Zurich University
of Applied Sciences

zh
aw

School of
Engineering

InIT Institute of Applied
Information Technology

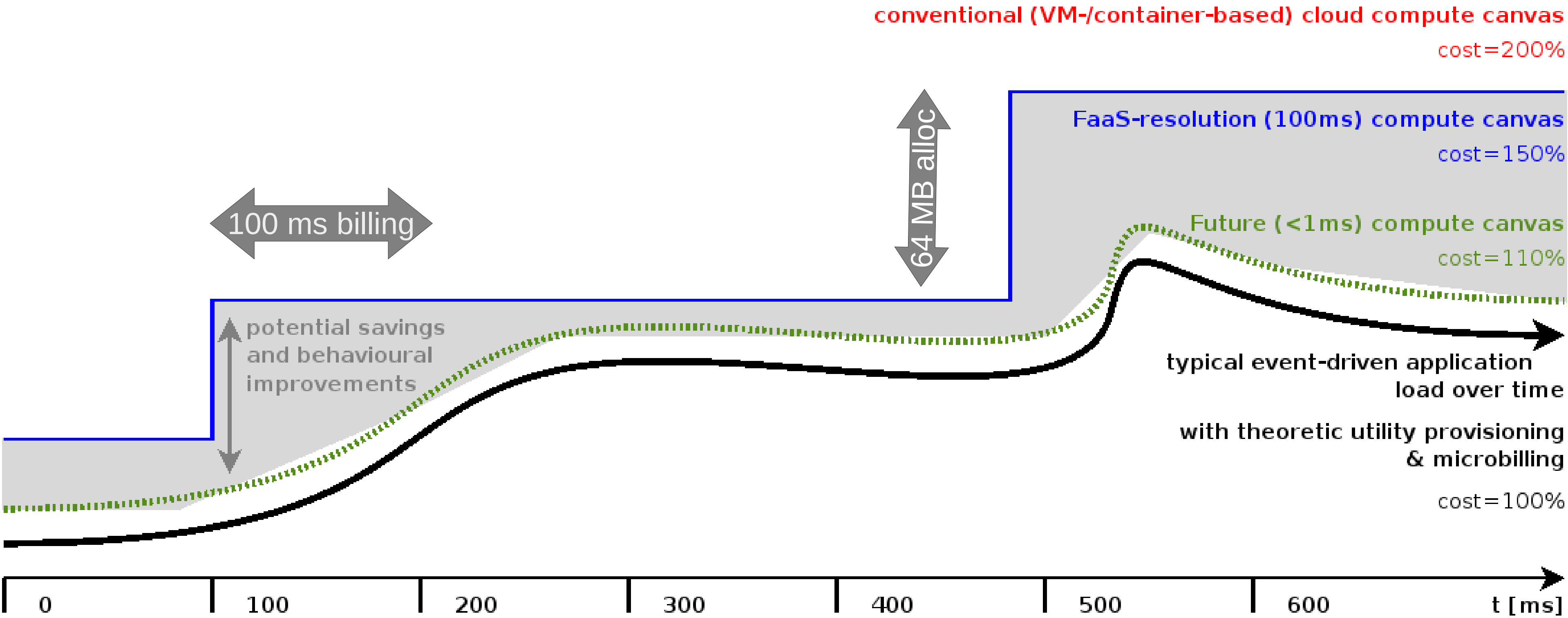
Resource Management for Cloud Functions with Memory Tracing, Profiling and Autotuning

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



time

significance: e.g.
sensor data processing <<100ms

dual solution:
better resolution (provider), re-use idle time (app)

