Performance testing done right

Jiří Holuša
QE team lead at Hazelcast
HAZELCAST IMDG is an operational, in-memory, distributed computing platform that manages data using in-memory storage, and performs parallel execution for breakthrough application speed and scale.

HAZELCAST JET is the ultra fast, application embeddable, 3rd generation stream processing engine for low latency batch and stream processing.
Quick terminology
**Metrics**

- **Throughput** - number of operations per time unit (ops/sec)

- **Latency, response time** - time from the making the request to getting the response (us, ms, s, ...)

Test types

- **Performance** - results are **numbers**
  - Throughput tests - the more operations done, the better
  - Latency tests - lower latency at *fixed number of operations*

- **Stability** - result is a **yes/no** answer
  - Load/soak tests - system has to remain stable under given (extreme) conditions
First presentation problem

Find funny images

Problems

Description

Example

Solution
Problem #1: Description

Not distinguishing between latency and throughput tests

- These metrics are **often** related
  - The bigger the throughput, the lower the latency and vice versa
- Still, they are two different properties of the system
- Plus the relation does **not always** hold!
Problem #1: Example (1/2)
Not distinguishing between latency and throughput test

Better latency, same (or worse) throughput
Problem #1: Example (2/2)

Not distinguishing between latency and throughput test

Better throughput, same (or worse) latency

- Adding number of response threads to web server
Problem #1: Solution

Not distinguishing between latency and throughput test

- Always differ between latency and throughput tests
  - Latency test - fix the throughput
- Make sure to understand what we want to test for a given scenario
Problem #2: Description

**Inadequate load on the system**

- Stressing the system over the limit is a stability test, not suitable for comparing the numbers
- Not stressing the system enough might cause suboptimal performance causing actually testing of something else unintentionally
Problem #2: Example (1/3)

Inadequate load on the system

Too much load

8 clients on 1 machine
16 clients on 1 machine
24 clients on 1 machine
24 clients on 2 machines
Problem #2: Example (2/3)

Inadequate load on the system

Too much load - extensive context switching

16 clients on 1 machine
24 clients on 1 machines
24 clients on 2 machines
Problem #2: Example (3/3)
Inadequate load on the system
Not enough load

1 thread per 4 clients
4 threads per 1 client
Problem #2: Solution

Inadequate load on the system

- Get to know the behavior of your system
- Start with simple scenarios, then add complexity, observe the behavior and understand why is something happening

How to origami
Problem #3: Description

Throwing away latency results information

- Showing only:
  - Average
  - Minimum, maximum
  - Selected percentiles (p90, p95, p99, ...)
  - Full latency distributions
Problem #3: Example (1/4)

Throwing away latency results information

<table>
<thead>
<tr>
<th>Dataset (e.g. latency of operations in ms)</th>
<th>A</th>
<th>B</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>1 1 1 1 1 1 1 1 1 1</td>
<td>52 100</td>
</tr>
<tr>
<td>B</td>
<td>5 5 5 5 5 5 5 5 5</td>
<td>30 50</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th>A</th>
<th>B</th>
</tr>
</thead>
<tbody>
<tr>
<td>average</td>
<td>16</td>
<td>12</td>
</tr>
<tr>
<td>minimum</td>
<td>1</td>
<td>5</td>
</tr>
<tr>
<td>maximum</td>
<td>100</td>
<td>50</td>
</tr>
<tr>
<td>p20</td>
<td>1</td>
<td>5</td>
</tr>
<tr>
<td>p50</td>
<td>1</td>
<td>5</td>
</tr>
<tr>
<td>p80</td>
<td>1</td>
<td>5</td>
</tr>
<tr>
<td>p90</td>
<td>52</td>
<td>30</td>
</tr>
</tbody>
</table>
Problem #3: Example (2/4)
Throwing away latency results information
Problem #3: Example (3/4)

Throwing away latency results information

Interval latency 50 percentile
Problem #3: Example (4/4)

Throwing away latency results information
Problem #3: Solution

Throwing away latency results information

- In general, generate as many charts as possible
  - latency, throughput, system stats, memory stats, GC info, networking etc.
- Look at all of them, everything is related
- Great tool: [HdrHistogram](https://hdrhistogram.org/)
Problem #4: Description

Unexpected operations ratio

- Operation types: reads and writes
- Customer scenario "We have 80:20 read:write ratio"
- Accidentally ending up with a different ratio without noticing
Problem #4: Example

Unexpected operations ratio

Test setup - 80 read clients and 20 write clients

latency(read) = 1 ms | latency(write) = 2 ms
Problem #4: Example

Unexpected operations ratio

**Test setup** - 80 read clients and 20 write clients

latency(read) = 1 ms | latency(write) = 2 ms

Read client: 1 s / 1 ms = 1000 ops / sec
Write client: 1 s / 2 ms = 500 ops / sec
Problem #4: Example

**Unexpected operations ratio**

**Test setup** - 80 read clients and 20 write clients

latency(read) = 1 ms | latency(write) = 2 ms

Read client: \( \frac{1 \text{ s}}{1 \text{ ms}} = 1000 \text{ ops/sec} \)
Write client: \( \frac{1 \text{ s}}{2 \text{ ms}} = 500 \text{ ops/sec} \)

1000 reads/s * 60 s * 5 min * 80 clients = 24 000 000 reads
500 writes/s * 60 s * 5 min * 20 clients = 3 000 000 writes

**Resulting ratio** \( \sim = 88:11 \)
Problem #4: Solution

Unexpected operations ratio

- Find more info about the test scenario
- Executing different operations based on probability
Problem #5: Description

Performance regression

- Code change causing performance degradation
- Worst thing to happen
- Customer is unhappy
Problem #5: Solution

Performance regression

- Automation, automation, automation
- Storing and organizing the results
- Check it the results on a regular basis
Useful resources

- **How NOT to measure latency**, Gil Tene
  - [https://www.youtube.com/watch?v=lJ8ydluPFeU](https://www.youtube.com/watch?v=lJ8ydluPFeU)
- **Optimizing Java**, Benjamin J. Evans, James Gough
- **Systems Performance**, Brendan Gregg

And ...

- **My Twitter!** @jholusa
QUIZ TIME!
Quiz

- Test setup
  - 2 servers, 2 clients
  - Client is doing writes with values of sizes 1 KB, 10 KB and 100 KB

- Results

<table>
<thead>
<tr>
<th>Value size</th>
<th>Throughput (ops / sec)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 KB</td>
<td>101 558</td>
</tr>
<tr>
<td>10 KB</td>
<td>11 214</td>
</tr>
<tr>
<td>100 KB</td>
<td>1 105</td>
</tr>
</tbody>
</table>

- Anything fishy going on?