Serverless Containers – rising viable approach to Scientific Workflows

Krzysztof Burkat, Maciej Pawlik, Bartosz Balis, Maciej Malawski, Karan Vahi, Mats Rynge, Rafael Ferreira da Silva, Ewa Deelman

Team 9

Siddharth Sheth | Patrick Moy | Srivatsav Gopalakrishnan

Outline

- Introduction
- Discussion of key terminology
- Related work from authors
- Advantages of Serverless computing
- FaaS vs CaaS
- Cluster of Containers vs Container Platforms
- Experimental Framework
- Experiment Evaluation (Fargate vs Lambda)
- Conclusion
- Critique (Strength/ Weakness)
- Gaps & Future-work
- Questions

Introduction

- What?
  - Evaluating capabilities of elastic containers and their usefulness for scientific computing for scientific workflows
- How?
  - Hyperflow engine
  - 4 real-world scientific workflows
- Major Contributions

Discussion of key terminology

- Scientific workflow
- Hyperflow
- AWS Fargate
- Google Cloud Run

Background: related works

Publication	Takeaway
Serverless execution of scientific workflows: Experiments with HyperFlow, AWS Lambda and Google Cloud Functions (2017)	●FaaS efficient, possibly more cost-effective than traditional IaaS ●Not all workloads are suitable - granularity
Serverless execution of scientific workflows: Experiments with HyperFlow, AWS Lambda and Google Cloud Functions (2017)	●AWS Lambda highly ideal for scientific workflow applications ●Hybrid execution DEWE superior to traditional cluster execution
Challenges for Scheduling Scientific Workflows on Cloud Functions (2018)	●Adapted existing Serverless Deadline-Budget Workflow Scheduling algorithm for AWS Lambda
Real-time resource scaling platform for Big Data workloads on serverless environments (2019)	●Auto-scaling container clusters used to exceed FaaS limitations and have flexibility of CaaS

So what’s next?

