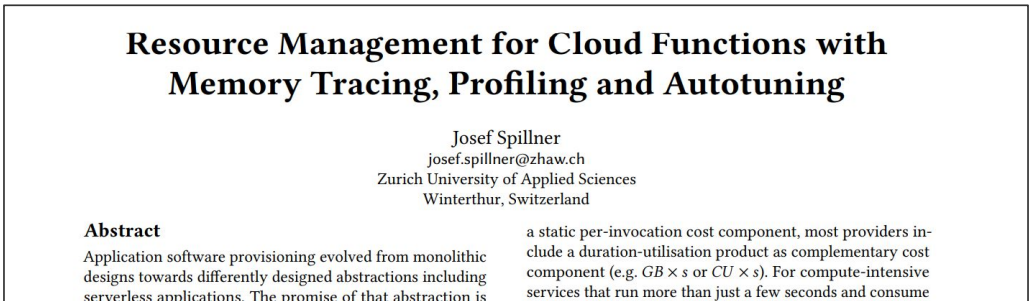
Resource Management for Cloud Functions with Memory Tracing, Profiling and Autotuning

Presented by Team 8

Duy Tran, Pragati Chidanand Patil, and Ranjana Bongale Ganesh



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

Josef Spillner
josef.spillner@zhaw.ch
Zurich University of Applied Sciences
Winterthur, Switzerland

Abstract

Application software provisioning evolved from monolithic designs towards differently designed abstractions including serverless applications. The promise of that abstraction is

a static per-invocation cost component, most providers include a duration-utilisation product as complementary cost component (e.g. $GB \times s$ or $CU \times s$). For compute-intensive services that run more than just a few seconds and consume

Agenda

- Problem Introduction
- Proposed Solution
- Summary of Technology Approach
- Experimental Evaluation
- Conclusion
- Critique (Strengths, Weaknesses, and Evaluation)
- GAPs
- Q&A

Problem Introduced in the Paper

FaaS - Function as a Service

- Advantages
  - Developers are free from Infrastructural concerns
    - Instance activation
    - Auto-scaling
- Problems introduced
  - Expose users to low-level decision making
    - Amount of memory to allocate
    - Coarse-grained profiles
    - No tracing tools available
    - Memory-allocation waste
    - Do not systematically trace actual versus declared memory consumption

How the problem is solved?

Author contributes tools to :

- Measure memory consumption of containerised functions execution over time to create trace profiles
  - FuncTracer
- Pricing forecast and adjust memory dynamically
  - AutoTuner
- Reuse Trace files to perform monetary analysis
  - Cost Calculator

Brief description on the tools are in the next slides.

Related Work (in working process…)

- **EMARS: Efficient Management and Allocation of Resources in Serverless**

  Aakanksha Saha;Sonika Jindal 2018 IEEE 11th International Conference on Cloud Computing (CLOUD) Year: 2018 | Conference Paper | Publisher: IEEE

- Two models proposed for efficient memory allocation:
  - **Workload based Modelling** - Number of requests for each function are logged and details are sent to the config generator thread which stores this value for every function and retrieved for use in any new function calls.
  - **Memory based Modelling** - Capture Memory requirements using docker stats.
  - **Config generator** - Workload based model and the Memory based model feed their data to the config generator thread which generates the optimal memory configurations per function and recommends these configurations for the creation and execution of containers.

Missing From the author's work:

- The authors have not performed any testing on the realistic workloads.
- The authors also mention that 'Some intelligence can be added to the config generator to figure out the optimal memory limits based upon the logged information'.

Summary of new technology, approach, or benchmark

- To reclaim the unused memory of invoking cloud functions, the author propose three tools:

functracer	autotuner	costcalculator
Measure the memory usage of a containerised function to create trace profiles.	Apply the trace profiles to record the runtime with memory limits.	Display the traces and calculate the potential economic gains.

Summary of new technology, approach, or benchmark

- To reclaim the unused memory of invoking cloud functions, the author propose three tools:

functracer | autotuner | costcalculator

Tracing method

- Convex hull function (Gift Wrapping Algorithm)
- It apply a tight fitting convex boundary around the points or shape.

