Google Cloud

How Google collected these benchmark results

Results in the blog post were collected using comparable latest-generation generally available Intel-based VMs with general purpose storage types from Google Compute Engine, Amazon Web Services and Azure to keep performance comparison close to being apples to apples. We selected VMs offering 16 vCPUs and ~64 GB memory with remote disks balancing performance and cost or 8 vCPUs and ~32 GB memory.

The VMs were provisioned with PerfKitBenchmarker (PKB), an open source benchmark suite including applications deployed in the cloud. PKB is used internally at Google and aims to fairly compare offerings by public clouds. To get started with PKB, we recommend starting with this walkthrough.

MySQL

This benchmark evaluates the performance of MySQL by running the HammerDB implementation of a TPC-C (OLTP) like workload, TPROC-C, and measuring the transactions per minute (TPM). To run the MySQL benchmark used in the blog on VMs offered by different cloud providers with PKB, vary these flags below. We showed results for the following machine and remote disk combinations:

Machine and Remote Disk	**Cloud (--cloud=)**	**Zone (--zone=)**	**Server Machine (--db_machine_type=)**	**Client Machine (--client_vm_machine_type=)**
c4-standard-16 with 1x hyperdisk-balanced disk provisioned with 40000 iops and 1200 MB/s throughput	GCP	us-east4-c	c4-standard-16	c4-standard-32
m7i.4xlarge with 4x striped gp3 disks each provisioned with 10000 iops and	AWS	us-east-1a	m7i.4xlarge	m7i.8xlarge

300 MB/s throughput				
Standard_D16s_ v5 with 1x Premium SSD v2 disk provisioned with 40000 iops and 1200 MB/s throughput	Azure	westus3[1]	Standard_D16s_ v5	Standard_D32s_ v5

The VMs and remote disks were chosen with the following in mind:

1) Use standard general-purpose VMs with 16 vCPUs and ~4 GB of memory per vCPU.
2) Use disks balancing performance and cost; Hyperdisk-Balanced on GCP, GP3 on AWS, and Premium SSD V2 disks are priced very similarly.
3) Provision remote disk IOPS throughput and capacity similarly. Each disk is 2 TB and supports 40K IOPS and 1200 MB/s throughput after disk striping. While a 16 vCPU C4 VM can support more than this, AWS and Azure's general purpose VMs do not, and these limits are high enough to not be a bottleneck with this benchmark configuration[2].

The following PKB command line provisions a GCP c4-standard-16 database server machine with an appropriate remote disk and a c4-standard-32 client machine for HammerDB in us-east4-c and produce MySQL results:

```
./pkb.py --benchmarks=mysql_tpcc --cloud=GCP --client_vm_disk_type=hyperdisk-balanced
--client_vm_machine_type=c4-standard-32
--config_override=mysql_tpcc.relational_db.vm_groups.servers.vm_spec.GCP.boot_disk_type
='hyperdisk-balanced'
--config_override=mysql_tpcc.relational_db.vm_groups.clients.vm_spec.GCP.boot_disk_type='
hyperdisk-balanced' --db_disk_size=2000 --db_disk_type=hyperdisk-balanced
--db_machine_type=c4-standard-16 --os_type=ubuntu2004 --project=<YOUR_PROJECT>
--provisioned_iops=40000 --provisioned_throughput=1200 --zone=us-east4-c
```

For AWS[3]:

```
./pkb.py --benchmarks=mysql_tpcc --cloud=AWS --client_vm_machine_type=m7i.8xlarge \
  --db_disk_size=500 --db_disk_type=gp3 --db_machine_type=m7i.4xlarge \
  --db_num_striped_disks=4 \
  --os_type=ubuntu2004 --provisioned_iops=10000 \
  --provisioned_throughput=300 --zone=us-east-1a
```

For Azure:

[1] This zone was chosen for increased likelihood of receiving Sapphire Rapids VMs.
[2] Verified with iostat. An iops-bound MySQL workload would require a different benchmark configuration with different VMs (e.g. r6in on AWS, Ebsv5 on Azure).
[3] On AWS gp3 disks only support up to 16k iops, so we stripe 4 of them together for a similar disk to the ones created on GCP and Azure.

```
./pkb.py --benchmarks=mysql_tpcc --cloud=Azure --azure_accelerated_networking=true \
  --client_vm_machine_type=Standard_D32s_v5  --db_disk_size=2000
--db_disk_type=PremiumV2_LRS \
  --db_machine_type=Standard_D16s_v5 \
  --os_type=ubuntu2004 --provisioned_iops=40000 --provisioned_throughput=1200 \
  --required_cpu_version=GenuineIntel_6_143_8 --zone=westus3-3
```

Redis

This benchmark evaluates the performance of a throughput optimized sharded Redis configuration with one Redis server per vCPU and measures throughput in operations per second (Total Ops Throughput). Memtier Clients run on separate VMs in the same availability zone and round-robin their requests across the servers. The databases are populated with enough data to ensure data is resident outside of cpu cache.

Machine[4]	**Cloud (--cloud=)**	**Zone (--zone=)**	**Machine (--machine_type=)**
c4-standard-8	GCP	us-east4-c	c4-standard-8
m7i.2xlarge	AWS	us-east-1a	m7i.2xlarge
Standard_D8s_v5	Azure	westus3	Standard_D8s_v5

To reproduce the Redis results using PKB, you can run a single command line. The following command line will provision a GCP c4-standard-8 database server machine and 2 c4-standard-8 client machines for Memtier in us-east4-c and produce redis results:

```
./pkb.py --benchmarks=redis_memtier_full_house_loaded --cloud=GCP \
--machine_type=c4-standard-8 \
  --config_override=redis_memtier_full_house_loaded.vm_groups.clients.vm_count=2 \
  --memtier_key_maximum=4125000 --memtier_load_key_maximum=3375000 \
  --os_type=ubuntu2004 --project=<YOUR_PROJECT> --zone=us-east4-c
```

For AWS:

```
./pkb.py --benchmarks=redis_memtier_full_house_loaded --cloud=AWS \
--machine_type=m7i.2xlarge \
  --config_override=redis_memtier_full_house_loaded.vm_groups.clients.vm_count=2
--memtier_key_maximum=4400000 \
  --memtier_load_key_maximum=3600000 \
  --os_type=ubuntu2004 --zone=us-east-1a
```

For Azure:

```
./pkb.py --benchmarks=redis_memtier_full_house_loaded --cloud=Azure \
--azure_accelerated_networking=true --machine_type=Standard_D8s_v5  \
--config_override=redis_memtier_full_house_loaded.vm_groups.clients.vm_count=2 \
```

[4] We do not specify disks since disk is not a bottleneck; data is resident in memory or evicted.

```
--memtier_key_maximum=4400000 --memtier_load_key_maximum=3600000 \
--os_type=ubuntu2004 --required_cpu_version=GenuineIntel_6_143_8 --zone=westus3-3
```

Regarding memtier_key_maximum and memtier_load_key_maximum, these are computed by multiplying the VM's memory (in GB) : vCPU ratio by 110% or 90% respectively. A c4-standard-8 has 30 GB memory while AWS and Azure equivalents have 32 GB, hence the slight discrepancies. Changing the GCP command line to use the same memtier_key_maximum and memtier_load_key_maximum as on AWS/Azure does not significantly impact results.

All performance results presented in this blog were created with the above methodology and are in alignment with internal testing of GCP. Results may vary due to changes to the underlying configuration, updates to PKB, and other conditions such as the placement of the VM and its resources, optimizations and other changes made by the cloud service providers, accessed cloud regions, co-tenants, and the types of other workloads exercised at the same time on the system.

We used on-demand list prices offered by respective cloud providers to create the price-performance graph. List price from the following regions were used:

- AWS: US East (N. Virginia)
- Azure: East US
- GCP: us-east4