Advantages of serverless computing

- Resources managed by Cloud Provider
- Elasticity and Scalability
- Cost

FaaS vs CaaS

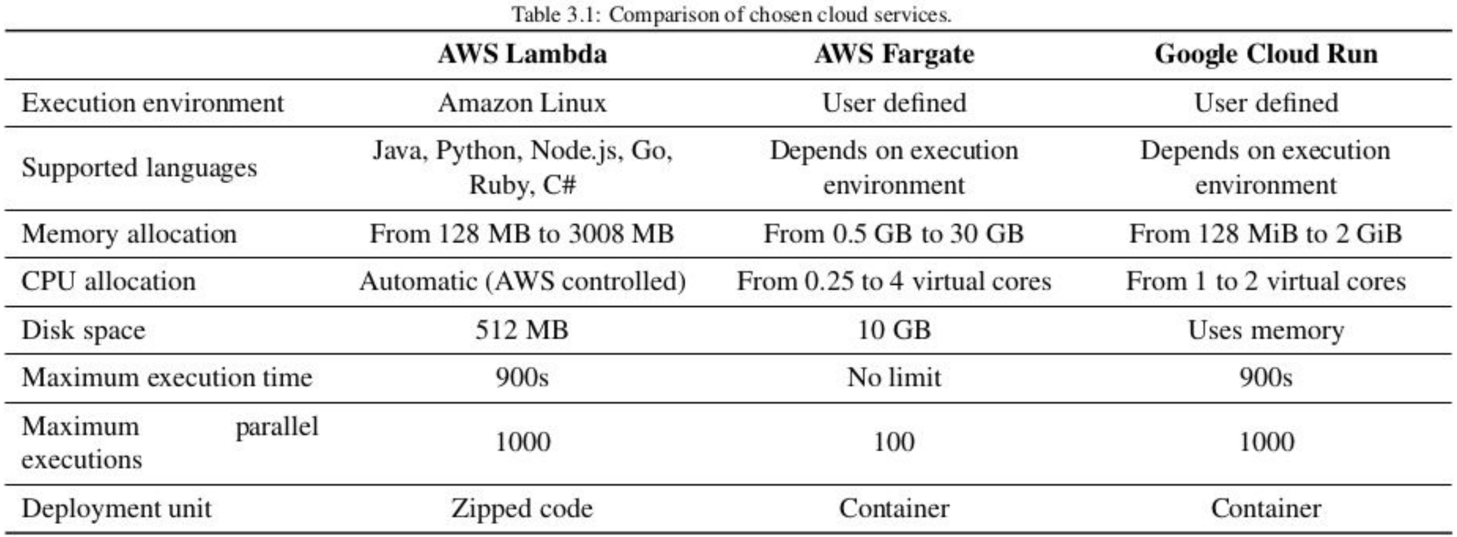


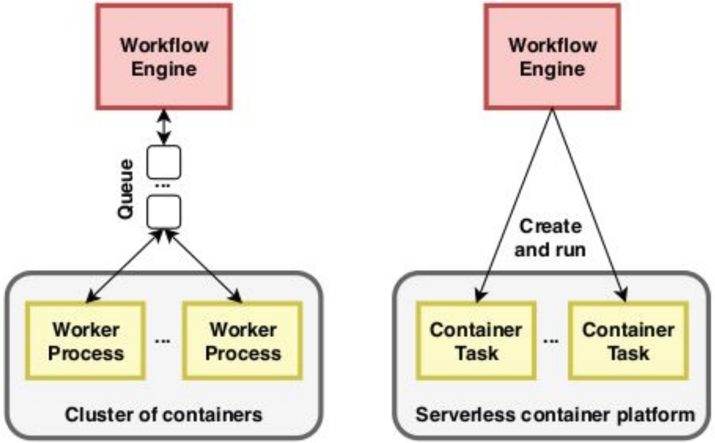
Table 3.1: Comparison of chosen cloud services.

	AWS Lambda	**AWS Fargate**	**Google Cloud Run**
Execution environment	Amazon Linux	User defined	User defined
Supported languages	Java, Python, Node.js, Go, Ruby, C#	Depends on execution environment	Depends on execution environment
Memory allocation	From 128 MB to 3008 MB	From 0.5 GB to 30 GB	From 128 MiB to 2 GiB
CPU allocation	Automatic (AWS controlled)	From 0.25 to 4 virtual cores	From 1 to 2 virtual cores
Disk space	512 MB	10 GB	Uses memory
Maximum execution time	900s	No limit	900s
Maximum parallel executions	1000	100	1000
Deployment unit	Zipped code	Container	Container

Cluster of containers vs Serverless container platforms

- The way tasks are mapped to containers.

- Workflow management.



Experimental Framework

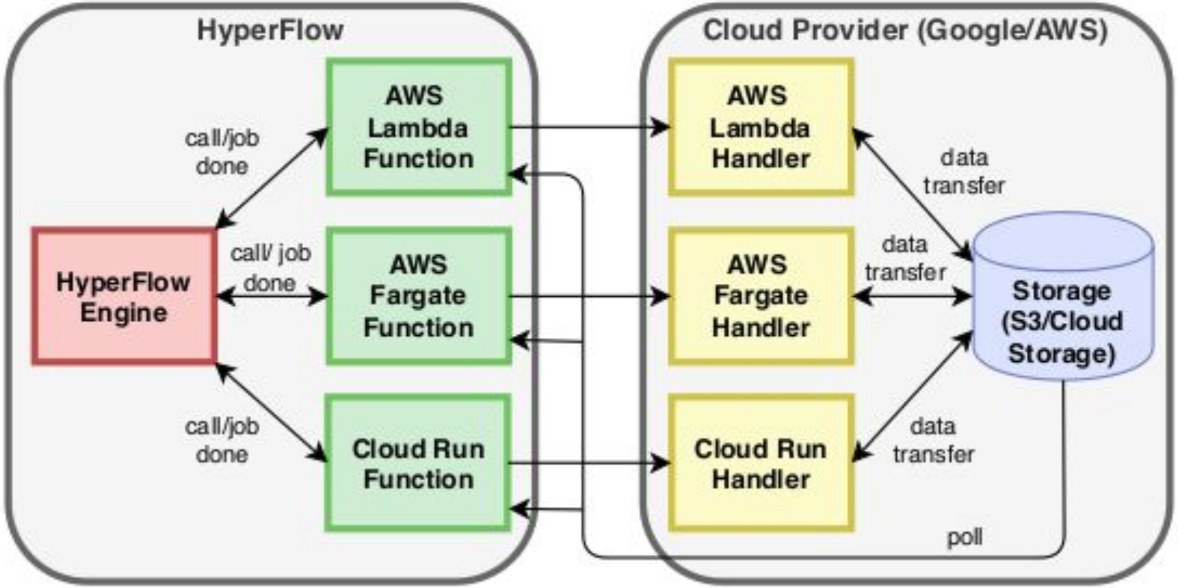


Figure 4.2: Proposed solution framework.

