Resource Management for Cloud Functions with Memory Tracing, Profiling and Autotuning

Josef Spillner <josef.spillner@zhaw.ch>

Distributed Application Computing Paradigms + Service Prototyping
https://blog.zhaw.ch/splab/

Sixth International Workshop on Serverless Computing (WoSC6) 2020
https://www.serverlesscomputing.org/wosc6/#p3 // Dec 8, 2020
Utility Computing: Provisioning+Billing

conventional (VM-/container-based) cloud compute canvas
  cost=200%

FaaS-resolution (100ms) compute canvas
  cost=150%

Future (<1ms) compute canvas
  cost=110%

typical event-driven application load over time
with theoretic utility provisioning & microbilling
  cost=100%

0  100  200  300  400  500  600  t [ms]

significance: e.g.
sensor data processing <<100ms
dual solution:
better resolution (provider), re-use idle time (app)

significance: fluctuating memory needs over time
dual solution:
better resolution (provider), dynamic adjustment (app)

https://github.com/serviceprototypinglab/faas-timesharing

https://github.com/serviceprototypinglab/lambda-docker-measurements
Memory Provisioning in FaaS

Cost := duration * memory
Duration :=~ memory (e.g. in AWS)

Approach: change cost rectangle until “idle/waste loss“ minimised

3 Limitations:
• coarse-grained memory stepping
  • economical
• static memory allocation (but dynamic input data)
  • technical: e.g. underlying Docker container isolation; API pass-through
• no tracing/optimisation suggestion tools
Function Isolation Methods in FaaS

Process isolation:
• practically no isolation $\leftrightarrow$ no memory boundaries

Containers (\texttt{docker run}):  
• good enough isolation $\leftrightarrow$ static boundaries  
• alternatives/emerging: rkt, cri-o, kata, scone, singularity, saurus, shifter, udocker...

Hypervisors (\texttt{kvm}):  
• strong isolation $\leftrightarrow$ memory ballooning  
• slow start, increasingly being solved (unikernels, firecracker, lightvm, kvm patch)

WebAssembly, ...

ELASTICDOCKER (CLOUD‘17), HoloScale (UCC’20), VEMoC (UCC’20):  
• autonomous vertical scaling
Characteristics of FaaS

Capturing target properties:
- Plain old web search for docs (non-trivial web automation)
- + knowledge from literature (sometimes contradicting docs)
- + dark knowledge from experience
→ plain old spreadsheets

<table>
<thead>
<tr>
<th>Restriction</th>
<th>Default value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Maximum number of functions that can be created under a single service</td>
<td>50</td>
</tr>
<tr>
<td>Maximum number of triggers that can be created under a single function</td>
<td>10</td>
</tr>
</tbody>
</table>

n > 5: slowness

Alibaba Cloud Function Compute is billed on a Pay-As-You-Go basis. The fee consists of three parts and the Internet Traffic Fee is optional. Users are only charged for the Internet Traffic Fee when using the Internet to transfer function data.

Total Fee = Request Fee + Duration Fee + Public Network Traffic Fee (optional)

Function Compute cost calculator

**Free Tier**

The free tier is shared by the primary account and subaccount.

Requests: The first **one million calls per month** are free of charge.

Duration: The first **400,000 GB-seconds per month** are free of charge.

Note: Free calls and execution duration are automatically cleared at the beginning of each calendar month, rather than accrued to the next month.

**Request Fee**

The Request Fee indicates the total number of function calls.

