

ECE 574 – Cluster Computing

Lecture 23

Vince Weaver

`https://web.eece.maine.edu/~vweaver`

`vincent.weaver@maine.edu`

13 April 2023

Announcements

- HW#9 was posted
- Project status due
- Second Midterm will be 18th (Tuesday)



Midterm Review

- Can have 1 page of notes (1 side, 8.5" x11")
- Cumulative, but concentrating on stuff since last exam
- Performance, Speedup, Parallel efficiency, Scaling
- Shared Memory vs Distributed Systems
- Tradeoffs. Given code and hardware, would you use MPI, pthreads, CUDA, etc?
- OpenMP: dynamic vs static scheduling
- MPI, especially its limitations
- GPGPU/CUDA/OpenCL: read code, know the tradeoffs



(overhead of copying memory around)

- BigData: sizes involved, distributed filesystems
- Reliability. Causes of errors. Tradeoffs of Checkpointing
- Energy, Energy Delay, Time, Performance



Aside on Sysadmin for a Cluster

- Always dangerous to update system mid-semester, things often break
 - Software package I installed broke MPI
 - Fixing MPI broke slurm
- Apparently you can write a CUDA program that can crash/hang the NVIDIA driver
 - Had to reboot the system
 - In theory OS should prevent this. Linux devels don't take bug reports on NVIDIA driver though because it's



proprietary

- Updated the driver, but that made the hw8 makefiles break
- You will need to add `-std=c++14` to the CFLAGS line in the Makefile
- Just installing software is a pain
 - Sourcecode from source, especially if by a grad student pushing things out as they graduate always tough to keep going
 - `pthread_yield()` turned to `sched_yield()` for some reason



- Can see where the flatpack people are coming from
- Issues with the pi cluster
 - Software isn't always well tested on ARM, especially HPC software



RAID5/6 notes from last time

- raid5, you need at least three drives. Data is striped across first two drives, third drive gets the bytes XOR with each other
- If a drive fails, you can xor the data from the two remaining drives to get the missing data
- This does waste some space. Also rebuilding can be stressful on remaining drives and cause them to fail
- RAID6 has an extra drive for redundancy, but reduces capacity



- Modern systems often do RAID10 (RAID1+0) which is striping plus full backup. Wastes half the space, but doesn't require XOR



Big Data Tools

- There are various
- Hadoop was one of the more popular



Hadoop

- A distributed filesystem (HDFS)
- A way to run map-reduce jobs



Hadoop

- Apache
- Distributed Processing and Distributed Storage on commodity clusters
- Java based
- Data spread throughout nodes
Large data sets split up and spread throughout the cluster
- Unlike traditional HPC clusters, code sent *to the nodes* that have data of interest, rather than taking data over



network to running code.

- HADOOP common – libraries
- HADOOP YARN – thread scheduling
- Hadoop Distributed File System – HDFS
- Hadoop MapReduce – processing algorithm
- Originally developed at Yahoo by Cutting and Cafarella.
Named after toy elephant.
- Many users. As of 2012 Facebook had 100PB of data,
said it grew at 0.5PB/day



Hadoop Distributed Filesystem

https://hadoop.apache.org/docs/r1.2.1/hdfs_design.html

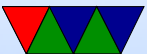
- Keeps working in face of hardware failures
- Streaming data access – optimize for bandwidth, not latency
- Relaxes some POSIX assumptions
- Large data sizes – optimized for files of gigabytes to terabytes
- Write-once-read-many – assumption is the data isn't being actively written.



- “Moving computation easier than moving data”
- blocksize and replication factor per-file
- Rack-aware filesystem
- “location awareness” Tries to spread code out multiple copies distributed physically
- Data spread throughout nodes. Default replication value of 3, duplicated three times, twice on same rack and once on different
- Namenode plus cluster of datanodes
- Namenode tracks filenames and locations, keeps entire map in memory



- Datanode stores data. Uses local computer's underlying filesystem. Just blocks of data, makes directories as appropriate but doesn't necessarily have any relationship to the files as seen from within HDFS.
- Communication is over TCP



HDFS Fault Handling

- Datanodes send heartbeats to namenode. When datanodes go missing, marked as dead, no new I/O sent to them. If any files fall below replication level they can be replicated on remaining nodes
- Rebalancing – if disk availability changes files might be moved around
- Integrity – checksums on files to detect corruption
- Namenode is a single point of failure. Keeps the edit log and fsimage, only syncs at startup



Data Organization

- Data broken up into chunks, default 64MB
- Creating a file does not necessarily allocate a chunk; it is cached locally and only sent out once enough data has accumulated to fill a block
- Replication pipeline: once file created starts being sent in smaller chunks (4kb) and it gets forwarded 1 to 2 to 3 in a pipeline until file in all places.
- Deleting a file does not delete right away, moved to /trash After configurable time gets deleted from trash



and the blocks are marked as free. It can take a while for this to all happen, deletes do not free up space immediately.

