Datacenter Storage Architecture for MapReduce Applications

Jeffrey Shafer, Scott Rixner, Alan L. Cox

Rice University
{shafer, rixner, alc}@rice.edu

Abstract

Data-intensive computing systems running MapReduce-style applications are currently architected with storage local to computation in the same physical box. This poster argues that upcoming advances in converged datacenter networks will allow MapReduce applications to utilize and benefit from network-attached storage. This is made possible by properties of all MapReduce-style applications, such as streaming storage access patterns. By decoupling computation and storage, stateless compute nodes containing inexpensive, low-power processors can be deployed in large numbers to increase application performance, improve reliability, and decrease power consumption.

1 Introduction

Data-intensive problems exist today in the domains of commerce, science, and engineering. These problems are solved only by writing applications that leverage the power of thousands of processors to manipulate data sets ranging in size from hundreds of terabytes to dozens of petabytes. Such applications can only be run in the datacenter, where they can take advantage of computation, storage, power, and cooling resources on a large scale [2].

A new programming model – called *MapReduce* – has emerged to support these data-intensive applications. Programs are divided into a *map* function and a *reduce* function. The map function takes a key/value pair and produces one or more intermediate key/value pairs. The reduce function takes these intermediate pairs and merges all values corresponding to a single key. Both of these stages can run independently on each key/value pair, exposing enormous amounts of parallelism. In a MapReduce application, a middleware layer (such as Yahoo's Hadoop) is responsible for distributing the parallelized computation to a large pool of loosely-synchronized processors and aggregating the results.

MapReduce-style applications use a different storage architecture than traditional datacenter applications such as transactional processing, which commonly use a centralized data store accessed across the network. A key premise for MapReduce-style computing systems is that there is insufficient network bandwidth to move the data to the computation, and thus computation must move to the data instead [2]. Based on the assumption that remote data can only be accessed with low bandwidth and high latency, current MapReduce architectures co-locate computation and storage in the same physical box, as shown in Figure 1. Storage is accessible across the network via a global file system that provides replication and load balancing. However, the MapReduce framework attempts to migrate computation to use data on locally-attached disks whenever possible.

Upcoming advances in converged network fabrics will enable a more flexible storage architecture for MapReduce applications without any loss of efficiency. Converged network technologies have previously focused on allowing applications to access existing storage-area networks without either transiting a front-end storage server or using a dedicated storage network adaptor. In contrast, we advocate using such converged networks to decouple computation and storage for MapReduce. The network properties provided by quality-of-service and lossless flow control protocols will allow disks to be removed from the compute nodes and instead attached to switches throughout the datacenter. No dedicated storage-area network will be required. Properties of MapReduce applications will allow this decoupling of computation from storage while maintaining equivalent performance, enabling a more flexible and efficient datacenter design.

2 Datacenter Architecture

We advocate a new storage architecture for MapReduce in the datacenter that incorporates remote storage, as shown in Figure 2 for a single rack. This design employs networked disks – simple unstructured block storage devices connected to the network [3] – along with the use of a converged network to decouple storage resources from computation resources. Compute nodes are equipped with

processors but not disks, and storage nodes are equipped with disks but are not used for computation. Instead of co-locating storage and computation in the same box as in the traditional MapReduce storage architecture, this design co-locates storage and computation on the same Ethernet switch.

The advantages of using remote disks in a datacenter are numerous. First, both computation and storage resources can be adjusted to meet application requirements during both cluster construction and operation. Second, computation and storage failures are decoupled from each other. This eliminates wasted resources that would have been used to reconstruct lost storage when a computation node fails. Third, fine-grained power management techniques can be used, whereby compute and storage nodes are enabled and disabled to meet current application requirements. Finally, because computation is now an independent resource, a mix of both high and low power processors can be deployed. The runtime environment managing application execution can change the processors being used for a specific application in order to meet administrative power and performance goals.

3 MapReduce and Remote Storage

MapReduce frameworks such as Hadoop make an explicit assumption that storage is local to computation. However, a fresh look at the architecture reveals that these resources can be decoupled and connected merely to the same network switch. This is made possible for two major reasons:

First, MapReduce applications use storage in a manner that is different than ordinary applications. Data is typically stored in large blocks (e.g. 64MB) and accessed using a simple write-once, read-many coherence model [1]. Application performance depends more on the bandwidth to access entire blocks than the latency to access any particular element in a block. Furthermore, data is accessed in a streaming pattern, rather than random access. This allows storage requests to be pipelined and data to be streamed across the network, minimizing any effect from network latency.

Second, modern network switches offer extremely high performance. A typical 48- or 96-port Ethernet switch provides the full bisection bandwidth across its switching fabric, allowing an entire rack of hosts to communicate with each other at full network speed. If more bandwidth is needed, link aggregation can be used to connect multiple Gigabit links to the same host. Modern switches also have low latency – under $2u$s for a minimum-sized Ethernet frame – and emerging cut-through fabrics promise to reduce latency further. Compared to hard disk seek latencies measured in milliseconds, the forwarding latency of modern switches is negligible[1]. Thus, high performance switches add minimal overhead to remote storage accesses that are located in the same rack as the computation node. By combining modern network switches and a streaming access pattern, remote disks can potentially achieve throughputs approaching that of local disks.