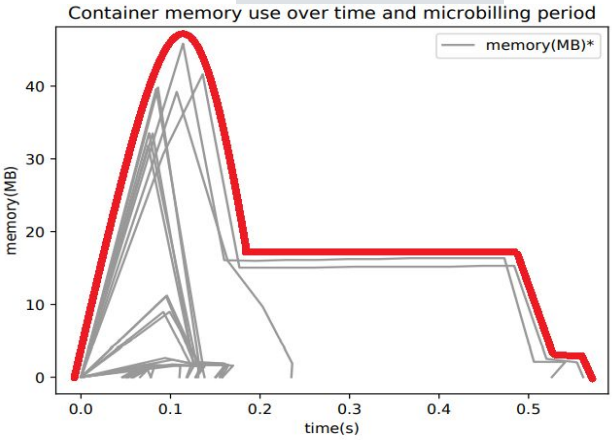


Figure 4. Overlay of 40 image processing memory traces with two desired input profiles

Summary of new technology, approach, or benchmark

- To reclaim the unused memory of invoking cloud functions, the author propose three tools:

functracer | autotuner | costcalculator

Interface	Metrics of interest
Docker	usage_in_bytes, limit_in_bytes
OpenFaas & Kubernetes	metrics-server
Google Cloud Functions & Google Run	function/user_memory_bytes container/memory/utilizations container/memory/allocation_time

Summary of new technology, approach, or benchmark

- To reclaim the unused memory of invoking cloud functions, the author propose three tools:

functractor | **autotuner** | costcalculator

Autotuning method

- Attached to functracer.
- Dynamically adjust the memory allocations for containers.
- Benefits FaaS providers to execute more contains if know the peak memory.

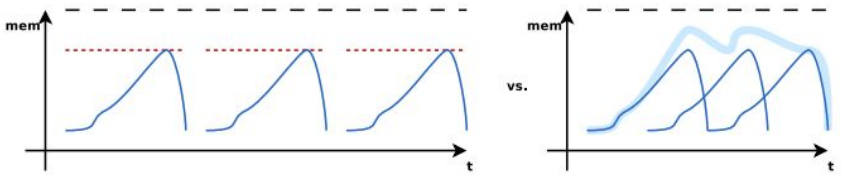


Figure 6. Dense scheduling of containers with memory autotuning – schematic

Summary of new technology, approach, or benchmark

- To reclaim the unused memory of invoking cloud functions, the author propose three tools:

functractor | autotuner | **costcalculator**

```
The function used 772 MB of memory and was allocated 832
     MB by AWS.
The total cost for AWS Lambda for 1 million requests per
     month would be 28.63 EUR.
The net cost would be 26.58 EUR, and the overhead cost
     2.05 EUR.
The price increases 7.71% due to wasted memory, and 0.00%
      due to wasted computation time.
0 milliseconds of computation time are being wasted.
60 MB of memory are being wasted.
```

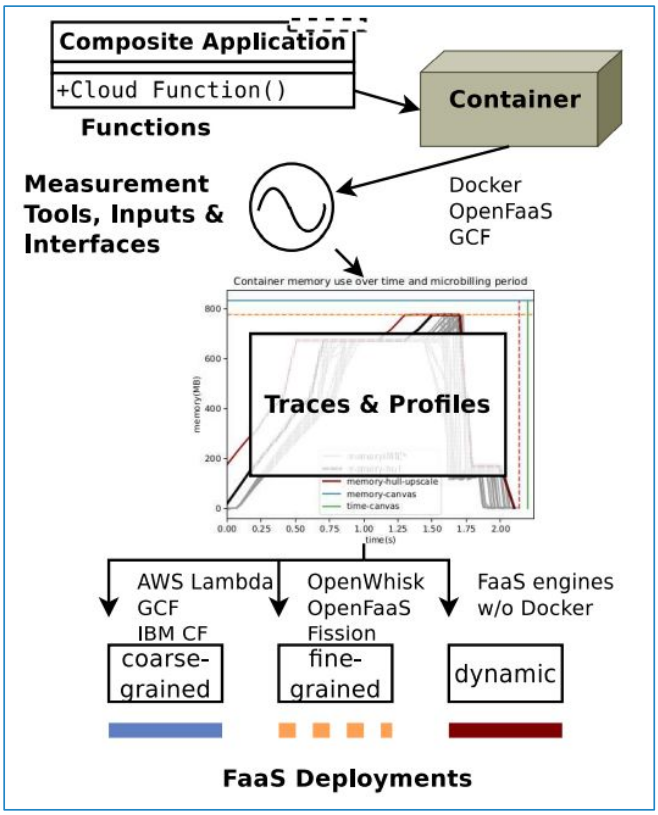
Cost calculation method

- Re-use the trace files to perform a monetary analysis.
- The log output will display the resource usage, wasted resource.

