Changing Architectures: New Datacenter Networks Will Set Your Code and Data Free

Tuesday, September 4, 2012 at 9:15AM

Todd Hoff in Strategy

One consequence of IT standardization and commodification has been Google's datacenter is the computer view of the world. In that view all compute resources (memory, CPU, storage) are fungible. They are interchangeable and location independent, individual computers lose identity and become just a part of a service.

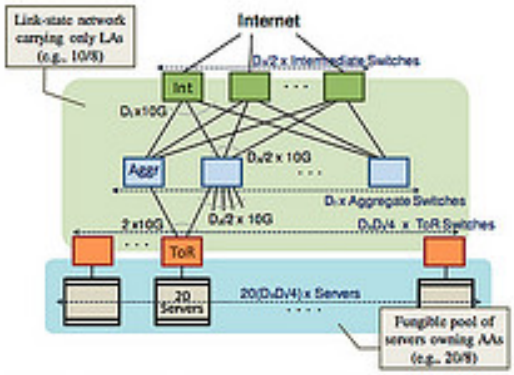

Figure 5: An example Clos network between Aggregation and Intermediate switches provides a richly-connected backbone well-suited for VLB. The network is built with two separate address families — topologically significant Locator Addresses (LAs) and flat Application Addresses (AAs).

Thwarting that nirvana has been the abysmal performance of commodity datacenter networks which have caused the preference of architectures that favor the collocation of state and behaviour on the same box. MapReduce famously ships code over to storage nodes for just this reason.

Change the network and you change the fundamental assumption driving collocation based software architectures. You are then free to store data anywhere and move compute anywhere you wish. The datacenter becomes the computer.

On the host side with an x8 slot running at PCI-Express 3.0 speeds able to push 8GB/sec (that's bytes) of bandwidth in both directions, we have enough IO to feed Moore's progeny, wild packs of hungry hungry cores.

And in the future System on a Chip architectures will integrate the NIC into the CPU and even faster speeds will be possible. Why we are still using TCP and shoving data through OS stacks in the datacenter is a completely separate question.

The next dilemma is how to make the network work. The key to bandwidth nirvana is explained by Microsoft in MinuteSort with Flat Datacenter Storage, which shows how in a network with enough bisectional bandwidth every computer can send data at full speed to every computer, which allows data to be stored remotely, which means data doesn't have to be stored locally anymore.

What the heck is bisectional bandwidth? If you draw a line somewhere in a network bisectional bandwidth is the rate of communication at which servers on one side of the line can communicate with servers on the other side. With enough bisectional bandwidth any server can communicate with any other server at full network speeds.

Wait, don't we have high bisectional bandwidth in datacenters now? Why no, no we don't. We typically have had networks optimized for sending traffic North-South rather than East-West. North-South means your server is talking to a client somewhere out in the Internet. East-West means you are talking to another server within the datacenter. Pre cloud software architectures communicated mostly North-South, to clients located outside in the Internet. Post cloud most software functionality is implemented by large clusters that talk mostly to each other, that is East-West, with only a few tendrils of communication shooting North-South. Recall how Google has pioneered large fanout architectures where creating a single web page can take a 1000 requests. Large fanout architectures are the new normal.

Datacenter networks have not kept up with the change in software architectures. But it's even worse than that. To support mostly North-

South traffic with a little East-West traffic, datacenters used a tree topology with core, aggregation, and access layers. The idea being that the top routing part of the network has enough bandwidth to handle all the traffic from all the machines lower down in the tree. Economics made it highly attractive to highly oversubscribe, like 240-1, the top layer of the network. So if you want to talk to a machine in some other part of the datacenter you are in for a bad experience. Traffic has to traverse highly oversubscribed links. Packets go drop drop fizz fizz.

Creating an affordable high bisectional bandwidth network requires a more thoughtful approach. The basic options seem to be to change the protocols, change the routers, or change the hosts. The approach Microsoft came up with was to change the host and add a layer of centralized control.

