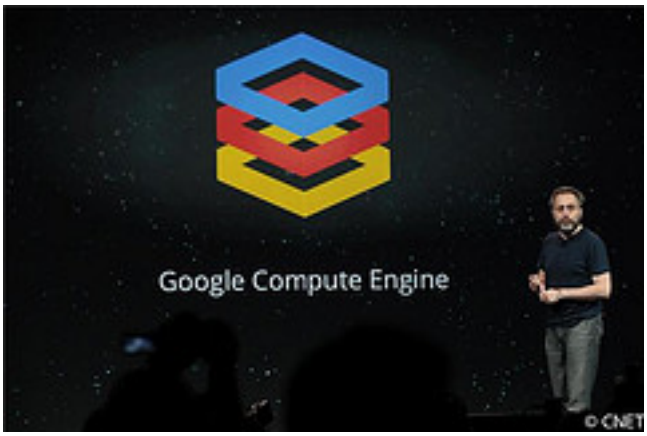
C is for Compute - Google Compute Engine (GCE)

Monday, July 2, 2012 at 9:56AM

Todd Hoff in cloud, google



After poking around the Google Compute Engine (GCE) documentation I had some trouble creating a mental model of how GCE works. Is it like AWS, GAE, Rackspace, just what is it? After watching Google I/O 2012 - Introducing Google Compute Engine and Google Compute Engine -- Technical Details, it turns out my initial impression, that GCE is disarmingly straightforward, turns out to be the point.

The focus of GCE is on the C, which stands for Compute, and that's what GCE is all about: deploying lots of servers to solve computationally hard problems. What you get with GCE is a Super Datacenter on Google Steroids.

If you are wondering how you will run the next Instagram on GCE then that would be missing the point. GAE is targeted at applications. GCE is targeted at:

- Delivering a proven, pure, high performance, high scale compute infrastructure using a utility pricing model, on top of an open, secure, extensible Infrastructure-as-a-Service.
- Delivering an experience that feels like you are in a datacenter and not at creating a massively multi-tenant cloud.

- Allowing you to become Google. Tackle the same problems Google tackles with the same infrastructure, minus all the data and people of course.
- Standing up VM instances quickly, do your work, and tear them down quickly.
- Performing better and better as cluster gets bigger. Google considers large clusters to start at 10-20K instances.
- Being a compute utility. You get resources affordably because of Google's efficiency at scale.
- Consistent performance. Google has pioneered consistent performance at scale and they are making a huge deal of this and it's mentioned several times in the demos. GCE is tuned for both high and consistent performance throughout the stack. The idea is you don't have to design for unstable or inconsistent system, so don't have to design for worst case. This allowed some customers to reduce their number of cores in half.
- Giving you a set of servers you can run anyway you want.
- Creating a technology you can bet your business on. Google is running Google business on the stack today.

Basic Overview of GCE

- Customers
  - Targeted at problems using large compute jobs, batch workloads, or that require high performance real-time calculations. Not building websites. In the future they plan on adding more features like load balancing.
  - Right now it's about work that can be parallelized. Will provide vertical scaling in the future, that is 32+ cores.
  - Seem to want enterprise customers that can make use of lots of cores, not little guys.
- Datacenters

    - Region: for geography and routing domain.
    - Zone: for fault tolerance
    - Currently operating 3 US datacenters/zones, located on the East coast of the US.
    - Working on adding more datacenters globally and adding more datacenters in the US.
- API
    - JSON over HTTP API, REST-inspired, authorization is with OAuth2
    - Main resources: projects, instances, networks, firewalls, disks, snapshots, zones
    - Actions GET, POST (create), DELETE, custom verbs for updates
    - A command line tool (gsutil), a GUI, and a set of standard libraries gives access to the APIs. Experience is like Amazon in that you have an UI and command line tools.
    - All Google tools use the API. There is no backdoor. The web UI is built on Google App Engine, for example. App Engine is the web facing application environment and is considered an orchestration system for GCE.
    - Partners like RightScale, Puppet, and OpsCode, also use the API to provide higher level services.
    - Want people to take their code and run it on their infrastructure. Open API. No backdoors. Can extend that stack at any level.
- Project
    - Everything happens within the context of a Project: team membership, group ownership, billing. A Project is a container for a set of resources that are owned by the Project and not by people. Every API action is traced back to a person instead of a credential.
- Service Account

