Performance considerations on execution of large scale workflow applications on cloud functions

M. Pawlik, K. Figiela, M. Malawski
m.pawlik@cyfronet.pl, {kfigiela,malawski}@agh.edu.pl

AGH University of Science and Technology
Kraków, Poland

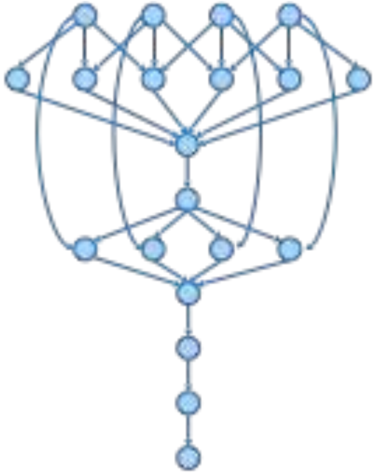
Presentation plan

1. Introduction
2. Objectives
3. Methodology
4. Results
5. Conclusions and future work

Scientific workflows and cloud

- Workflow - paradigm of implementing and preserving scientific process.
  - graph representation
  - allow for modeling complex procedures
  - provide a level abstraction over implementation details and infrastructure
  - enable parallelization
  - Workflow Management System is required to execute the workflow

- IaaS Cloud as an execution environment:
  - dynamic infrastructure provisioning
  - elastic billing models

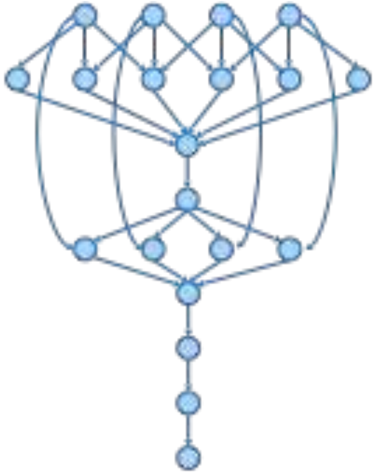


Montage workflow

Scientific workflows and FaaS

- Even greater elasticity.
- No need to manually provision the infrastructure.
- Very fine billing granularity.

- Some limitations:
  - single task runtime limit
  - limited set of function configurations (memory tied to cpu etc.)
  - reduced control (eg. cold starts)
  - introduction of overheads etc.



Montage workflow

Objectives

Measure parameters significant for assumed scenario: computation offloading to FaaS:

1. Performance specific for scientific applications
2. Infrastructure provisioning time
3. Overhead of the API (REST)
4. Other interesting characteristics

Provide basis for constructing performance models of scientific large scale workflows, which will allow for improvements in scheduling algorithms.

Studied infrastructures

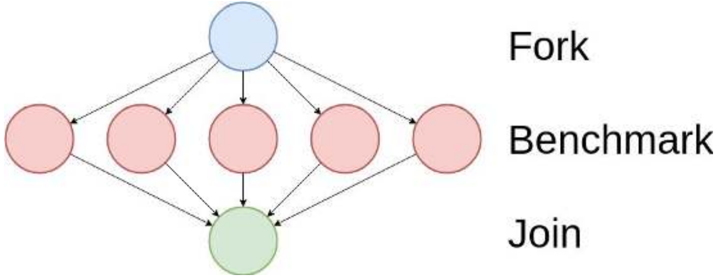
- Amazon: AWS Lambda (eu-west-1, 256MB, 512MB, 1024MB, 1536MB, 2048MB, 3008MB)
- Google Cloud Functions (us-central1, 256MB, 512MB, 1024MB, 2048MB)
- IBM Cloud Functions (UK, 256MB, 512MB)

- Most providers don’t supply exact information about infrastructure
- Even if they do it’s just a rough estimate

Benchmarking toolkit

- In-house developed workflow management system: HyperFlow
  - https://github.com/hyperflow-wms
  - written in Node.js, easy to use and extensible
  - supports FaaS

- Testing application: a “bag of tasks” workflow with 5120 tasks
  - exceeds limits imposed by most providers



- Testing load: Linpack
  - problem size of 3408x3408
  - concentrates on raw computing power (FLOPS)