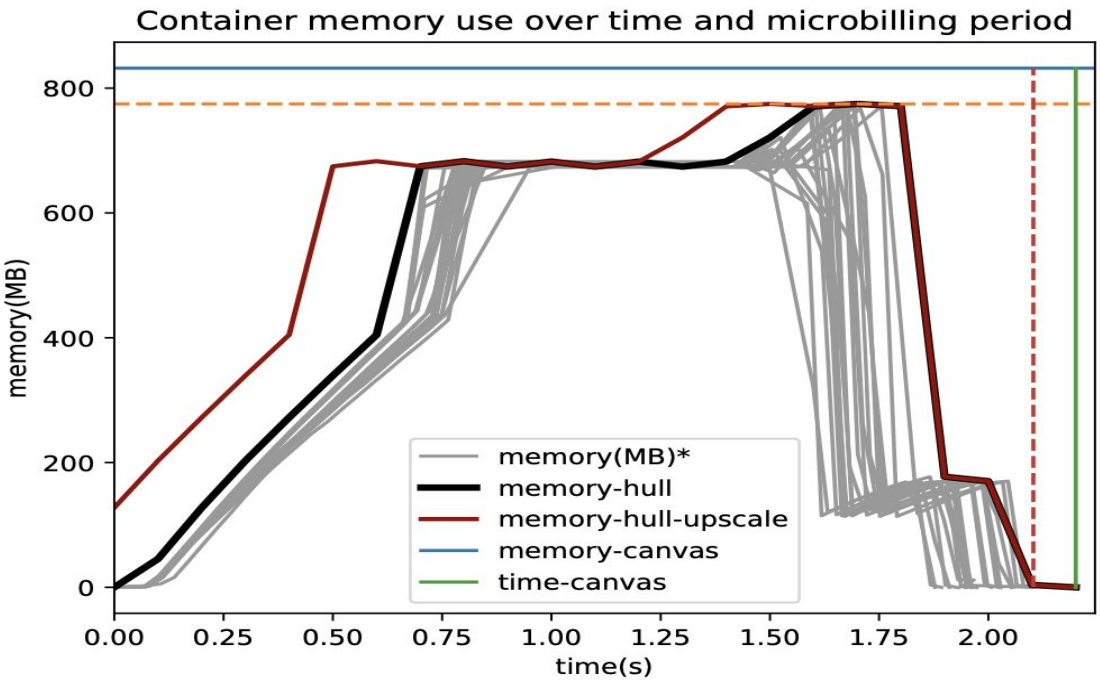
Experimental Evaluation & Approach

Three practical tools

- **functracer**
- autotuner
- costcalculator



Coarse-grained vs Fine-grained Allocation



Functracer - Tracing method

The ability to trace depends on the interfaces

- OpenFaaS running on docker(FaaSd)
- Kubernetes(faasnetes)
- Google cloud functions(GCFs)
- Pure Docker container tracing

Tracing for Docker

- Using recent versions of docker command line tools, tracing can be done more accurately.
- This technique works for short lived as well as for long duration function.
- Experiments shows fine grained dynamic allocation with auto tuning helps to reduce memory and time wastage across different functions.

Experimental Evaluation & Approach

Three practical tools

- Functracer
- **Autotuner**
- Cost Calculator

Autotuning

- Dynamically adjust memory allocation for containers.
- It can be standalone or part of functracer
- With autotuning we can save 0-50% for fine grained static allocation whereas we can save up to 90% for dynamic allocation.

Conclusion

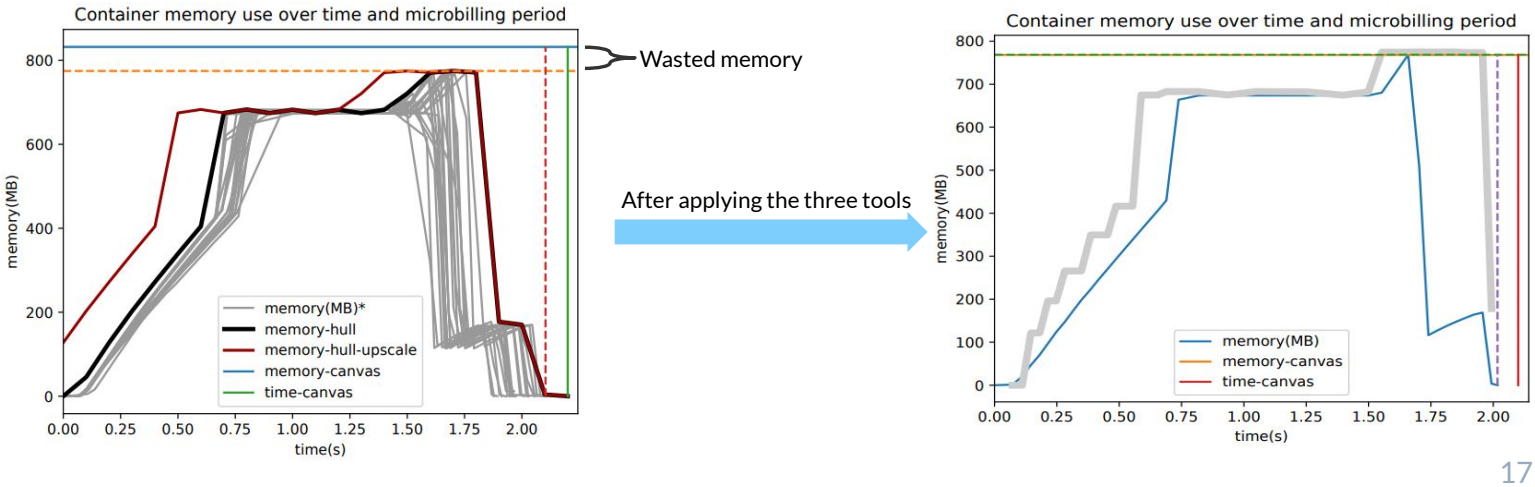
Reduce costly memory over allocation in cloud functions.

