Application performance best practices

Last updated by | Lisa Liu | Nov 6, 2020 at 10:35 AM PST

Application performance best practices

Monday, January 13, 2020
4:32 PM

a) *Connection pooling*: This significantly reduces connection latency by reusing existing connections and enables higher database throughput (transactions per second) on the server.

Typically, application side pooling opens a bunch of connections which sit idle at the database. Idle applications consume 10MB of memory per connection and can also create contention for establishing new connections and slow down queries. You can learn much more about this *here*.

Essentially if you see a high number of idle connections, it would be recommended to put a PostgreSQL specific connection pooler in place (server side connection pooler). We recommend leveraging pgbouncer for this. You can find steps to configure and install pgbouncer *here*.

b) *Accelerated Networking*: enables single root I/O virtualization (SR-IOV) to a VM, greatly improving its networking performance. This high-performance path bypasses the host from the datapath, reducing latency, jitter, and CPU utilization, for use with the most demanding network workloads on supported VM types.

c) Improve Performance of Read Intensive Workloads on Azure DB for PostgreSQL using Query Caching: *Pgpool-II Query Caching*, Redis cache , Memcached are the popular external query caching solutions read more about query caching *here*.

d) Application and Database should be in the same Data Center: make sure your application is in the same data center as your database server to avoid overhead network latency.

e) CPU exhaustion: Single-threaded applications can result in CPU exhaustion of one CPU while the other CPUs are under-utilized. Consider parallelizing your workload to take advantage of all the vCores available.

f) TCP_NODELAY: TCP_NODELAY is a client-side setting that should be considered on a client Virtual Machine (VM). Applications that benefit from the TCP_NODELAY option typically tend to do smaller infrequent writes and are particularly sensitive to latency. As an example, latency can be reduced from 15-40 ms to 2-3 ms with this setting.

Created with Microsoft OneNote 2016.

How good have you found this content?