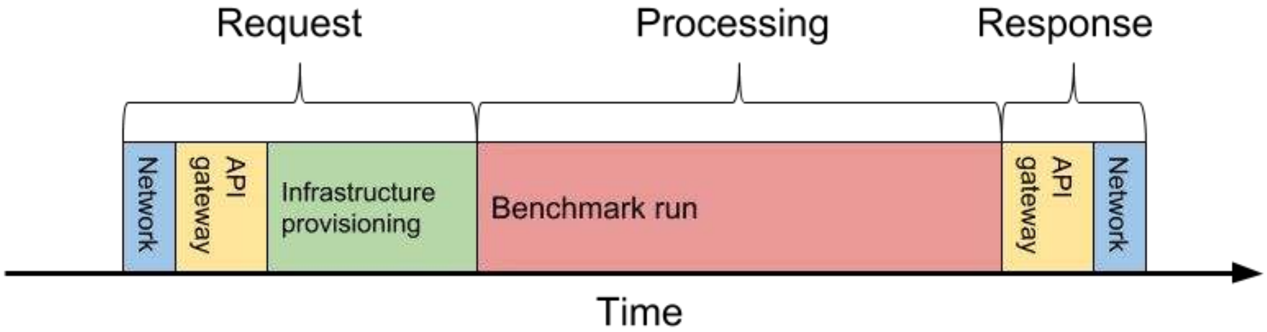
- Automation and reproducibility

serverless

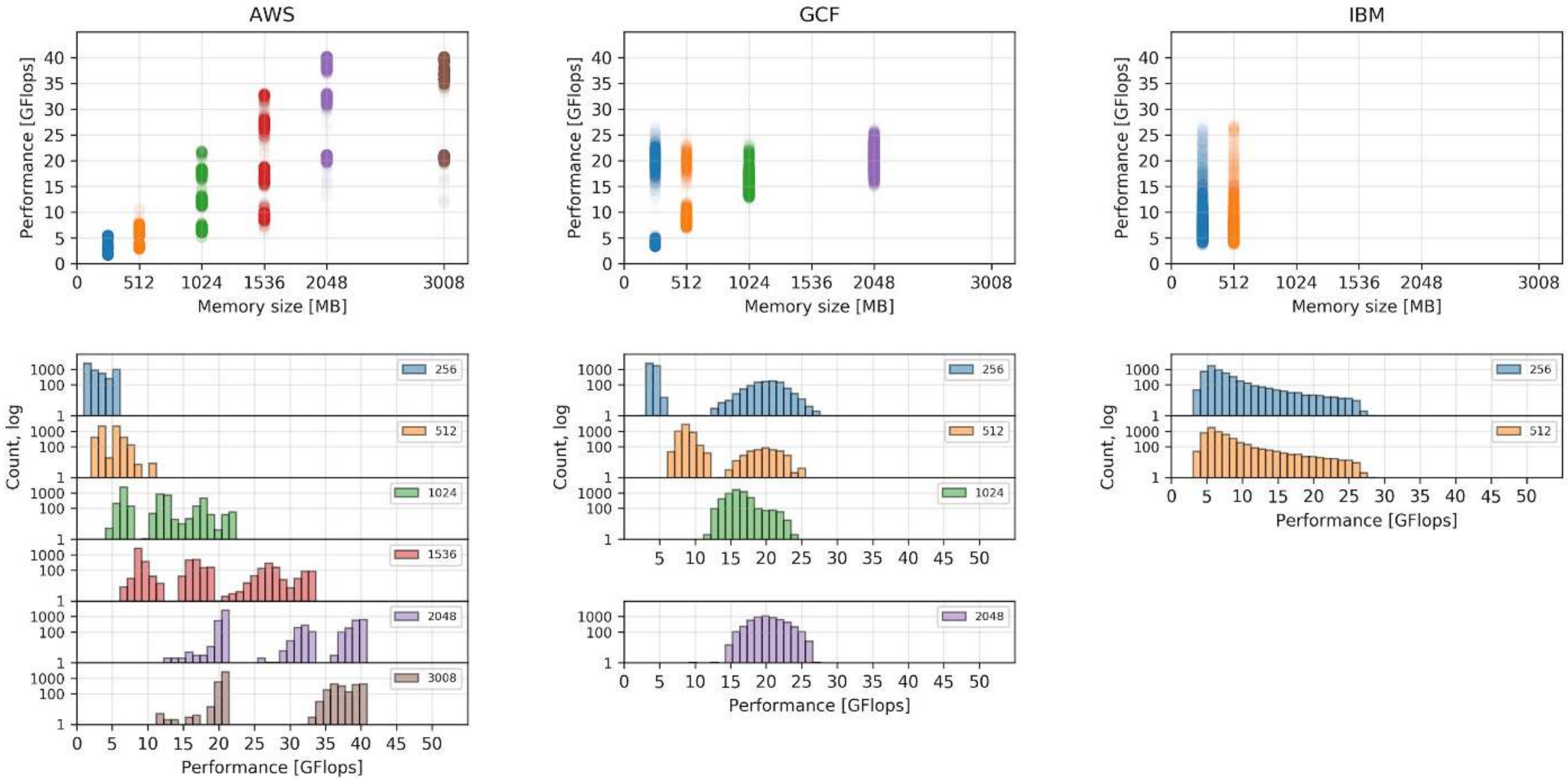
Experiment setup

- Local workflow orchestrator, workload was offloaded to FaaS.