- Current FaaS – Trace memory consumption and configure the minimum possible allocation.
- Next gen FaaS - Dynamically adjust the memory allocation through vertical container resource scaling.

The coarse-grained metrics by cloud providers limits the approach, this can be overcome in future.

Strength of the Paper

- First, his tool **autotuner** promotes parallelism to execute more containers (more overlapped grey lines).
- He claims that his tools maintain the overhead of static allocation below 50%.
- Without his **autotuner**, the overhead could rise above 90%. This is expensive from a cost perspective.



Weakness of the Paper

- Needs lot of domain knowledge
- The approach differs depending on the interfaces, so it is complex.
- The approach is not domain agnostic as the approach needs adaptation based on type of interface.
- The coarse-grained metrics by cloud providers limits the approach.
- The approach does not work for most of the use cases due to the above limitation.

Critique: Evaluation

- Authors' paper requires more research and clearer definition of the following
  - No environment setup details provided
  - Unclear use of terms (ex: Advanced FaaS, hull?)
  - Not enough information to repeat/reproduce tests
  - Use of acronyms in Abstract and elsewhere in the paper
  - Assumptions not consistent with current knowledge
  - No credits provided to the co-developer of tools

Paper's Gaps

- The author, Josef Spillner, did make an achievement by reclaiming the wasted memory.
- Since this is a workshop paper, his paper lacks the explanation on how he implemented the three tools (**functracer**, **autotuner**, and **costalculator**).
- Many future works and limitations:
  - In his most important tools, the **autotuner** needs to be improved by reducing further the memory gap between static coarse-grained memory allocation and actually used memory.
  - **autotuner** needs to interface with more commercial FaaS management APIs. Currently, there are only three available interfaces.

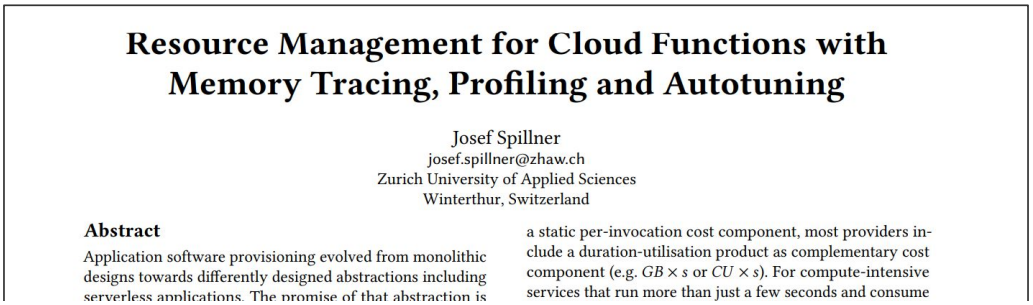
```
CONTAINER           CPU %               MEM USAGE / LIMIT     MEM %               NET I/O             BLOCK I/O
redis1              0.07%               796 KB / 64 MB        1.21%               788 B / 648 B       3.568 MB / 512 KB
redis2              0.07%               2.746 MB / 64 MB      4.29%               1.266 KB / 648 B    12.4 MB / 0 B
```

Thank you for listening!

Q&A session

Presented by

Duy Tran, Pragati Chidanand Patil, and Ranjana Bongale Ganesh



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

Josef Spillner
josef.spillner@zhaw.ch
Zurich University of Applied Sciences
Winterthur, Switzerland

Abstract

Application software provisioning evolved from monolithic designs towards differently designed abstractions including serverless applications. The promise of that abstraction is

a static per-invocation cost component, most providers include a duration-utilisation product as complementary cost component (e.g. $GB \times s$ or $CU \times s$). For compute-intensive services that run more than just a few seconds and consume