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

mem

significance: fluctuating
memory needs over time

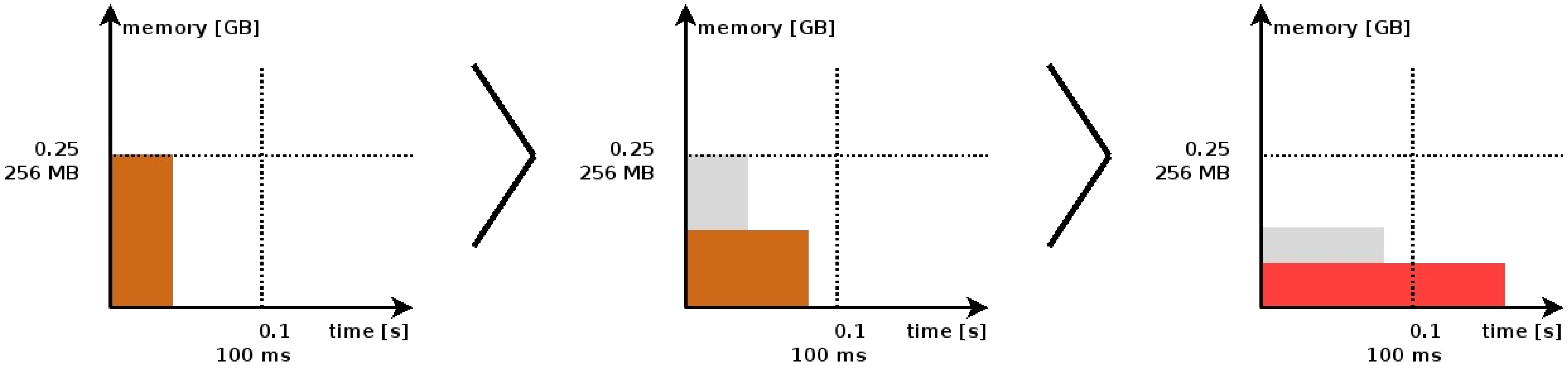
dual solution:
better resolution (provider), dynamic adjustment (app)

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 ←→ no memory boundaries

Containers (`docker run`):

- good enough isolation ←→ static boundaries
- alternatives/emerging: rkt, cri-o, kata, scone, singularity, saurus, shifter, udocker...

Hypervisors (`kvm`):

- strong isolation ←→ memory ballooning
- slow start, increasingly being solved (unikernels, firecracker, lightvm, kvm patch)

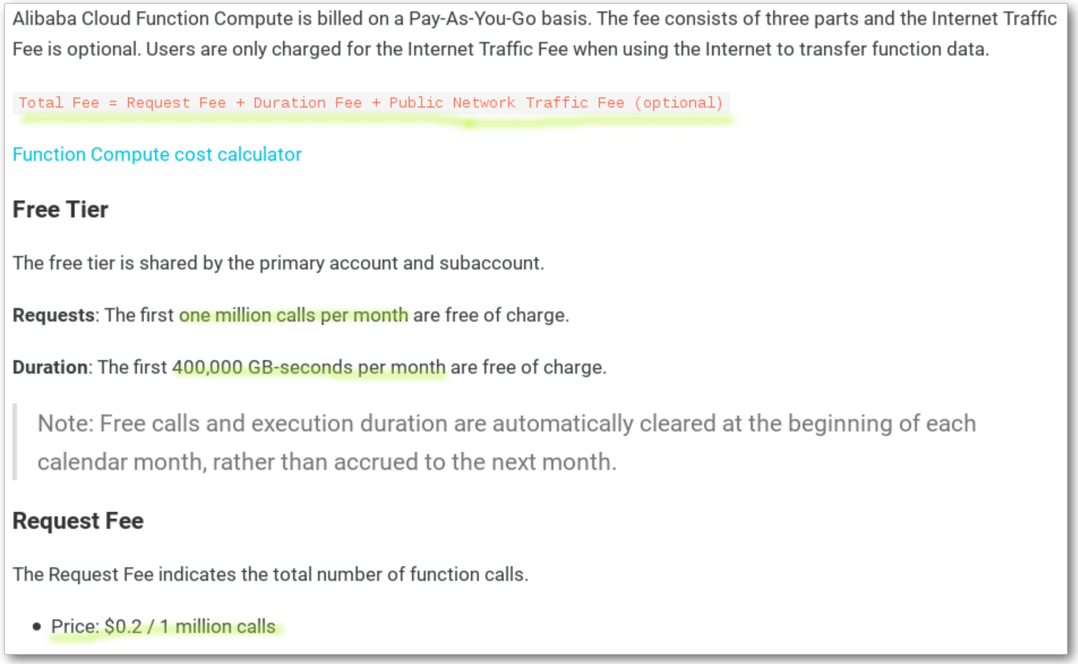
WebAssembly, ...

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



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.

