective of Computer Science. The prevalent solution consists in using a large computing facility where a Distributed File System (DFS) enables users to share files throughout the nodes composing the infrastructure. Although several contributions have been done since the first proposals [1], the design of DFSs has not really evolved and modern solutions such as Lustre [2], GoogleFS [3] and HDFS [4] have all been built on the metadata and I/O servers (or datanodes) model. In such systems, files are divided into chunks spread across distinct I/O servers and a metadata server is used to maintain the locality of each chunk. Unfortunately, with the increasing amount of data that modern applications manipulate as well as the way of accessing them, the single metadata model is no more satisfying [5]. Following former proposals such as Ceph [6], a promising model consists in balancing the management of both data and metadata across several nodes leveraging pre-defined distribution strategies. Although these systems push forward the limits in terms of scalability and performance, they are still not considering the network traffic inherent to their internal structure as a fundamental concern.

From our point of view, the design of DFSs should take into account the infrastructure's physical topology in order to limit expensive network communications as much as possible. To have more reference data and better understanding of the impact of the network traffic, we conducted several experiments upon the Grid'5000 testbed [1] and discussed the results. The questions we put forward are:

- Why "local scope" applications should suffer the penalty of external server communications in charge of managing either data or metadata?
- Why data should be pushed over the network if it will be used by local nodes or simply deprecated by the end of the applications execution?
- Why data has to be pushed from one location to another instead of using a on-demand pulling model?
- Should all data have the same level of reliability and how ensure availability minimizing network traffic?

Our ultimate objective is to propose a new model of DFS that intrinsically reduce the exchanges related to the DFS to the minimal needs of each application.

[1] http://www.grid5000.fr

II. Analyzing Network Impacts

We conducted several experiments to analyze the impact of the network exchanges related to DFSs at LAN and WAN levels. More precisely, five scenarios have been defined as combinations between the placement of the clients (C), the datanodes (D), the metadata server (M) and the way they are interconnected. We used the "-" and "/" symbols to refer respectively to LAN and WAN interconnections. For example, the *C-D-M* scenario corresponds to the experiment where applications run within a single site interconnected through a LAN whereas the *CD/M* makes reference to a WAN-wide scenario where the metadata server is separated by a WAN interconnection from the clients and the datanodes, which are installed at the same node in this case.

Due to the large usage of Map/Reduce applications, we ran the *Grep*, *Write* and *Sort* benchmarks provided with the Hadoop suite [4]. The two former ones are data intensive applications while the latter is more metadata intensive. Since the cost of the data and metadata management depend on the file system protocols, we selected two well known DFSs which differ in the way they spread data across I/O servers: Lustre [2] and HDFS [4]. To measure only the relevant data traffic, we set the replication factor of HDFS to 1 (default is 3). The chunk sizes were configured to 64 MB for HDFS and 4 MB for Lustre (default values). Experiments have been run on the Nancy site of Grid'5000 inside an subset of nodes isolated from other applications. WAN constraints have been emulated using the Linux Traffic control tool (TC) by inserting a 200 ms latency between nodes according to the scenario.

Table I presents results of the first experiments where the number of nodes grows from 16 to 64 nodes writing 64 slides of 128 MB, summing 8 GB. For each scenario, the number of clients (*tasktrackers*) equals the number of datanodes, leading to a testbed of up to 129 nodes (64 clients, 64 datanodes and 1 metadata server).

As expected, using more nodes improves the performance with a better completion time. However, looking deeper, the more nodes are used, the less significant the gain is since there are more DFS exchanges. This adds an overheard, which limits the performance. In most cases, it is a waste of resources to use additional nodes, especially if they are placed outside the local infrastructure. This is particularly true for the HDFS results. As an example, the completion times for the local instances of the 16 nodes sort scenarios (*CD-M* and *C-D-M*) are similar to the

Systems	HDFS			Lustre		
Scenarios	Grep	Writer	Sort	Grep	Writer	Sort
Tests with 16 nodes						
CD-M	81	61	115	110	54	149
C-D-M	83	66	119	114	48	151
CD/M	135	76	153	110	52	148
C-D/M	134	76	164	125	55	159
C/D-M	169	408	715	345	194	590
Tests with 32 nodes						
CD-M	76	45	73	89	41	87
C-D-M	76	57	77	89	45	81
CD/M	121	62	117	88	41	92
C-D/M	122	63	118	99	48	95
C/D-M	136	245	470	275	175	512
Tests with 64 nodes						
CD-M	68	44	60	73	36	68
C-D-M	82	62	65	90	56	80
CD/M	110	63	102	80	36	92
C-D/M	127	72	105	101	60	90
C/D-M	141	224	354	272	176	525

TABLE I: Runtime of the tests with 8 GB file size

WAN ones with twice or four times more nodes (*CD/M*, *C-D/M* and *C/D-M*). Regarding the Lustre results, the DFS exchanges' impact is less significant when compared to the HDFS values. This is due to the way HDFS and Lustre manage their data and metadata. In Lustre, data is spread on a 4MB chunk basis, leading to more traffic related to data accesses in comparison with the metadata ones. Indeed, the impact becomes significant when both the metadata and the datanodes are separated from the clients by WAN interconnections. In such situations, the overhead becomes critical (in average 4 times worse).