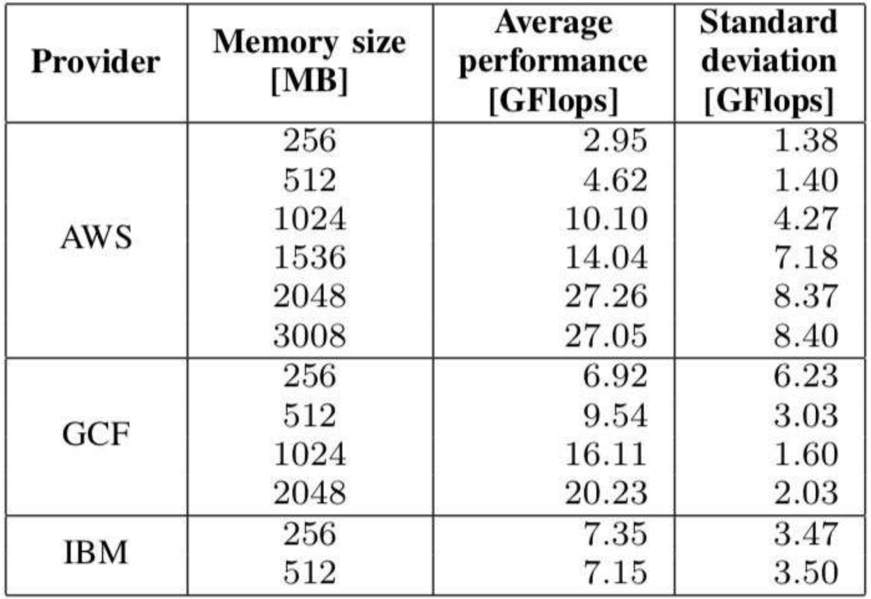
- Instrumentation was added to WMS and Cloud functions to gather detailed traces of execution.
- Task execution was divided into three stages:



Performance results

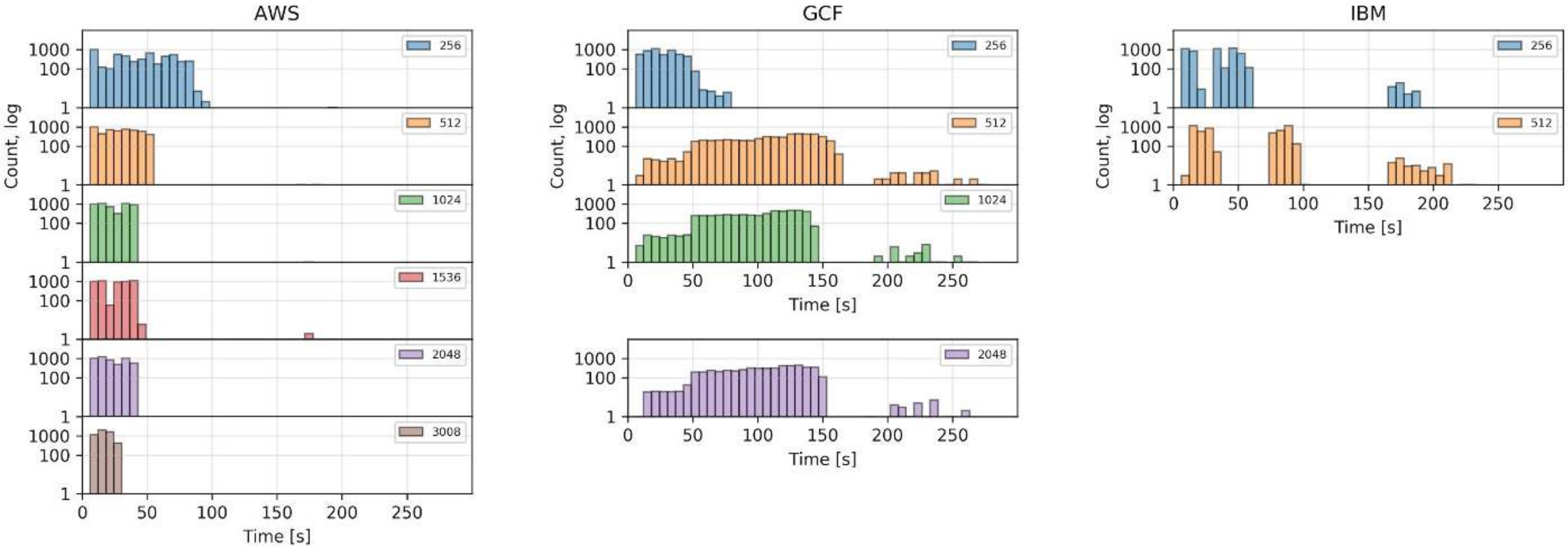


Performance results, cont.

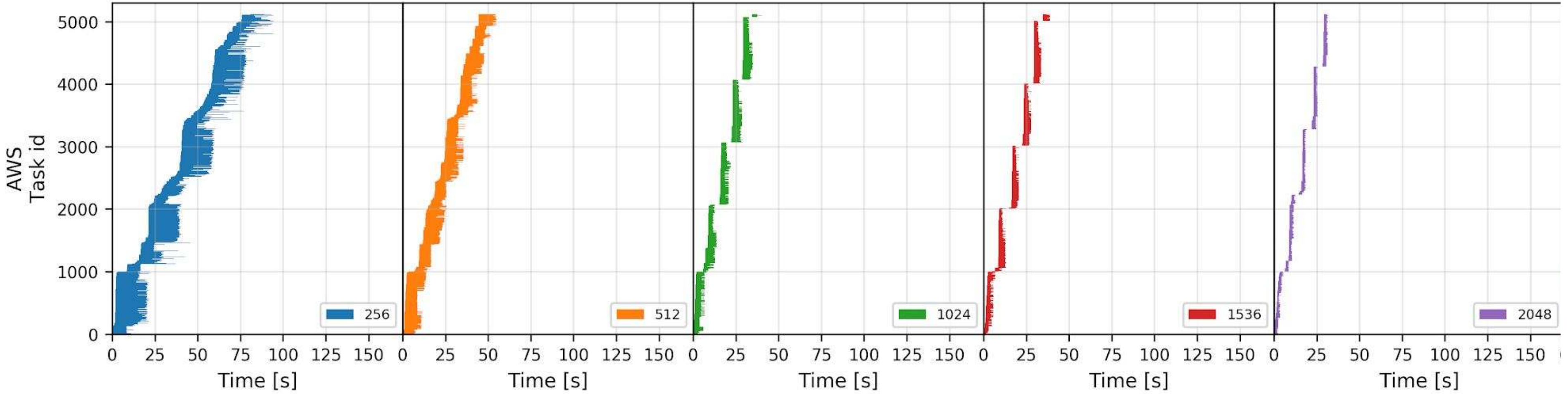


Provider	Memory size [MB]	Average performance [GFlops]	Standard deviation [GFlops]
AWS	256	2.95	1.38
	512	4.62	1.40
	1024	10.10	4.27
	1536	14.04	7.18
	2048	27.26	8.37
	3008	27.05	8.40
GCF	256	6.92	6.23
	512	9.54	3.03
	1024	16.11	1.60
	2048	20.23	2.03
IBM	256	7.35	3.47
	512	7.15	3.50