- Not a full POSIX filesystem. Writes are slow, and you can't write to an existing file.



Map Reduce

- Originally popularized by Google, but not really used by them anymore (after 2014)
Jeffrey Dean, Sanjay Ghemawat (2004) MapReduce: Simplified Data Processing on Large Clusters, Google.
- For processing large data sets in parallel on a cluster
- Similar to MPI reduce and scatter operations
- Map() – filters and sorts data into key/value pairs
Stateless, can run in parallel
can contain Combiner() – combines duplicates?



- Reduce() – the various worker nodes process each group in parallel.
Shuffle() – redistribute data so all common data on same node
- Can do with single processor systems, but not any faster typically. Shines on parallel systems



Map Reduce Example

The quick brown fox jumped over the lazy dog.

MAP split by key (in this case, number of letters)

3: [the, fox, the, dog]

4: [over, lazy]

5: [quick, brown]

6: [jumped]

REDUCE each thread/node gets one of these. Reduce might simply count.



3: 4

4: 2

5: 2

6: 1



Another Map Reduce Example: Hello World

This is the example they like to use.

Map: key is the word

To be or not to be, that is the question.

to: [1, 1]

be: [1, 1]

or: [1]

not: [1]

that: [1]

is: [1]



the: [1]
question: [1]

Reduce:

to: 2
be: 2
or: 1
not: 1
that: 1
is: 1
the: 1
question: 1



Real world friends example

- <http://stevekrenzel.com/finding-friends-with-ma>
- https://www.tutorialspoint.com/hadoop/hadoop_mapreduce.htm



Why would you do things like this?

- You can see how this comes out of search research
- Have terrabytes of spidered websites you want to do a text search on? (Maybe HPC?)
- Having one thread read all terabytes over the network to central location and searching, take forever
- Instead, data spread across millions of machines
- Send code that first does a map to find out how many times HPC occurs on each file
- Then reduce down to MAX and find out which are most



relevant



Submitting a Job

- Job:
 - Specify input and output on filesystem
 - The jar file (java class) of the map and reduce functions
 - Job configuration
- Hadoop client sends this to the scheduler



Scheduling

- Each location of system known. Try to run code on same system as data for locality, If not possible, run on one nearby.
- Small cluster has single master node, and multiple worker nodes.
- Hardware does not have to be fault tolerant; if a map/reduce fails it is simply retried again (on another machine)
- You can add/remove hardware at any time



Hadoop Update

Can set up Hadoop on single machine, even the name and data servers. Just download big chunk of Java, have Java and ssh installed.



Data Warehouse

- Enterprise Data Warehouse (EDW)
- Business gather data
- ETL: Extract, Transform, Load



Other Big Data codebases

- Google BigQuery
- Apache Spark
- Apache Storm



Google Big Query

- “serverless data warehouse”
- Petabytes of data
- “Platform as a service”
- SQL, Machine learning
- Import data as CSV, JSON, etc
- Use Google Dremel (for interactive querying of large databases)



Apache Spark

- Interface for programming clusters with data parallelism and Fault Tolerance (made at Berkeley)
- Resilient Distributed Datasets (RDD), read only multiset of data distributed over large cluster, fault tolerant
- Dataset API
- Replacement for Map Reduce / Hadoop, latency several orders of magnitude better
- Iterative algorithm can repeatedly visit
- Good for machine learning workloads



- Has a cluster manager
 - Native Spark
 - Hadoop Yarn
 - Apache Mesos
 - Kubertenes
- Uses distributed storage
 - Alluxio
 - HDFS
 - Casandra
 - Amazon S3
 - Openstack



- Kudu
- ?



More Apache Spark

- “HPC is dying and MPI is Killing it” article (2015)
- Java / Scala / Python / R
- Two components
 - Driver, converts code to multiple tasks
 - Executor: runs on nodes
- Originally ran on Hadoop Yarn, can also now via Kubertenes



Apache Spark – RDDs

- Resilient Distributed Dataset (RDD)
- Can be text, SQL, NoSQL, amazon s3 bucket
- Fault-tolerant, immutable (can't change) distributed set of objects, divided into logical partitions
- Creation/Transform/Act:
 - Create from file or bucket and parallelize with a command
 - Run a transform on it (sort of like map)
Doesn't update current RDD, but creates new one



- Run an action on it. Count, first, max, reduce, collect



Apache Spark Example

- Install it
- Run spark-shell
- Run spark-submit



Apache Storm

- Uses clojure (Lisp/Java)
- Distributed Stream Processing
- Distributed Process Stream Data
- Pass it Directed Acyclic Graph (DAG), “spouts” and “bolts” at vertices, edges are streams
- Master nodes execute daemon, Numbys
- Worker nodes



Apache Drill

- Clone of google dremel



Apache Impala

- Massively Parallel SQL query engine
- ?



Facebook Presto

- ?