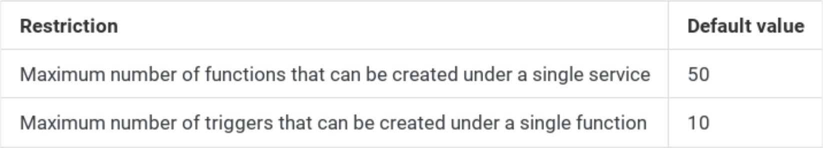
Duration: The first 400,000 GB-seconds per month are free of charge.

> 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

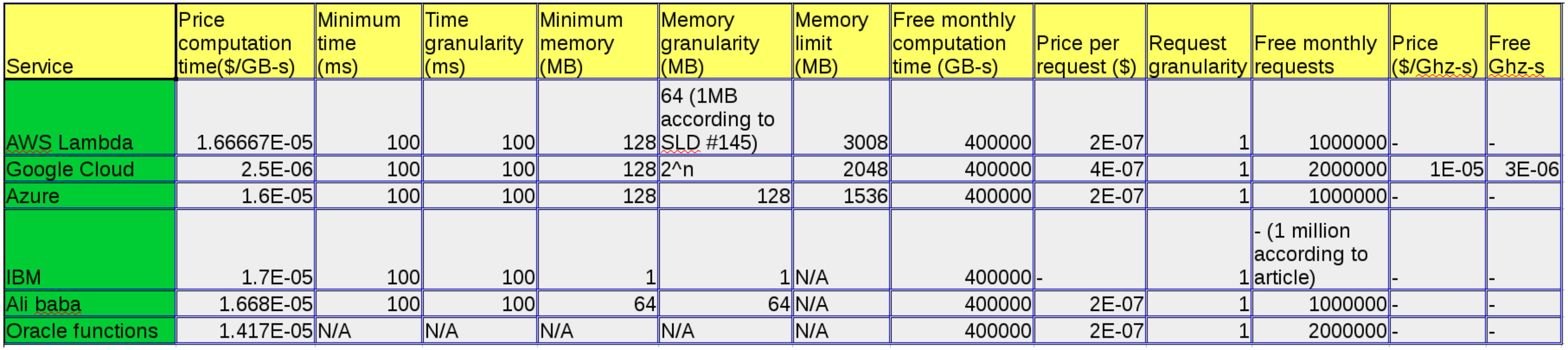
The Request Fee indicates the total number of function calls.

- Price: $0.2 / 1 million calls



Restriction	Default value
Maximum number of functions that can be created under a single service	50
Maximum number of triggers that can be created under a single function	10

n > 5: slowness



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

Characteristics of FaaS

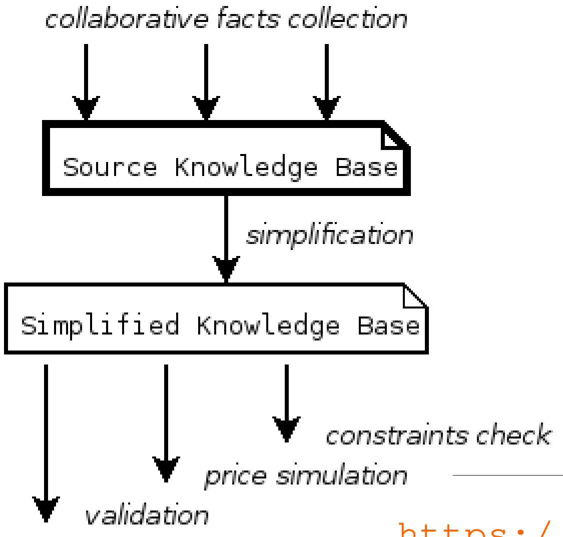
Capturing target properties:
FaaS Characteristics & Constraints Knowledge Base

https://zenodo.org/record/1236763
DOI 10.5281/zenodo.1236763

http://www.rohub.org/rodetails/faascckb/overview

```
---
 name: IBM Cloud Functions
  synonyms: IBM OpenWhisk
  duration:
  - 1523164605: 300
  - 1524979005: 600 # https://www.ibm.com/blogs/bluemix/2018/04/ibm-cloud-functions-doubling-time-limit-executing-actions/
- name: Microsoft Azure Functions
  synonyms: Azure Functions
  duration: # https://docs.microsoft.com/en-us/azure/azure-functions/functions-scale
  - 1502948087: 300
  - 1524979005: 600 # https://buildazure.com/2017/08/17/azure-functions-extend-execution-timeout-past-5-minutes/
```

