Paper review

The Datacenter as a Computer

I. Problem and why it's important

The trend of server-side computing brought benefits to both users and vendors. In addition to greater user-side management and accessibility, it makes it easier for vendors to manage resources and make improvements. On the other hand, this book also introduces the features and great challenges of scaling up warehouse-scale computers to accommodate the accelerating demand for computing capabilities. Physical (discrepancies in latency, bandwidth and capacity among servers) and economical constraints (networking fabric and storage) limit cost efficiency, availability and fault-tolerance. This paper covers the high-level DC design including storage, networking fabric, power usage and fault handling. Chapter 2 describes the key characteristics of cloud software deployment and challenges such as workload churn (harness parallelism), tail latency, health monitoring, etc. The “coarse-grained” importance of this book is due to the significant role that server-side computing is playing today.

II. Hypothesis

The paper mainly focused on description of datacenter design and evolution, especially WSCs. The core idea is that a datacenter should behave just like a desktop machine. To achieve this, Chapter 1 hypothesized that the general DC architecture tends to be stable so that we can discuss the current challenges of each DC component. Chapter 2 introduces the software layers in WSC (platform-level software, cluster-level infrastructure, and application-level software).

III. Proposed solution and key insights

Chapter 1 introduces DC design solutions by each major component. Chapter 2 discussed the concepts of software deployment and toolbox.

- In terms of storage, disks can be separated (NAS) from compute nodes for simpler outsourcing and QoS guarantees, or attached (DFS) to compute nodes for higher hardware and networking fabric utilization. Those two approaches focused on different scenarios.
- To smooth out the discrepancies of latency, bandwidth and capacity among different servers with a WSC, for example, today’s WSC architects gradually deploy Flash (SSD) to balance the cost and performance.
- Another challenge is the high failure rate of different components. At application level common techniques such as replication and erasure coding are used to tolerate failures.
- Common metrics/goals required by applications on WSCs include performance, consistency, availability, etc. Table in section 2.2 is a great list of fundamental but important techniques.

IV. Drawback, limitation and improvement

Most of the drawbacks or tradeoffs mentioned in the first 2 chapters are further discussed in later chapters. So here we selected part of the tradeoffs in the first two chapters:

- The high performance of Flash in datacenters creates new bottlenecks and demands higher bandwidth from the fabric; its random write performance is much slower than read (although many software-level techniques are developed e.g. LSM)
- Reed-Solomon codes are saving space and improving availability but requires higher CPU computation
- Eventual consistency might not be strong enough for some scenarios where data freshness is highly required

V. Gist of discussion

Critical resources in a datacenter: they vary from scenario to scenario

- Generally power usage and cost are critical to datacenter maintenance
- A GPU-heavy workload is very likely to be memory-intensive
- Bandwidth is quite important, including network and SSD IOPS if huge data transmission is required
- Although hardware might bring groundbreaking improvements, softwares are often the weakest point but of great potential.

Datacenters tend to balance the usage of different resources for sustainable performance.

Paper review

Jupiter Rising: A Decade of Clos Topologies and Centralized Control in Google's Datacenter Network

I. Problem and why it's important

Problem: described the authors' experience with building five generations of custom data center network hardware and software leveraging commodity hardware components, while addressing the control and management requirements introduced by our approach.

The problem is of significance because traditional datacenter network features were not useful, but increased cost, complexity, delayed time to market, and made network management more difficult. The scale of Jupyter expects robust and graceful reactions to highly frequent network events, requiring the fabric to support heterogeneous hardware and speeds .

II. Hypothesis

Inspired by the community's ability to scale out computing with parallel arrays of commodity servers, the authors proposed a similar approach for networking.

III. Proposed solution and key insights

The authors built a centralized control mechanism based on a global configuration pushed to all datacenter switches.

- Adopted multi-stage Clos topologies to support graceful fault tolerance, increase the scale/bisection of our datacenter networks, and accommodate lower radix switches.
- Applied bandwidth-dense but feature-limited merchant silicon because the authors emphasized bandwidth density and frequent refresh cycles to keep pace with server bandwidth demands and Moore's Law.
- Collected and distributed dynamically changing link state information from a central, dynamically-elected, point in the network as an adaptation to Clos topologies.

IV. Drawback, limitation and improvement

- Routing would experience a huge churn on an entire fabric reboo. In this case, the liveness protocol would not be able to respond to heartbeat messages in time. This could create a snowball effect for routing where link state would spuriously go from up to down and back to up again.
- Aging hardware led to degradation of inbuilt fabric. This could result in unidirectional failures of some CPN links which eventually lead to unhandled corner cases in our routing protocols.

- The fabric didn’t implement locking to prevent simultaneous read/write access to the BGP configuration. As a consequence, the partial configuration would lead to undesirable behavior between Freedome and its BGP peers.

V. Gist of discussion

Inspired by the success of large-scale distributed storage systems with a centralized manager, the authors treated the datacenter network as a single fabric with tens of thousands of ports.
A top-down view was selected over a bottom-up approach, which regarded the entire cluster network as a single static fabric composed of switches with pre-assigned roles. The authors limited the number of choices at the cluster-level and provided a simple menu of fabric sizes and options based on the projected maximum size of a cluster as well as the chassis type available. Little overhead is introduced in the distribution of a single configuration to all switches in the cluster.