DSC-204A: Scalable Data Systems, Winter 2024

19: Batch Processing

Lecturer: Hao Zhang *Scribe: Leo Chen, Chenze Fan*

1 Distributed Storage

In distributed systems, it is crucial to balance the workload and ensure that requests are processed by the correct storage containing the queried data.

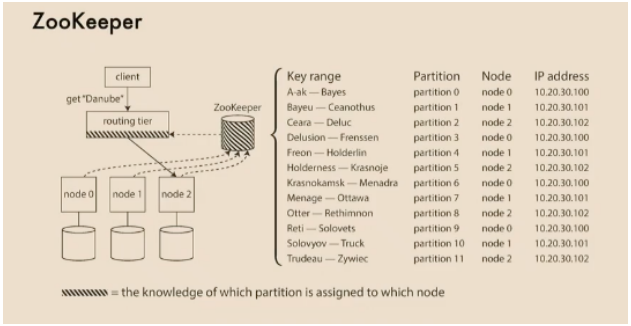
One approach is to randomly assign requests to nodes, checking if the key is present. If not, the request is forwarded to the next node until a reply is received.

Another method is to route all requests to a routing tier first, which then directs them to the appropriate storage.

A third approach involves clients being aware of partitioning and the assignment of partitions to nodes, although this is uncommon as clients typically do not handle storage directly.

1.1 ZooKeeper

ZooKeeper plays a role in this process by having each storage node register itself with metadata. ZooKeeper maintains this mapping in real-time, notifying subscribed actors of any changes in partition mapping.

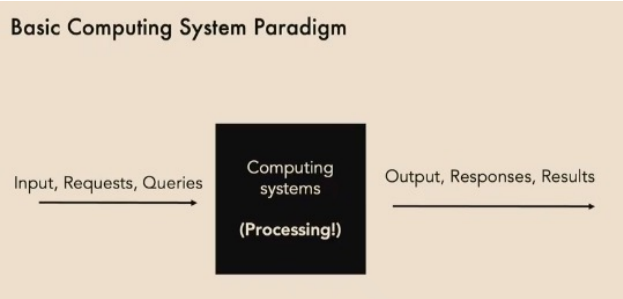


1.2 Remark

Regarding data distribution, replicating data improves fault tolerance but requires more disk space, making it less scalable. The single-leader strategy is a common replication solution. Partitioning data allows for scaling but can lead to hot partition issues. Methods like key range, hashing, and hybrid approaches aim to mitigate hot partition problems.

2 Distributed Computing

With the expansion of the Google search engine and increasing data processing workloads, there's a shift towards more complex and heavy computing on distributed storage. Deep learning algorithms, in particular, demand significant compute resources due to their iterative nature during gradient updates, leading to the rise of platforms like Spark and TensorFlow.



As depicted in the figure above, computing systems process various user inputs and return results to the user. Similar to network latency, computing also experiences processing latency, categorized into three main types:

1. Batch-processing systems: Results are not required in real-time and typically take hours or overnight to complete tasks. For example, ChatGPT scans documents with millions of words to generate a short summary.
2. Stream processing: Near real-time systems, such as ChatGPT providing results in a streaming fashion or the Google search engine displaying the first page while generating subsequent pages based on relevance scores.
3. Online systems: Immediate feedback systems, like Bard providing entire results at once.

However, the boundaries between these three types of latency remain somewhat unclear.

2.1 IO/Unix Pipes

2.1.1 Batch processing

Terminal is a prototype of batch and stream processing system. Basic Unix commands includes ls, mkdir, cat, cut,....

A pipe is the communication between commands. Pipe use | symbol to seperate different commands, like **cat dups.txt | sort | uniq** do the commands sequentially and take the output of a command as the input of the next command. Such pipe is a simple batch processing command.

Communication between commands can be achieved by several ways. One is to write the output to a file and take the file as the input to the next command.

2.1.2 Unix philosophy

The Unix philosophy is a set of cultural norms that followed by Unix developers.

1. Make each program do one thing well. To achieve a new task, start afresh rather than complicate old programs by adding new features.

2. Expect the output of every program to become the input to another, as yet unknown, program. Therefore, don't clutter output with extraneous information. Avoid stringently columnar or binary input formats. Don't insist on interactive input.

3. Design and build software, even operating systems, to be tried early, ideally within weeks. Don't hesitate to throw away the clumsy parts and rebuild them.

4. Use tools in preference to unskilled help to lighten a programming task, even if you have to detour to build the tools and expect to throw some of them out after you've finished using them.

2.1.3 Stream Processing

Unix also support stream processing. The command tail is a good example. When launching a program, user can tail the log of the program with tail command.Tail processes data as a stream, it doesn't need to read the entire file into memory to operate. Whenever a new line is written, the tail command will update the output in real-time.

The limitation of Unix is that it runs on a single machine. That's where tools like Hadoop comes in.

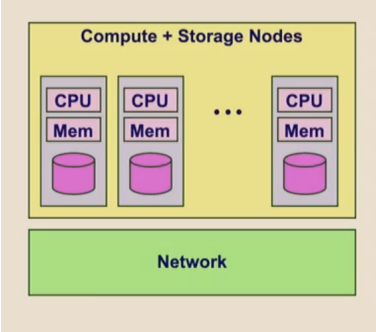
2.2 MapReduce

MapReduce is a programming paradigm that enables massive scalability across hundreds or thousands of servers in a Hadoop cluster. The MapReduce is built on GFS(Google File System) or HDFS(Hadoop Distributed File System). MapReduce is originally built on GFS. HDFS is an open source system that basically a replication of GFS.

TensorFlow is an open source machine learning framwork produced by Google. While its a good project, it failed eventually. Meta built Pytorch, which is a better framework and built by a more consolidated team.

2.3 Cluster

Typical cluster machine built by Google is constituted by nodes and network. Each node has medium-performance processor, modest memory, and 1-2 slow disks. The network is around 100Gb/s across racks, and 10Gb/s within racks.



The problem is how to build a system which support processing large data like 1 terabyte by commands like ls.