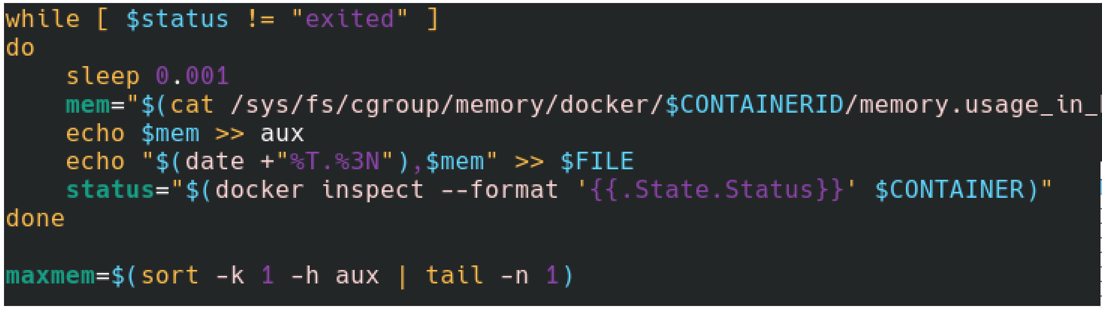
```
---
- name: AWS Lambda
  synonyms: Lambda, Amazon Lambda, λ
  duration:
  - 1524979005: 300 # https://aws.amazon.com/de/lambda/faqs/
  blocked:
  - 1524979005: ingress, egress:25, egress:udp, ptrace
  localdisk:
  - 1524979005: 500
  memory:
  - 1524979005: [128, 256, 512, 1024, 3008]
  parameters: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



adaptation (reconfiguration, migration, ...)

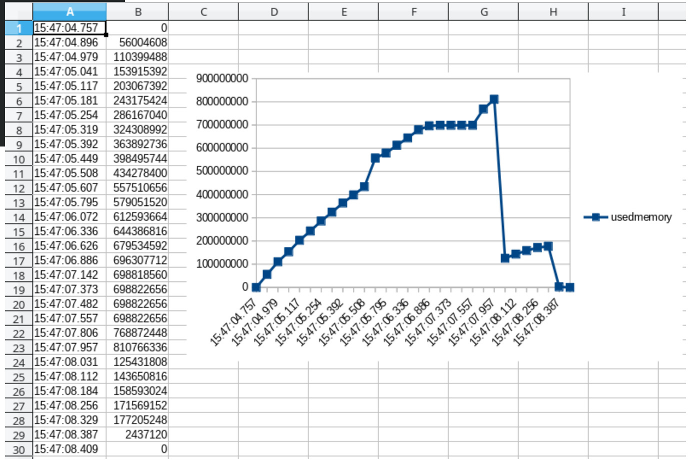
Application/Function Consumption

Memory tracing of function running as container



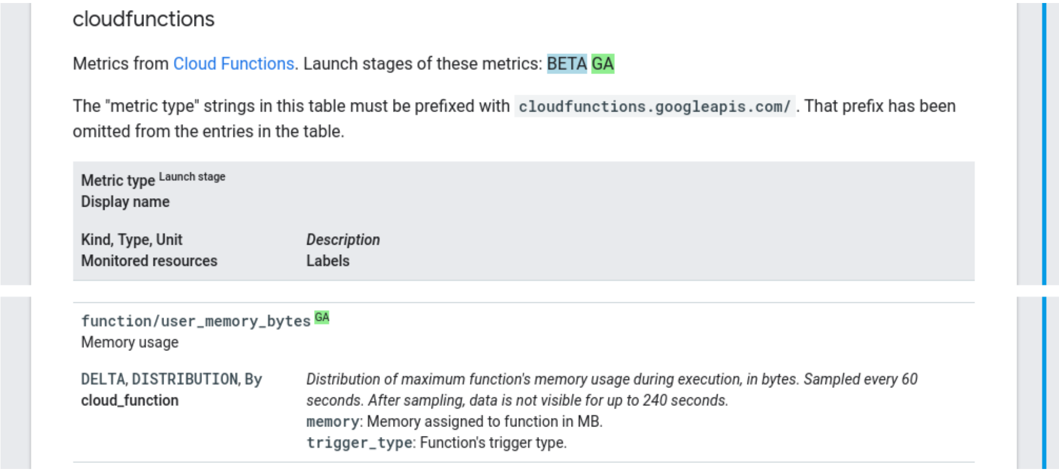
```
while [ $status != "exited" ]
do
    sleep 0.001
    mem="$(cat /sys/fs/cgroup/memory/docker/$CONTAINERID/memory.usage_in_bytes)"
    echo $mem >> aux
    echo "$(date +"%T.%3N"),$mem" >> $FILE
    status="$(docker inspect --format '{{.State.Status}}' $CONTAINER)"
done

