Data Centers

Kai Shen

Data Centers

- Large distributed systems of computers and storage (size of warehouses)
  - Host interactive web services
  - And also batch processing / data mining
- Can be at very large scale:
  - Up to 10s/100s of thousands of machines (millions of processors)
  - Megawatts or more in power usage
  - Serve millions of users (thousands at a time)
  - Host large amount of data: partitioning and replication

Data Center Hosting Web Search

Motivations for Extreme Scale

- Highly popular applications/services
- Big data
- Cost efficiency at scale (buying bulk is cheaper)
  - Resources to applications can be elastically managed
  - Repair and maintenance is centralized
  - Mistakes from individual incidents are learned and used to improve management at scale
Data Center Architecture

- Earlier days of parallel computing
  - Scientific computing is parallelizable, but complex synchronization glue the parallel processes
  - Distributed databases are also tightly coupled
  - Many points of failure in commodity machine clusters
  - Supercomputers with custom network and node design: close integration but expensive, hard to scale
- Late 90s, early 2000
  - Embarrassingly parallel web services: web search, e-commerce
  - Commodity machines and networks, glued together by software
  - Turn independent failures from a curse to a strength, migrate a server farm
  - Cheaper at scale since the software cost is amortized

Network Fabric

- Network matters to distributed data processing
- A Gbps network switch allows simultaneous traffic at full link speed
  - Crossbar: Complexity grows up quickly with the number of ports

- Switch with a small number of ports is cheaper (~$30 per port)
- Cost goes up non-linearly (an order of magnitude higher per port cost)
- One design choice
  - Cheap, few-port switch to link machines on a rack
  - Expensive, many-port switch to link all the racks together

Network Switching at Data Centers

- Thousands of servers with hierarchical network switching

- Reduce congestions:
  - Network load balance across nodes
  - Place highly communicating server modules within same switch domain, or better yet, in same machines
  - Redundant backbone links → Equal-Cost Multi-Path (ECMP) routing

Storage Architecture

- Each machine has local storage, managed by distributed file system such as GFS (Colossus), HDFS
- In addition (alternatively), storage-level parallelism
  - specialized storage systems plugged into the network switch that manages replication, parallel I/O etc. (scalable RAID system)
- Contrast
  - Storage-level parallelism generally provides better I/O performance
  - File system parallelism integrates better with higher-level software (such as MapReduce), facilitates data locality through smart software management, and puts less stress on network
Memory/Storage Hierarchy

- CPU cache, memory, Flash SSD, and disk
  - Larger space but longer latency
- In a data center, we have CPU cache, local memory, local disk, rack memory, rack disk, data center memory, data center disk
  - Figures 1.3, 1.4 (page 10)

Data Center Workloads

- Unique, large jobs
  - E.g., compute page ranks from the 50B crawled web pages
- Many, small tasks
  - E.g., Amazon cloud computing, Google search service
- Parallelism
  - Data parallelism
  - User / request parallelism

Platform / Application Adaptation

- Platform: basic hardware and infrastructure system software
- Applications developed on top
- Platform / application adaptation
  - Platforms are somewhat homogeneous
  - Application upgrades are fast
  - Focus on basic cost-effectiveness on the platform construction and let application software adapt to the platform

Infrastructure Software: Buy, Adapt, Build

- Infrastructure software: operating system, file systems, databases, parallel computation, replication, ...
- Choices:
  - Buy from vendors
  - Take open-source software and make adaptation
  - Build new software from scratch
- Google’s argument: adapt or build, but don’t buy
  - Need full control (performance debugging), flexibility, customization (nosql)
  - Few vendors (think of databases) are capable of building and testing their software at the datacenter scale
Cost-Efficient Analysis on Multiprocessor

- You have N parallel tasks
  - \( c \) is the computation time, \( d \) is the number of data accesses, \( l_{\text{local}} \) is the latency of local data access, \( l_{\text{remote}} \) is the latency of remote data access
  - If each task runs on a separate machine
    - Assume data is evenly distributed, the chance of a data access to be local is \( 1/N \)
    - Then the execution time is: \( c + d \cdot \left( l_{\text{local}} \cdot \frac{1}{N} + l_{\text{remote}} \cdot (1-1/N) \right) \)
  - If you have a large multiprocessor (host all N tasks):
    - Then the execution time is: \( c + d \cdot l_{\text{local}} \)

In practice, your multiprocessor isn’t large enough to host all (host \( n < N \) tasks)

- Then the execution time is: \( c + d \cdot \left( l_{\text{local}} \cdot \frac{n}{N} + l_{\text{remote}} \cdot (1-n/N) \right) \)

Figure 3.2 (page 36)
Reliability at Scale

- Why does a machine fail?
  - Hardware problems, software problems
- Reliability rate for one machine is p, for n machines is $p^n$
- In a very large system, you may not always be able to find out problem cause and fix the problem, so
  - Detect problem and tolerate/mitigate it
- Redundancy/replication
  - Memory ECC
  - Machines
  - Data storage
    - Replication of mutable states introduces consistency problems
  - Networking
  - Power supply

Management

- Easy to add/remove components
  - Container-based data center modules
Disclaimer

- Preparation of this class was helped by materials in the book “The Datacenter as a Computer” by Luiz A. Barroso, Jimmy Clidaras, and Urs Holzle.