    - Synthetic identity acting as a user when performing operations in code. Connects seamlessly with GAE, Cloud Storage, Task Queues, and other Google services.
    - When launching a VM an OAuth2 scope is provided that is stored in a special metadata server that is used transparently between services. No configuration or password is required.
- Virtual Machine
    - Linux virtual machines with root access. For security and performance reasons the kernel is locked down. The kernel is tuned to work with their networking environment.
    - Two stock versions of Linux: Ubuntu and CentOS. They say you can run whatever Linux distribution you want, but I'm not sure how that fits with locked down kernel policy.
    - Comes installed with gsutil, turned off password authentication so only use ssh authentication is used, turned on automatic security updates.
    - High performing 2.6 GHz Intel Sandy Bridge processor.
    - Available in 1,2,4, or 8 virtual CPUs. Each virtual CPU is mapped to a hyperthread. For a 2 CPU instances you get both halves of a real physical core.
    - 3.7GB RAM per core. 420GB local/ephemeral storage.
    - 8 core instances have dedicated spindles. You are the only one reading and writing from the disk, so you have more predictable/consistent performance.
    - Invented performance unit: the Google Compute Engine Unit (GQ). Roughly matches Amazon's compute unit. Each virtual CPU is rated at 2.75 GQs.
    - Smaller machines will be available for prototyping and debugging.
    - Big boxes because focussed on high performance computing.
- Instances
    - A combination of KVMs (Kernel Virtual Machines) and

Linux cgroups are used for the underlying hypervisor technology. Linux scheduler and memory manager are reused to handle the scheduling of the machines.
    - KVM provides virtualization. Cgroups provides resource isolation. Cgroups was pioneered by Google to keep workloads isolated from each other.
    - Internally Google can run virtualized and non-virtualized workloads on the same kernel and on the same machine, which allows them to deploy and test one single kernel.
    - Located in a zone.
    - Fast boot times: 2 minutes.
- Instance Metadata
    - Solving the configuration problem to customize VMs at boot time.
    - A dictionary of key-value pairs are available on the instance via a private HTTP metadata server just for that machine. This metadata can be set for the instance to control its boot/ configuration/role process. Can be read using curl.
    - Project wide metadata is also available that is inherited by all instances. Used to push SSH keys into VM at boot time. A default image knows how to read a special bit of metadata called SSH Keys and then installs them into the VM.
- Startup Scripts
    - Simple bootstrapping scripts, similar to rc.local, that run on boot.
    - Use to install software and start other software.
- Service Orientation, not Server Orientation
    - Build across zones to deal with failure.
    - Use startup scripts and metadata for automatic configuration.
    - Use local disk as a cache or scratch area.
    - Build automation using GAE or their partners.
- Networking - VPN

    - Google considers their network a distinguishing feature. It features high cross sectional bandwidth, that is, machines can talk more directly to each other without competing with neighboring traffic on a bus. This reduces network latency and increases the consistency of performance. They won't publish any numbers though.
    - Each project gets its own secure VPN that is unshared with anyone else. Spans across all your VMs, no matter where they are.
    - Networking traffic does not transit the Internet. It is routed over Google's secure, high performance private network.
    - Network is all L3 using private IP addresses that are guaranteed to come from a machine on your VPN.
    - VM name = DNS name. VMs have normal looking hostnames that you can assign and use the DNS to find. This is very convenient when bringing up an arbitrary set of hosts.
    - IPv6 in the future.
    - You can have many VPNs per project, but by default there is one called default that is used by default.
    - Broadcast and multicast are not supported, which if you have a VPN removes a lot of interesting architectures. Maybe with v6?
- Networking - Internet
    - Traffic from the Internet to your machine is shunted on to Google's private network as soon as they can and given a "first class" ticket to your VPN. This is like an overlay network you see on CDNs.
    - 1-to-1 NAT. Every VM can be assigned an external IP address that is rewritten as it enters and exits your VPN. They don't exist on the VM when you do an ifconfig.
    - IP addresses can be detached from a VM in one region and attached to a VM in another region and Google will make sure

the traffic is routed properly.
    - Built in firewall to control who talks to what in the system.
    - Can't use SMTP. Only UDP, TCP, and ICMP can be used to the Internet.
    - IP addresses are advertised with Anycast, then they encapsulate it, and then forward it to your VPN.
- Storage
    - Focused on creating persistent block device that offers performance / throughput so you don't need to push storage local.
    - Two block storage devices: Persistent Disk and Local Disk.

- Persistent disk
    - Off instance durably replicated storage medium. High consistency. High throughput solution. Secure. Backing store for database. Built from scratch to be highly performant and gives good 99.95 percentile performance.
    - Allocated to a zone.
    - Can be mounted read/write to a single instance or read only to a set of instances.
    - Data is transparently encrypted when it leaves your VM, before it is written to disk. Using new processors there's very little to no overhead. It seems to use Google keys and not your keys.
    - Less than 3% variance in IO bandwidth when doing 4K random reads and writes. This is their consistency theme. Less variance than a local disk, which can vary by 13%.
    - For large block read and writes there's triple the local bandwidth compared to local disk.
- Local/ephemeral disk
    - Ephemeral on reboot. When the VM goes away the data goes away.