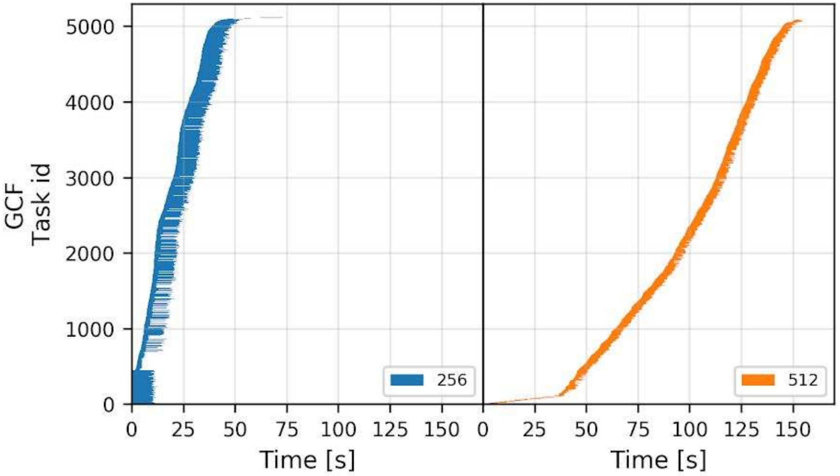
Task start delay

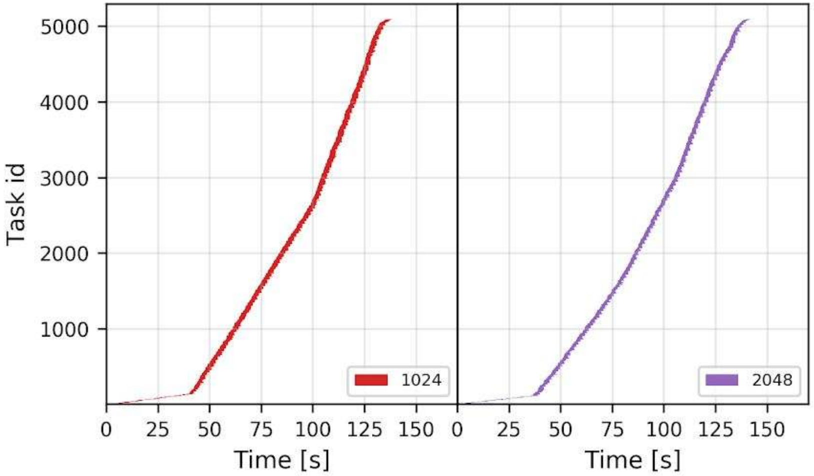


Infrastructure provisioning, Gantt charts, AWS

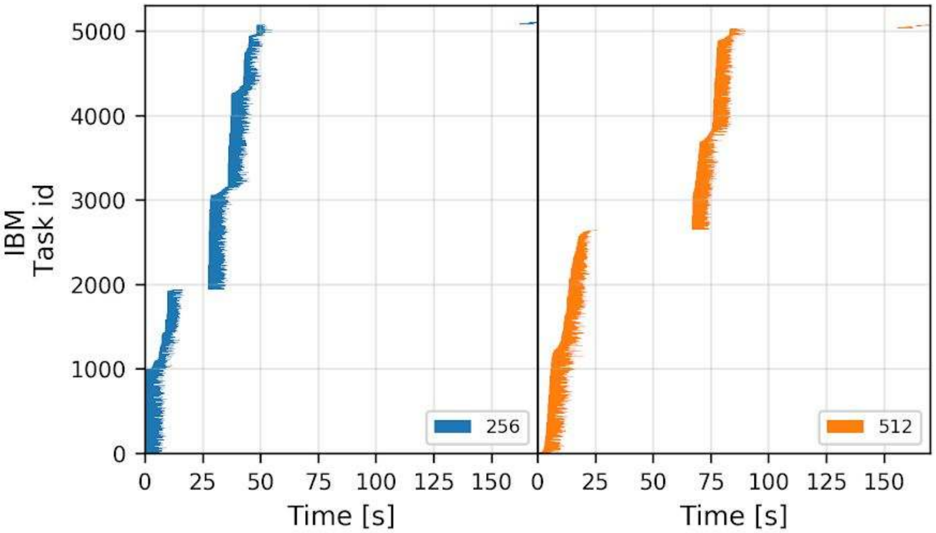