- Price: $0.2 / 1 million calls

<table>
<thead>
<tr>
<th>Service</th>
<th>Price computation time($/GB-s)</th>
<th>Minimum time (ms)</th>
<th>Time granularity (ms)</th>
<th>Minimum memory (MB)</th>
<th>Memory granularity (MB)</th>
<th>Memory limit (MB)</th>
<th>Free monthly computation time (GB-s)</th>
<th>Price per request ($)</th>
<th>Request granularity</th>
<th>Free monthly requests</th>
<th>Price ($/Ghz-s)</th>
<th>Free Ghz-s</th>
</tr>
</thead>
<tbody>
<tr>
<td>AWS Lambda</td>
<td>1.66667E-05</td>
<td>100</td>
<td>100</td>
<td>128</td>
<td>64 (1MB according to SLD #145)</td>
<td>3008</td>
<td>400000</td>
<td>2E-07</td>
<td>1</td>
<td>1000000</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Google Cloud</td>
<td>2.5E-06</td>
<td>100</td>
<td>100</td>
<td>128</td>
<td>2^n</td>
<td>2048</td>
<td>400000</td>
<td>4E-07</td>
<td>1</td>
<td>2000000</td>
<td>1E-05</td>
<td>3E-06</td>
</tr>
<tr>
<td>Azure</td>
<td>1.6E-05</td>
<td>100</td>
<td>100</td>
<td>128</td>
<td>128</td>
<td>1536</td>
<td>400000</td>
<td>2E-07</td>
<td>1</td>
<td>1000000</td>
<td></td>
<td></td>
</tr>
<tr>
<td>IBM</td>
<td>1.7E-05</td>
<td>100</td>
<td>100</td>
<td>1</td>
<td>1 N/A</td>
<td>400000</td>
<td>-</td>
<td>1</td>
<td>- (1 million according to article)</td>
<td>1000000</td>
<td>-</td>
<td></td>
</tr>
<tr>
<td>Ali baba</td>
<td>1.668E-05</td>
<td>100</td>
<td>100</td>
<td>64</td>
<td>64 N/A</td>
<td>400000</td>
<td>2E-07</td>
<td>1</td>
<td>1000000</td>
<td>-</td>
<td>-</td>
<td></td>
</tr>
<tr>
<td>Oracle functions</td>
<td>1.417E-05</td>
<td>N/A</td>
<td>N/A</td>
<td>N/A</td>
<td>N/A</td>
<td>400000</td>
<td>2E-07</td>
<td>1</td>
<td>2000000</td>
<td>-</td>
<td>-</td>
<td></td>
</tr>
</tbody>
</table>
Characteristics of FaaS

Capturing target properties:
FaaS Characteristics & Constraints Knowledge Base

---
- name: IBM Cloud Functions
  synonyms: IBM OpenWhisk
  duration:
  - 1523164605: 300

- name: Microsoft Azure Functions
  synonyms: Azure Functions
  duration: # https://docs.microsoft.com/en-us/azure/azure-functions/functions-scale
  - 1502948087: 300

- name: AWS Lambda
  synonyms: Lambda, Amazon Lambda, λ
  duration:
  - 1524979005: 300 # https://aws.amazon.com/de/lambda/faqs/
  blocked:
  - 1524979005: ingress, egress:25, egress:udp, ptrace
  localdisk:
  - 1524979005: 500
  memory:
  - 1524979005: [128, 256, 512, 1024, 3008]
  parameters:
  - python:
  - 1524979005: [event, context]

adaptation (reconfiguration, migration, ...)

https://zenodo.org/record/1236763
https://www.rohub.org/rodetails/faascckb/overview
https://github.com/serviceprototypinglab/faascc
Application/Function Consumption

Memory tracing of function running as container

```bash
while [ $status != "exited" ]
  do
    sleep 0.001
    mem=$(cat /sys/fs/cgroup/memory/docker/$CONTAINERID/memory.usage_in_bytes)
    echo $mem >> aux
    echo "$(date +"%T.%3N"),$mem" >> $FILE
    status="$(docker inspect --format '{{.State.Status}}' $CONTAINER)"
  done

maxmem=$(sort -k 1 -h aux | tail -n 1)
```

Function running someplace else

cloudfunctions

Metrics from Cloud Functions. Launch stages of these metrics: BETA GA

The 'metric type' strings in this table must be prefixed with `cloudfunctions.googleapis.com/`. That prefix has been omitted from the entries in the table.

<table>
<thead>
<tr>
<th>Metric type</th>
<th>Launch stage</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Kind, Type, Unit</td>
<td>Display name</td>
<td></td>
</tr>
<tr>
<td>Monitored resources</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Function/user_memory_bytes

Memory usage

DELTA, DISTRIBUTION, By cloud_function

Distribution of maximum function's memory usage during execution, in bytes. Sampled every 60 seconds. After sampling, data is not visible for up to 240 seconds.

memory: Memory assigned to function in MB.

trigger_type: Function trigger type.
Derivation of Consumption Model

Composite Application
+Cloud Function()

Container

Functions

Tool

isolation
+API

Container memory use over time and microbilling period

memory(MB) vs. time(s)

AWS Lambda
GCF
IBM CF
coarse-grained

OpenWhisk
OpenFaaS
Fission
fine-grained
dynamic

FaaS Deployments

mem vs. t
Memory Consumption Examples

Input data profiles available?
- No: based on maximum
- Yes: based on binning or feature models...
Autotuning (Next-gen FaaS)

- initial allocation (∞)
- sampling rate (« 100ms)
- safety buffers (for upscaling)