Experiment Evaluation

Services compared

- Amazon Fargate
- Google Cloud Run

Cold start & **C**ache for containers

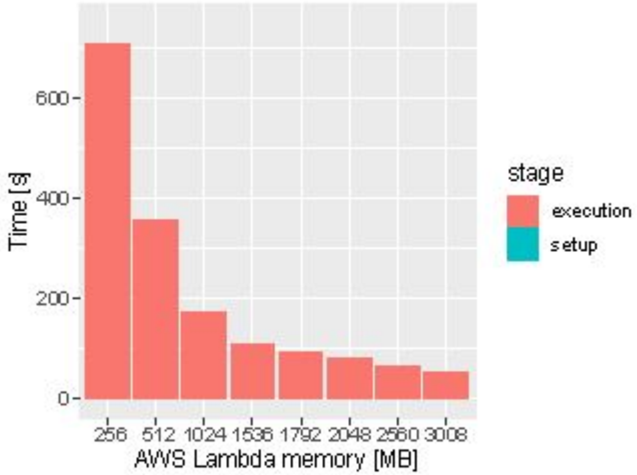
Objectives

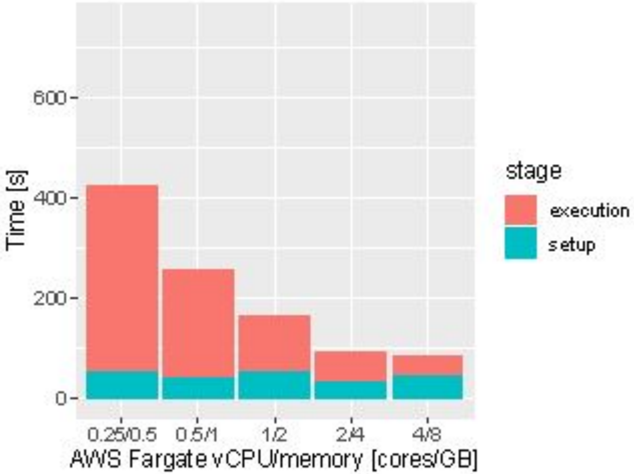
- Fargate vs Lambda
- Cloud Run vs Fargate limits and Burst rate
- Hybrid approach