We have tested this proposed architecture on a small scale using a cluster computer running the Hadoop MapReduce framework. This cluster was configured in two ways: First, using disks installed in the compute nodes, and second, moving those same disks to other non-compute nodes attached to the same Ethernet switch. In the second case, the lightweight ATA-over-Ethernet protocol was used to export each disk across the network as a low-level block storage device with minimal overhead. Preliminary experiments conducted using data-intensive benchmarks such as a random data generator showed that the decoupled storage architecture had equivalent performance to the local storage architecture.

4 Conclusions

We propose a new architecture for MapReduce applications that decouples computation from storage by using a converged network and networked disks. Removing the disks provides an opportunity for more flexible and efficient datacenter design. The ratio between computation and storage resources can be changed both during datacenter construction and dynamically during operation to meet application performance and power requirements. Clusters can also be equipped with a mix of computation nodes containing either high-performance (i.e., Intel Xeon) or high-efficiency (i.e., Intel Atom) processors, allowing applications to select the performance and power mix that is desired.

References

[1] HDFS (hadoop distributed file system) architecture. `http://hadoop.apache.org/core/docs/current/hdfs_design.html`, 2008.

[2] R. E. Bryant. Data-intensive supercomputing: The case for DISC. Technical Report CMU-CS-07-128, Carnegie Mellon University, May 2007.

[3] Gibson et al. A cost-effective, high-bandwidth storage architecture. In *ASPLOS-VIII*, 1998.

[1] Even solid-state disk latencies are much higher than network switching latencies. Regardless, conventional disks are still the likely choice for MapReduce datacenter storage due to capacity and cost.

5 Appendix

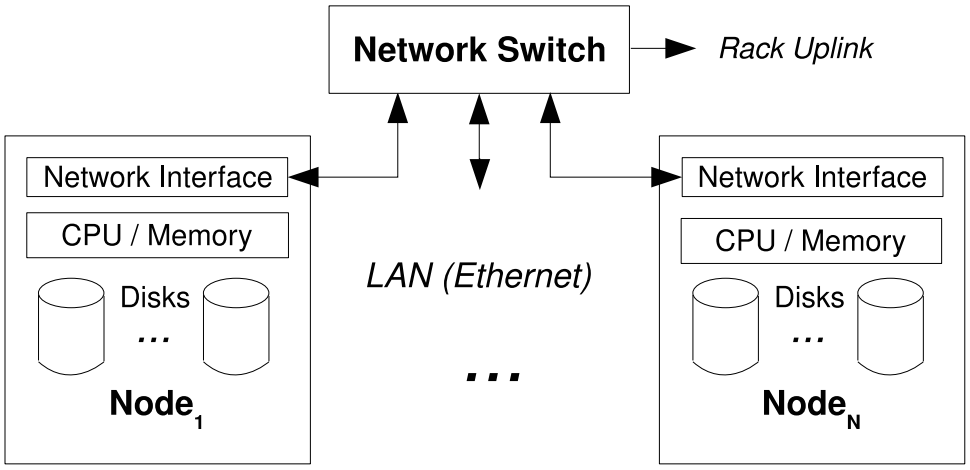

Figure 1: Local-Storage Architecture (Single Rack)

Figure 1 shows the storage architecture for a traditional datacenter running a MapReduce framework such as Hadoop. In this architecture, data is stored on disks local to the processors. Although the data is accessible over the network, computation is migrated to the data whenever possible.

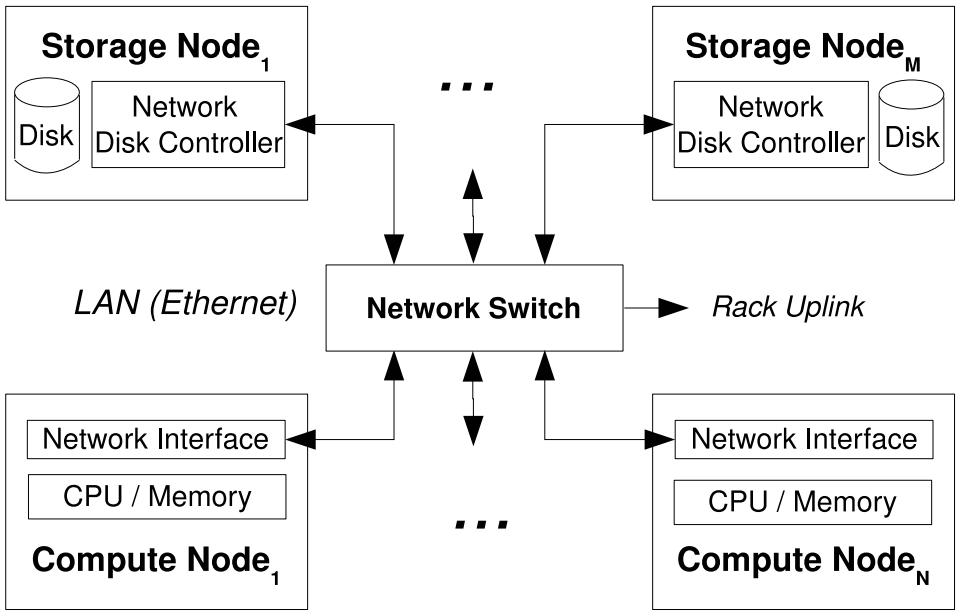

Figure 2: Remote-Storage Architecture (Single Rack)

Figure 2 shows the proposed remote storage architecture. In this datacenter, storage is provided by network-attached disks connected to the same network switch as the compute nodes.