Their creation is fully described in VL2: A Scalable and Flexible Data Center Network:

> A practical network architecture that scales to support huge data centers with uniform high capacity between servers, performance isolation between services, and Ethernet layer-2 semantics. VL2 uses (1) flat addressing to allow service instances to be placed anywhere in the network, (2) Valiant Load Balancing to spread traffic uniformly across network paths, and (3) end-system based address resolution to scale to large server pools, without introducing complexity to the network control plane.

The general idea is to create a flat L2 network using a CLOS topology. VMs keep their IP addresses forever and can move anywhere in the datacenter. L2 ARP related broadcast problems are sidestepped by changing ARP to use a centralized registration service to resolve addresses. No more broadcast storms.

This seems strange, but I attended a talk at Hot Interconnects on VL2 and the whole approach is quite clever and seems sensible. The result delivers the low cost, high bandwidth, low latency East-West flows needed by modern software architectures. A characteristic that seems to be missing in Route Anywhere vSwitch type approaches. You can't just overlay in performance when the underlying topology isn't supportive.

Now that you have this super cool datacenter topology what do you do with it? Microsoft implemented a version of the MinuteSort benchmark that was 3 times faster than Hadoop, sorting nearly three times the amount of data with about one-sixth the hardware resources (1,033 disks across 250 machines vs. 5,624 disks across 1,406 machines).

Microsoft built the benchmark code on top of the Flat Datacenter Storage (FDS) system, which is distributed blob storage system:

> *Notably, no compute node in our system uses local storage for data; we believe FDS is the first system with competitive sort performance that uses remote storage. Because files are all remote, our 1,470 GB runs actually transmitted 4.4 TB over the network in under a minute*
>
> *FDS always sends data over the network. FDS mitigates the cost of data transport in two ways. First, we give each storage node network bandwidth that matches its storage bandwidth. SAS disks have read performance of about 120MByte/sec, or about 1 gigabit/sec, so in our FDS cluster a storage node is always provisioned with at least as many gigabits of network bandwidth as it has disks. Second, we connect the storage nodes to compute nodes using a full bisection bandwidth network—specifically, a CLOS network*

> *topology, as used in projects such as Monsoon. The combination of these two factors produces an uncongested path from remote disks to CPUs, giving the system an aggregate I/O bandwidth essentially equivalent to a system such as MapReduce that uses local storage. There is, of course, a latency cost. However, FDS by its nature allows any compute node to access any data with equal throughput.*

Details are in the paper, but as distributed file systems have become key architectural components it's important for bootstrapping purposes to have one that takes advantage of this new datacenter topology.

With 10/100 Gbps networks on the way and technologies like VL2 and FDS, we've made good progress at making CPU, RAM, and storage fungible pools of resources within a datacenter. Networks still aren't fungible, though I'm not sure what that would even mean. Software Defined Networking will help networks to become first class objects, which seems close, but for performance reasons networks can never really be disentangled from their underlying topology.

What can we expect from these developments? As fungibility is really a deeper level of commoditization we should expect to see the destruction of approaches based on resource asymmetry, even higher levels of organization, greater levels of consumption, the development of new best practices, and even greater levels of automation should drive even more competition in the ecosystem space.

Related Articles

- A Guided Tour through Data-center Networking
- Data in the Fast Lane

- Minutesort with Flat Datacenter Storage
- Minutesort with Flat Datacenter Storage by Andrew Wang (On Reddit)
- A quick summary of the VL2 data-center network scheme
- FULL MESH IS THE WORST POSSIBLE FABRIC ARCHITECTURE
- Microsoft's FDS data-sorter crushes Hadoop
- MINUTESORT WITH FLAT DATACENTER STORAGE
- Explaining L2 Multipath in Terms of North/South, East West Bandwidth
- VL2: A Scalable and Flexible Data Center Network by by MS Researc
- VL2: A Scalable and Flexible Data Center Network by Murat Demirbas
- Jellyfish: Networking Data Centers Randomly