Infrastructure provisioning, Gantt charts, Google

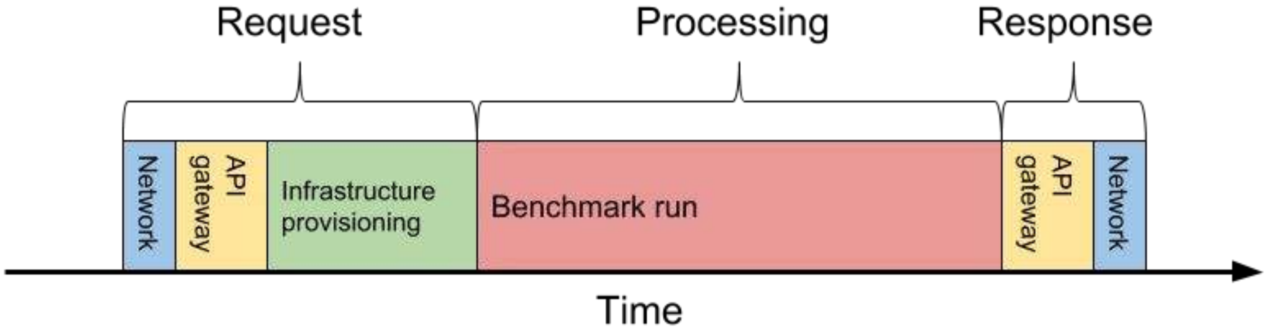




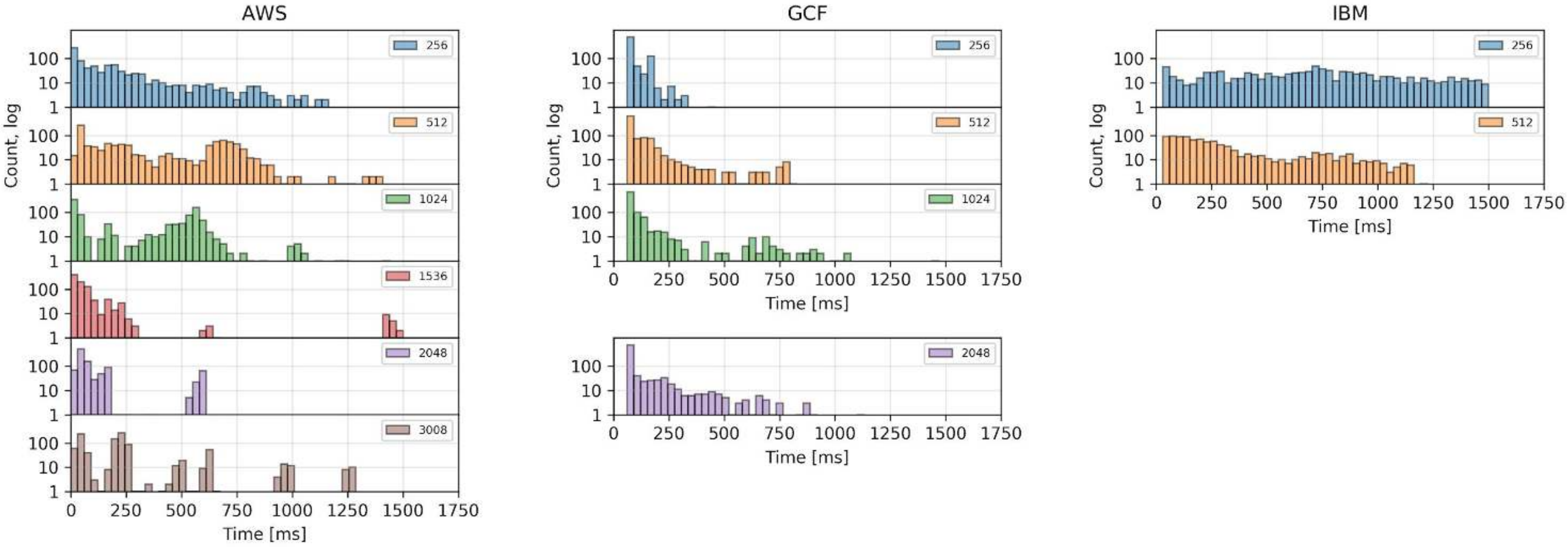
Infrastructure provisioning, Gantt charts, IBM