4 Scientific workflows

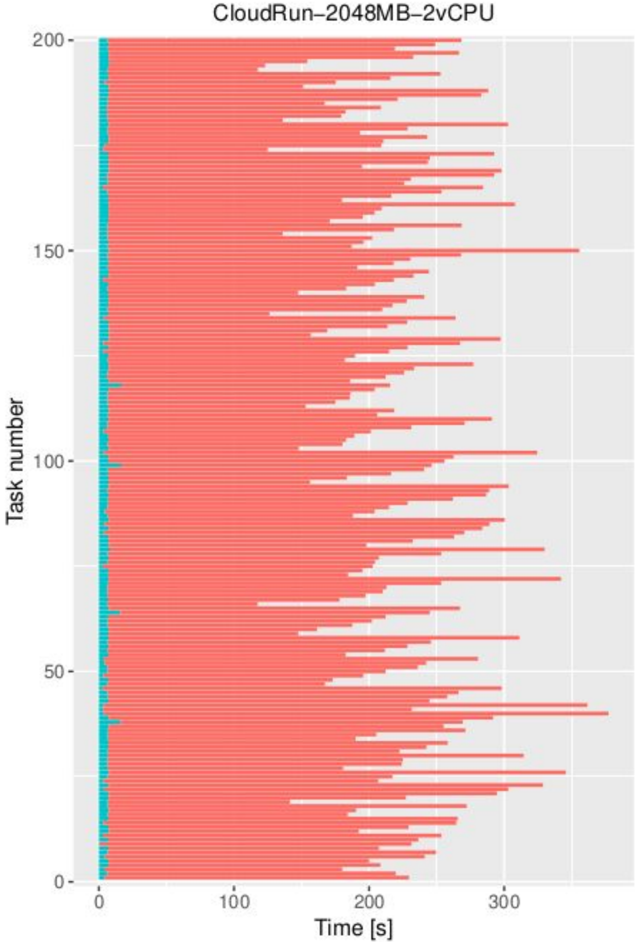
- Ellipsoids
- Vina
- KINC
- Soy-KB

Comparing the performance of Fargate and Lambda



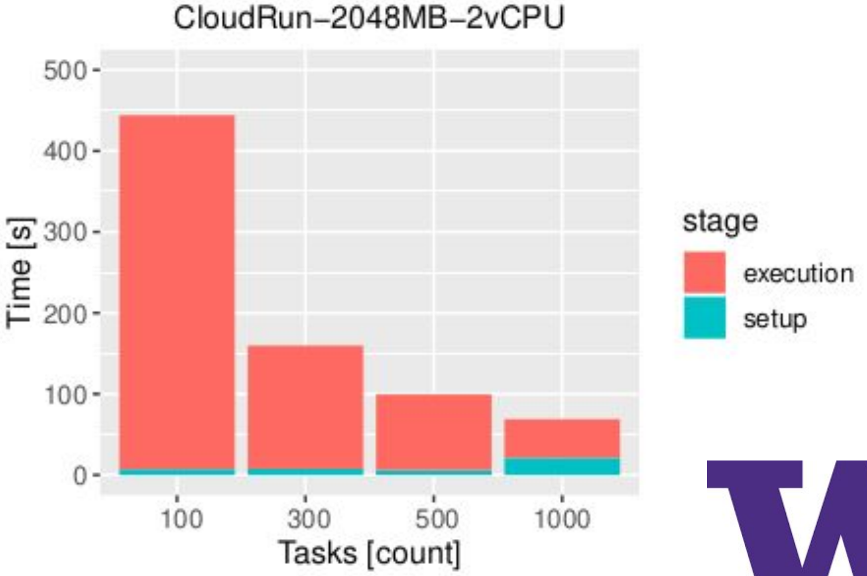


Comparing Cloud Run and Fargate limits-1

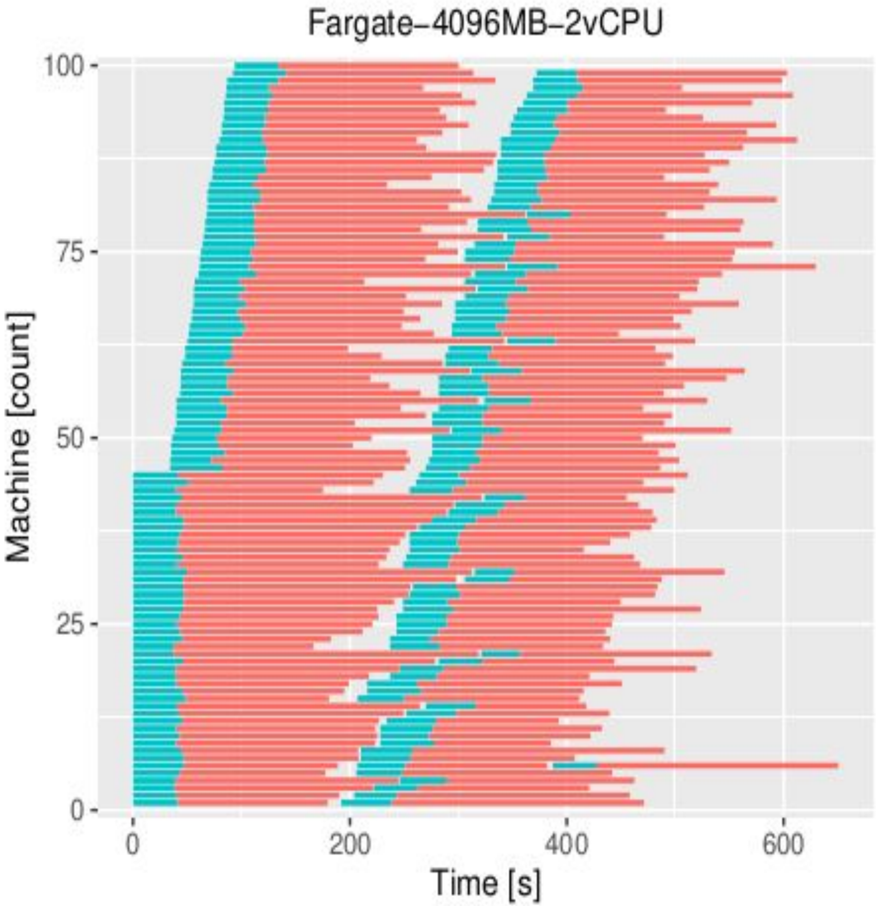


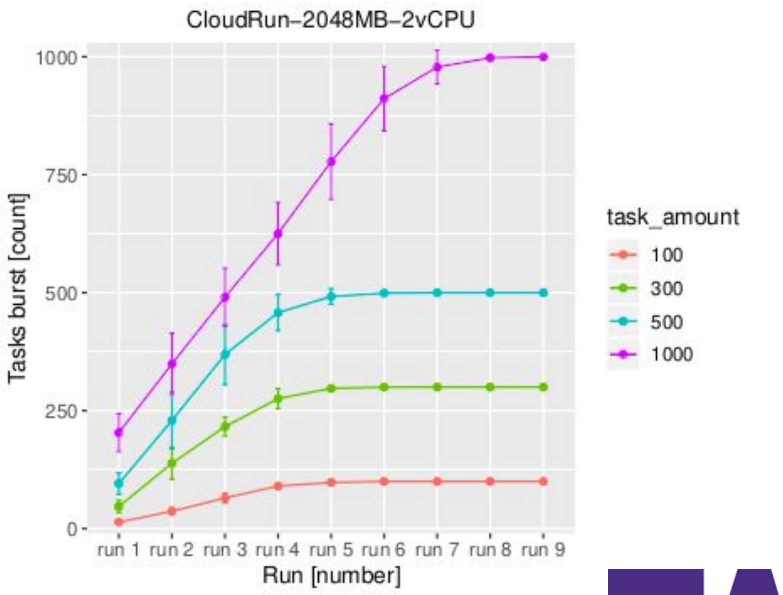
A

B



