PgBouncer and a Bit of Queueing Theory

Peter Eisentraut

2ndQuadrant
PostgreSQL

peter.eisentraut@2ndquadrant.com
@petereisentraut
max_connections = 10000
max_connections = 10000

- RAM
- I/O
- CPUs
- ...
How many then?

And what to do with the rest?
How many then?

https://wiki.postgresql.org/wiki/Number_Of_Database_Connections
(ca. 2012):

\((\text{core\_count} \times 2) + \text{effective\_spindle\_count}\)
max_connections vs. connection limit

ALTER ROLE ... CONNECTION LIMIT xxx;
Some benchmarking

server: AWS EC2 m5d.2xlarge (8 core, 32 GiB, 300 GB)
pgbench: same

database fits in RAM: pgbench -i -s 1024 = 17 GB

pgbench -T 60 -j32 -c32
latency average = 2.655 ms
tps = 12080.052280 (including connections establishing)
tps = 12083.253808 (excluding connections establishing)
Some more benchmarking

server: AWS EC2 m5d.2xlarge (8 core, 32 GiB, 300 GB)
pgbench: same

**database exceeds RAM:** pgbench -i -s 8192 = 136 GB

pgbench -T 60 -j32 -c48
latency average = 4.822 ms
tps = 9953.933322 (including connections establishing)
tps = 9956.487118 (excluding connections establishing)
What to do with the rest?
PgBouncer configuration

[databases]
myapp = host=elsewhere port=5432 dbname=myapp

[pgbouncer]
;listen_port = 6432
;pool_mode = session
default_pool_size = 32
max_client_conn = 10000
About pool modes

- `pool_mode = session`
- `pool_mode = transaction`
- `pool_mode = statement`
Deterministic queueing ex.

pool size = 1

transaction time = 10 ms = 100 tps

arrival rate = every 25 ms :-(
arrival rate = every 10 ms :-(
arrival rate = every 8 ms :-(

Deterministic queueing ex.

pool size = 10

transaction time = 10 ms = 100 tps

arrival rate = every 2.5 ms :-)
arrival rate = every 1.0 ms :-
arrival rate = every 0.8 ms :-(

Queueing nodes

\( \lambda \): arrival rate
\( \mu \): departure rate
Kendall’s notation

A/S/c

A = arrival process
S = service time distribution
c = number of servers
Kendall’s notation examples

D/D/1
D/D/k
M/D/1
M/M/1
M/M/k
M/G/1
...

Little’s law

\[ L = \lambda W \]

- **\( L \)**: average number of jobs in system (load)
- **\( \lambda \)**: average arrival rate
- **\( W \)**: average time spent in system
M/M/1 queue

arrival rate $\lambda$
service rate $\mu$
server utilization $\rho = \lambda/\mu$

must: $\rho < 1$

$$\pi_i = (1 - \rho) \rho^i$$

$$\pi_0 = (1 - \rho)$$
$$\pi_1 = (1 - \rho) \rho$$

avg. nr. jobs $L = \rho/(1 - \rho)$
**M/M/1 queue example**

arrival rate \( \lambda = 1/25\text{ms} = 40/\text{s} \)

service rate \( \mu = 1/10\text{ms} = 100/\text{s} \)

server utilization \( \rho = \frac{\lambda}{\mu} = \frac{40}{100} = 0.4 \)

\[ \pi_0 = (1 - \rho) = 0.6 \]

\[ \pi_1 = (1 - \rho)\rho = 0.24 \]

avg. nr. jobs \( L = \frac{\rho}{1 - \rho} = 0.67 \)
M/M/1 queue response time

\[ W = \frac{L}{\lambda} = \ldots = \frac{1}{(\mu - \lambda)} \]

Example:

\[ W = \frac{1}{(100 - 40)} = 0.0167 \text{ s} \]
**M/M/c queue**

arrival rate $\lambda$

service rate $\mu$

server utilization $\rho = \lambda/(c\mu)$

must: $\rho<1$
M/M/c queue analysis

probability of having to wait:

\[ P = \text{ErlangC}(\frac{\lambda}{\mu}, c) \]

avg. nr. jobs in system:

\[ L = \frac{\rho}{1-\rho} \text{ErlangC}\left(\frac{\lambda}{\mu}, c\right) + c\rho \]

response time:

\[ W = \frac{\text{ErlangC}(\frac{\lambda}{\mu}, c)}{c\mu - \lambda} + \frac{1}{\mu} \]
def ErlangC(A, N):
    L = (A**N / factorial(N)) * (N / (N - A))
    sum_ = 0
    for i in range(N):
        sum_ += (A**i) / factorial(i)
    return (L / (sum_ + L))
M/M/c queue examples

$\lambda = 1/25\text{ms} = 40/\text{s}$
$\mu = 1/10\text{ms} = 100/\text{s}$

$c = 1$
$P = 0.4 \quad L = 0.67 \quad W = 0.0167 \text{s}$

$c = 2$
$P = 0.067 \quad L = 0.41 \quad W = 0.0104 \text{s}$

$c = 3$
$P = 0.008 \quad L = 0.40 \quad W = 0.010 \text{s}$
A final example

10000 tps
pool_size = 48

so

c = 48
\mu = \frac{10000}{48} = 208

\begin{tabular}{c c c c}
\lambda & P & L & W  \\
1000 & 0 & 4.8 & 0.0048  \\
2000 & 0 & 9.6 & 0.0048  \\
4000 & 0 & 19.2 & 0.0048  \\
6000 & 0.0007 & 28.8 & 0.0048  \\
8000 & 0.09 & 38.8 & 0.0049  \\
9000 & 0.37 & 46.7 & 0.0051  \\
\end{tabular}
Summary

- arrival rate (measure, calculate)
- service rate (measure, benchmark)
- server count/pool size (benchmark)
- load (measure)
- response time (measure, calculate)
- waiting probability
- Little’s law
- M/M/c queue
- Erlang-C