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



	A	B	C	D	E	F	G	H	I
1	15:47:04.757	0							
2	15:47:04.896	56004608							
3	15:47:04.979	110399488							
4	15:47:05.041	153915392							
5	15:47:05.117	203067392							
6	15:47:05.181	243175424							
7	15:47:05.254	286167040							
8	15:47:05.319	324308992							
9	15:47:05.392	363892736							
10	15:47:05.449	398495744							
11	15:47:05.508	434278400							
12	15:47:05.607	557510656							
13	15:47:05.795	579051520							
14	15:47:06.072	612593664							
15	15:47:06.336	644386816							
16	15:47:06.626	679534592							
17	15:47:06.886	696307712							
18	15:47:07.142	698818560							
19	15:47:07.373	698822656							
20	15:47:07.482	698822656							
21	15:47:07.557	698822656							
22	15:47:07.806	768872448							
23	15:47:07.957	810766336							
24	15:47:08.031	125431808							
25	15:47:08.112	143650816							
26	15:47:08.184	158593024							
27	15:47:08.256	171569152							
28	15:47:08.329	177205248							
29	15:47:08.387	2437120							
30	15:47:08.409	0							

Function running someplace else



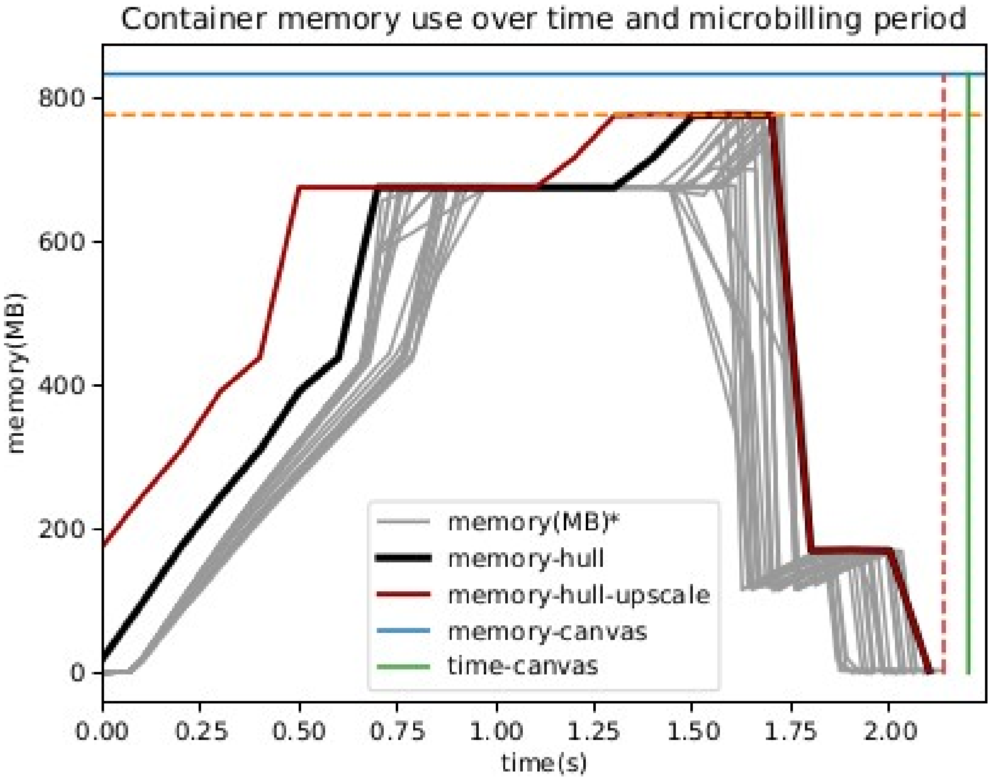
cloudfunctions

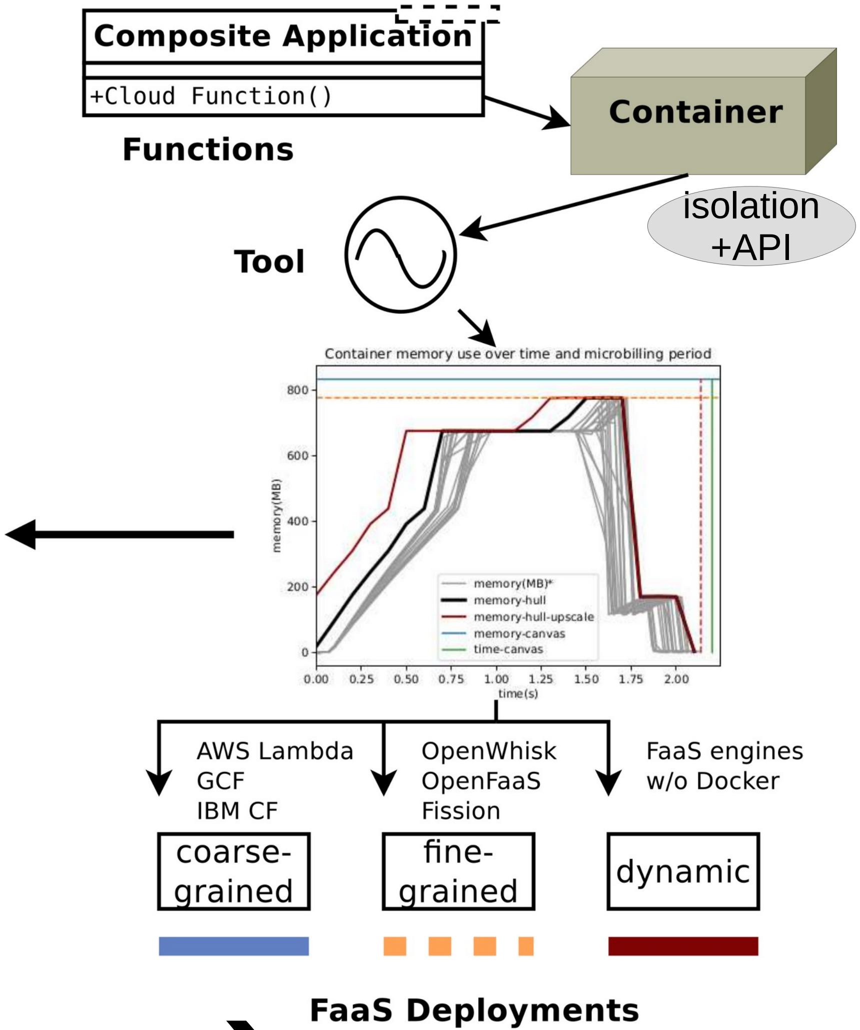
Metrics from Cloud Functions. Launch stages of these metrics: BETA GA

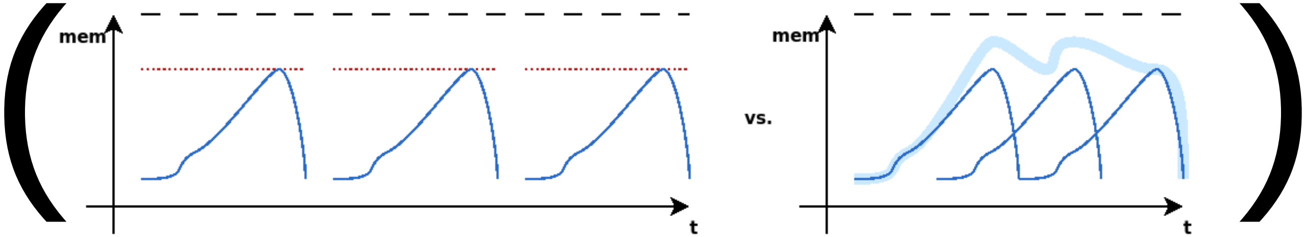
The "metric type" strings in this table must be prefixed with `cloudfunctions.googleapis.com/`. That prefix has been omitted from the entries in the table.

Metric type Launch stage Display name Kind, Type, Unit Monitored resources	*Description* Labels
`function/user_memory_bytes` GA 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's trigger type.