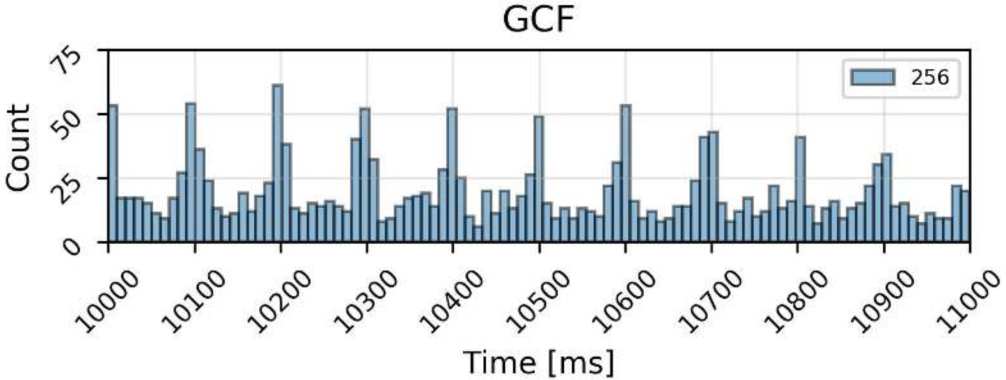
Response time



Response time



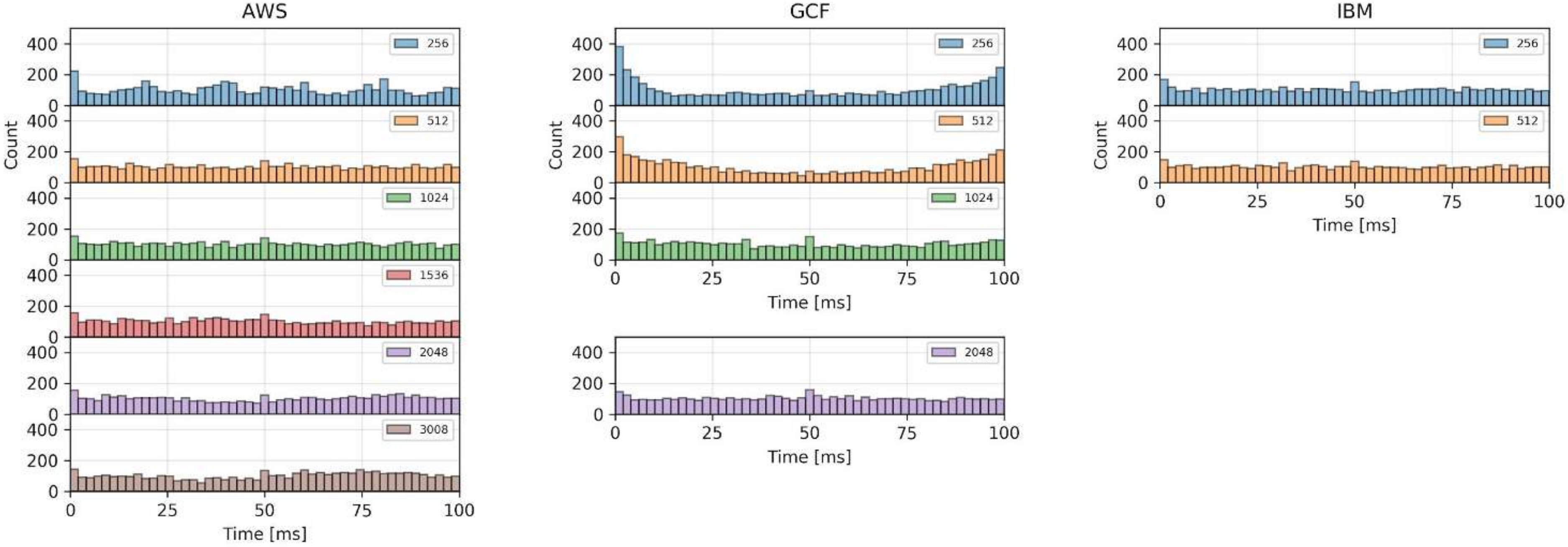
Patterns in measured run time



Concept of “remainder time”: $t_r = t \bmod 100ms$

- functions are billed for each 100ms
- runtime is rounded to nearest 100ms

Remainder time



Conclusions and Future work

- Presented work depicts significant characteristics of offered FaaS services.
- In terms of performance:
  - it varies significantly
  - largest function usually gives the least gains
  - faster function doesn’t always translate to shortest timespan
- HTTP API overhead is significant.

- Release full source code and automation scripts.
- Construct performance model and incorporate it into scheduling.
- Scheduling validation.

References

[1] E. Deelman, D. Gannon, M. Shields, and I. Taylor, “Workflows and e-science: An overview of workflow system features and capabilities,” Future Generation Computer Systems, vol. 25, no. 5, pp. 528–540, 2009.
[2] K. Figiela, A. Gajek, A. Zima, B. Obrok, and M. Malawski, “Performance evaluation of heterogeneous cloud functions,” Concurrency and Computation Practice Experience, vol. accepted, 2017.
[3] I. Baldini, P. Castro, K. Chang, P. Cheng, S. Fink, V. Ishakian, N. Mitchell, V. Muthusamy, R. Rabbah, A. Slominski et al., “Serverless computing: Current trends and open problems,” in Research Advances in Cloud Computing. Springer, 2017, pp. 1–20.