Comparing Cloud Run and Fargate limits-2





Hybrid approach - Fargate & Lambda

SoyKB workflow

- Many stages
- Different number of tasks
- Different execution time

Lambda

- Small-grained tasks

Fargate

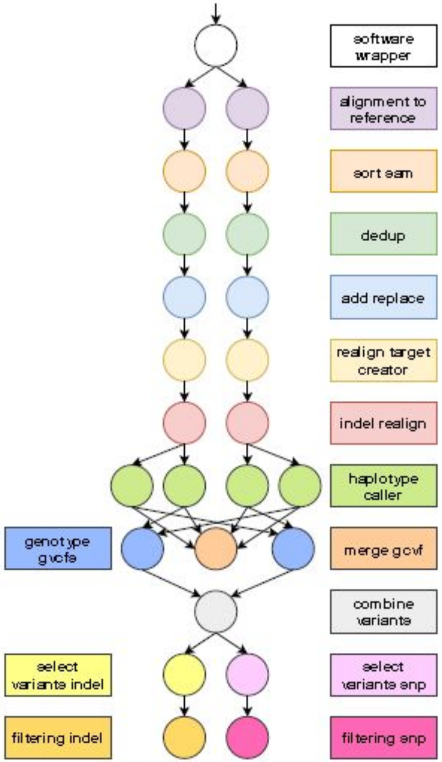
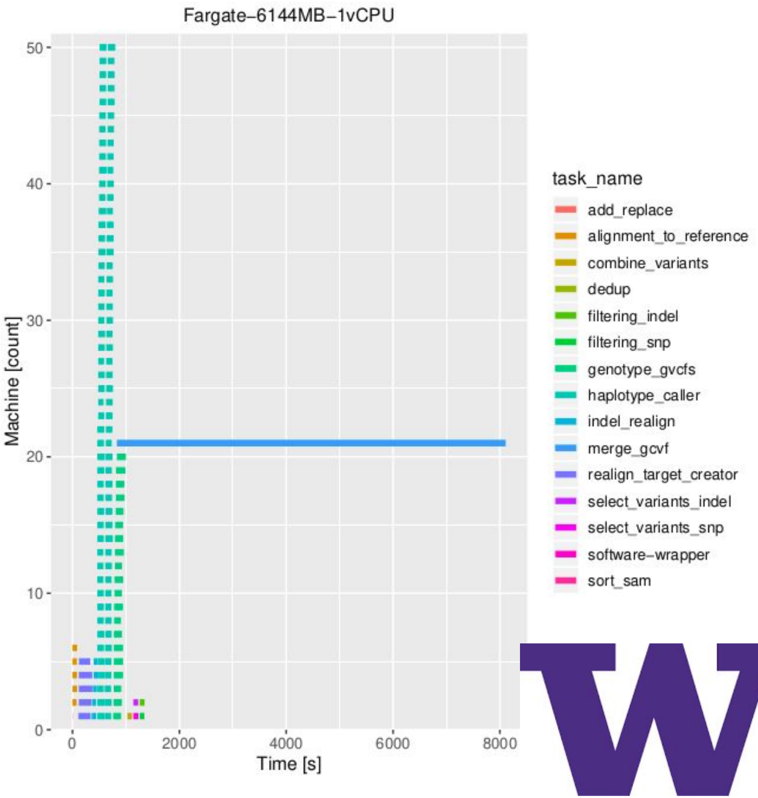


Figure 5.13: Structure of the SoyKB workflow.