In the following experiments, we grew the size of the file from 8 GB to 32 GB with 64 clients, 64 datanodes and 1 metadata server with the best and the worst scenarios of the sort benchmark from Table I. As we can see on Figures 1 and 2, the impact of the DFS exchanges gets worse when the file size increases.

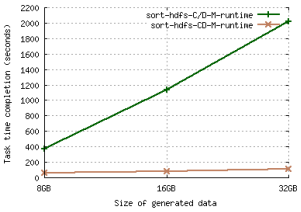

Fig. 1: Sort runtime at HDFS

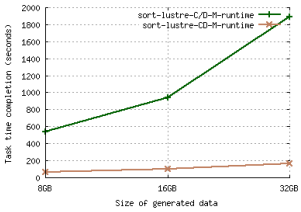

Fig. 2: Sort Runtime at Lustre

Although we didn't conduct experiments where client nodes are spread between distinct sites (such as *C-D-M/C*, *C-D-M/C-D* or *CD-M/CD* scenarios), we can assume that the impact on the performance will be significant. A Map/Reduce program spread between several sites will be penalized by the DFS's structure.

III. Model overview and perspectives

Our goal is to investigate a new DFS model that takes into account the impact on the performance implied by the physical topology. Our idea is to keep both data and metadata as close to the processes as possible. More precisely, we promote to place them along the locations where the new content is written. We define this concept as *implicit striping strategy*: files will be split into chunks as usual but their size and location will be adapted according to the access pattern of each application and the physical topology.

We assume that a physical infrastructure can be divided into subgroups interconnected by links with physical limitations in terms of latency and bandwidth. A group can be either a node (HDD/SSD), a cluster (LAN) or a federation of clusters (WAN). Our model aims at optimizing the "local" data placement in order to maximize each application's performance by leveraging these groups.

Applying such an approach to both data and metadata management will enable us to limit the scope of DFS exchanges to the application's scope. By such a way, once retrieving its data, an application that is LAN-wide will not suffer the penalty of relying on remote entities in charge of managing either data or metadata. In the meantime, it will be possible to extend a LAN-wide application to a WAN-wide scope dynamically while restricting the expensive traffic between sites to the minimal, providing the concept of *solicited* vs *unsolicited* traffic. Finally, several applications will be able to run simultaneously, using its own scope of data and metadata management. This can also reduce bottlenecks that may occur when several applications compete for the same metadata or data nodes (the impact of an I/O intensive application should not be significant on the performance of the others).

As an example, the performance we target for one application dealing with 16GB upon two sites should be close to the performance of two applications, each using half of the nodes, dealing with 8 GB and running as if they were independent (on each site with their own DFS).

The concept of adapting the DFS behavior according to its effective usage and to applications' needs/behaviors may be extended to support *solicited reliability* aspects, dealing with hotspots, etc. We are currently leveraging a symmetric DFS proposal [7] to finalize a proof of concept.

References

[1] E. Levy and A. Silberschatz, "Distributed file systems: concepts and examples," *ACM Comput. Surv.*, 1990.

[2] P. Schwan, "Lustre, Building a file system for 1000 node clusters," in *Proc. of the Linux Symposium*, 2003.

[3] S. Ghemawat, H. Gobioff, and S. T. Leung, "The Google file system," in *Proc. of the 19th ACM SOSP*, 2003.

[4] D. Borthakur, *The Hadoop Distributed File System: Architecture and Design*. The Apache Software Foundation, 2007.

[5] M. K. McKusick and S. Quinlan, "Gfs: Evolution on fast-forward," *Queue*, 2009.

[6] S. A. Weil, S. A. Brandt, E. L. Miller, D. D. E. Long, and C. Maltzahn, "Ceph: A scalable, high-performance distributed file system," in *Proc. of the 7th OSDI*, 2006.

[7] P. Riteau, A. Lebre, and C. Morin, "Handling persistent states in process checkpoint/restart mechanisms for hpc systems," in *CCGRID*, 2009.