[4] S. Fouladi et al., “Encoding, Fast and Slow: Low-Latency Video Processing Using Thousands of Tiny Threads,” in 14th {USENIX} Symposium on Networked Systems Design and Implementation ({NSDI} 17). Boston, MA: {USENIX} Association, 2017, pp. 363–376.
[5] E. Jonas, Q. Pu, S. Venkataraman, I. Stoica, and B. Recht, “Occupy the cloud: Distributed computing for the 99%,” in Proceedings of the 2017 Symposium on Cloud Computing. ACM, 2017, pp. 445–451.
[6] M. Malawski, A. Gajek, A. Zima, B. Balis, and K. Figiela, “Serverless execution of scientific workflows: Experiments with HyperFlow, AWS Lambda and Google Cloud Functions,” Future Generation Computer Systems, vol. (In Print), nov 2017.
[7] J. Spillner, C. Mateos, and D. A. Monge, “Faaster, better, cheaper: The prospect of serverless scientific computing and hpc,” in Latin American High Performance Computing Conference. Springer, 2017, pp. 154– 168.
[8] H. Lee, K. Satyam, and G. C. Fox, “Evaluation of production serverless computing environments,” in Proceedings of the 3nd International Workshop on Serverless Computing. ACM, (In Print).
[9] M. Malawski, K. Figiela, A. Gajek, and A. Zima, “Benchmarking heterogeneous cloud functions,” in European Conference on Parallel Processing. Springer, 2017, pp. 415–426.
[10] B. Balis, “Hyperflow: A model of computation, programming approach and enactment engine for complex distributed workflows,” Future Generation Computer Systems, vol. 55, pp. 147–162, 2016.

Thank you for attention!

Any questions?

http://cloud-functions.icsr.agh.edu.pl/dashboard/db/providers

contact: m.pawlik@cyfronet.pl, malawski@agh.edu.pl