Conclusion/ Takeaway

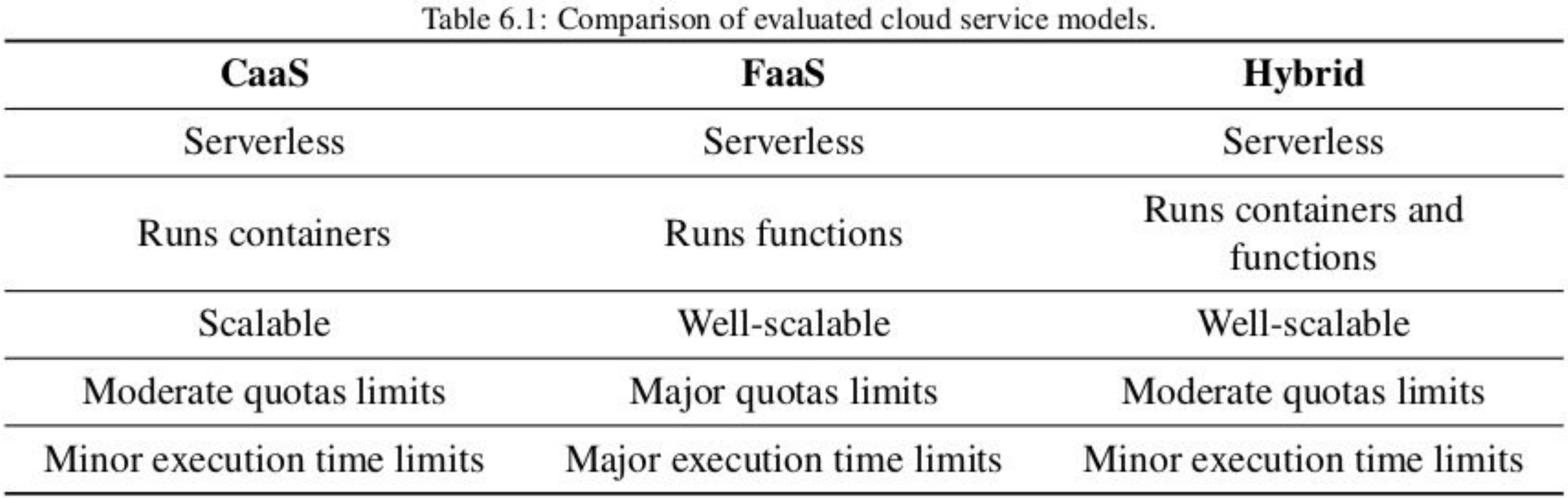


Table 6.1: Comparison of evaluated cloud service models.

CaaS	**FaaS**	**Hybrid**
Serverless	Serverless	Serverless
Runs containers	Runs functions	Runs containers and functions
Scalable	Well-scalable	Well-scalable
Moderate quotas limits	Major quotas limits	Moderate quotas limits
Minor execution time limits	Major execution time limits	Minor execution time limits

Strengths

- Detailed explanation
- Elasticity and Scalability
  - Workflow system does not need to manage resource decisions
- Hybrid approach
  - Choose task based on limits
  - Memory, disk space, or CPU requirements.

Weakness

- Caas and Scientific workflows

- Fargate memory limit coupled to vCPU value
  - May pay for extraneous memory when seeking CPU performance

- Limitations of Fargate
  - Fargate task limit
  - Burst rate Throttling Exception

Evaluation

- Authors don’t investigate the theorized AWS API limitations

- Overall workflow-to-model evaluation not rigorous enough
  - Only one or two workflows for each model
  - Only one data-intensive workflow (soyKB) evaluated

- CaaS viable for workflows?
  - To a degree, but has several limitations
  - Hybridized approach with FaaS necessary
  - Preliminary - more research necessary

GAPS & Future Work

- Lambda vs CloudRun (or) Google functions vs Cloud Run ?
- Other services (Azure)
- Extend prototype implementation
- Hybridization favored - what about PaaS?
- CPU allocation decisions crucial for CaaS but not discussed

THANK YOU!