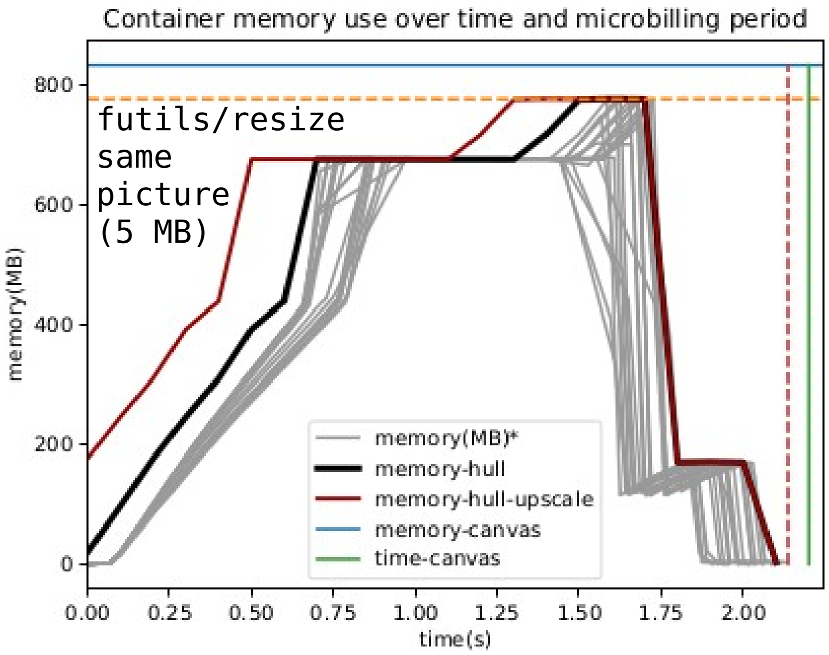
Derivation of Consumption Model

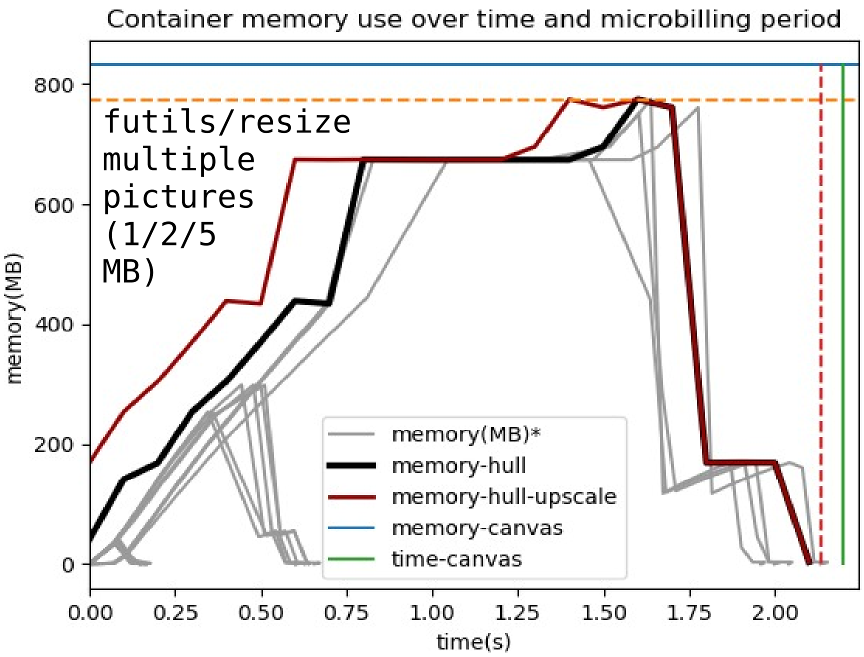




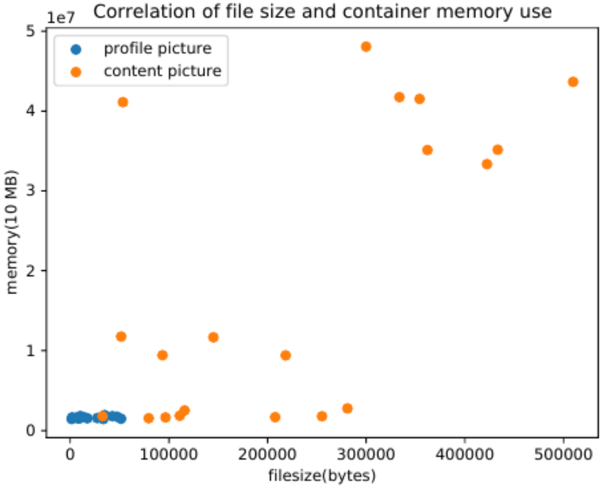


Memory Consumption Examples





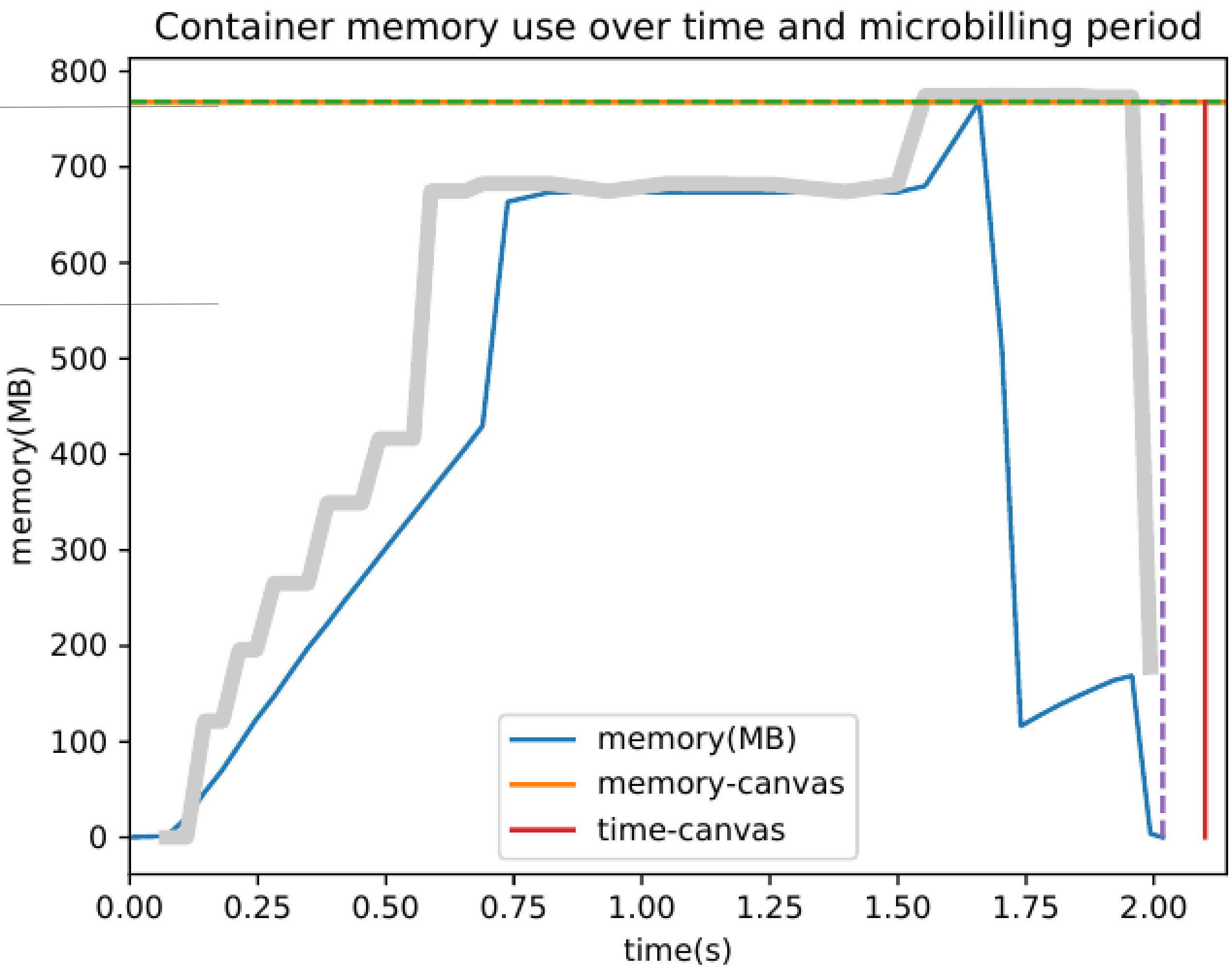
Input data profiles available?
→ No: based on maximum
→ Yes: based on binning or feature models...



Autotuning (Next-gen FaaS)

fine-grained static allocation saving (0-50%)

dynamic allocation saving (-90%)



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