    - It’s encrypted using a VM specific key.
    - Currently all instances boot of off local disk, looking to boot off of persistent disk in the future.
    - 3.5TB with the 8 CPU instance.
    - With larger instances (4-8 core) you get dedicated spindles. One spindle with the 4 core instance and 2 spindles with the 8 core instance.
- Google Cloud Storage
    - Enterprise grade Internet object store.
    - HTTP API for getting and setting values.
    - Don’t have to worry about managing data. Replication is happening for you.
    - Publicly readable objects are cached close to where they will be used. Sounds a bit like a CDN. Data will be replicated to where it is needed and available quickly.
    - Uses Google global high performance internet backbone.
    - Read your writes consistency.
    - Bulk data. Useful for getting data in and out of Google’s cloud using Google’s high capacity pipes.
- Pricing
    - 50% more compute for your money when compared to AWS.
    - Billed on demand by the hour.
    - SLA and support open to commercial customers.

Examples of GCE Usage

Invite Media

Runs a real-time ad exchange that has a very high volume of traffic, 400K QPS, and as with all real-time markets requires consistently low latencies, 150ms end-to-end, in order to calculate the best deals. For each ad request they have time budget of 10ms to find a backend server to

serve the request and establish a connection.

Found the GCE model familiar. You have Linux VMs, you have disks, you can assign static IPs, create startup scripts, and have a nice API. Took two weeks to port their system to GCE.

Comparing existing provider with GCE, using 8 core instances:

- 350 QPS vs 650 QPS (while respecting latency requirements)
- 284 machines vs 140 machines
- 5% connection errors vs < .05%
- 11% of requests timed out vs 6% - means 5 percent more ad requests they can buy for advertisers

Decided to migrate entire operation to GCE.

Hadoop on GCE

This is example code created by someone at Google and will be released in the future.

- Can run from command line or GAE.
- Launch a coordinator has an API to set up all the other VMs in the cluster (100 nodes), monitor, etc.
- Booting from a fresh Ubuntu image the setup was pretty fast. The coordinator installs Hadoop and launches nodes. Took a while, but relatively quick.
- Launched a job on Hadoop master to process 60GB of compressed wikipedia revision history. Slices data in CSV format. Took 1.5 minutes writing 70GB of data.
- The CSV is piped into Big Query to answer questions like which wikipedia article had the most edits, who are the top editors, and

other interactive questions.

Video Transcoding

This is very common cloud demo.

1. Video loaded into a job queue.
2. Consumers, and you can run a lot on GCE, take job and perform the transcode.
3. Transcoded video is sent to the Google storage service.

MapR on Terasort

MapR ran the Terasort benchmark on a 1250 node cluster in 1:20 minutes at a cost of $16. This was near record performance and they estimate to buy the same hardware to run the test locally would cost nearly $6 million.

They found GCE blazing fast with great disk drive disk and network bandwidth. They were able to provision thousands of VMs in minutes

BuildFax

Put their database and production servers on GCE. They are very pleased with the consistent performance. Their service delivers insurance related data points to customers at the time they write policies. Results were returned in less than 4 seconds with a very low variance. Again, this is the consistent performance claim.

Observations

1. With GCE Google has designed an experience familiar to Amazon users, with some nice second system improvements in configuration and operations, and a lot of special Google sauce in performance.

2. Better late than never. GCE is late to the game, but it has a strong performance, pricing, and development model story that often helps with customers wins over first to market entrants. If you need huge scale and/or great performance then why wouldn't you consider GCE? Performance requires carefull design from the start. It's hard to add in later. And after all of Google's bragging about their cool infrastructure this is your chance to give it a spin and see what it is made of.
3. Kind of bummed that it's not targeted more at front facing websites. There's no reason you can't run a website in GCE it seems, but unlike AWS you won't get a lot of help. Like in the early days of EC2 it's all up to you, but that's probably OK for a lot of people.
4. As Google deals with more and more customers can they maintain quality? As we've seen, most things go bad when problems occur and a lot of traffic is flowing through the system. Shared state is the system killer and Google still has plenty of that. Google has yet to test their cloud infrastrucure in this way.
5. Where will egress pricing end up once the low promotional pricing ends? Google lockin will occur if it's expensive to transfer your data out of Google's cloud. Google pricing in general is a bit scary.
6. Will AWS Direct Connect be avaialble to GCE?
7. Is GCE a target for migration or integration? BigData jobs are an obvious target for GCE, but we've also seen examples where real-time services benefit from GCE, so running a few select services in GCE might be a good toe in the water strategy. Concerns over data transfer costs are part of the ecosystem lockin play. Resilience alone however argues for implementing systems in more than one cloud.
8. Amazon has a huge advantage in services. Will Google go upstack as Amazon has done? Or is this your cloud equivalent of a chance to tap the Android market while everyone else is creating apps for theiPhone?

Related Articles

- On Hacker News
- Google IO Videos
