Implications of Programming Language Selection for Serverless Data Processing Pipelines

Robert Cordingly, Hanfei Yu, Varik Hoang, David Perez, David Foster, Zohreh Sadeghi, Rashad Hatchett, Wes Lloyd

August 17-24, 2020

School of Engineering and Technology
University of Washington Tacoma
CBDCom 2020: IEEE International Conference on Cloud and Big Data

Outline

- Background and Motivation
- Research Questions
- Serverless Application Analytics Framework (SAAF)
- TLQ Pipeline and Static Code Analysis
- Experiments and Results
- Conclusions


aws
Google Cloud
Azure
IBM Cloud
3


aws

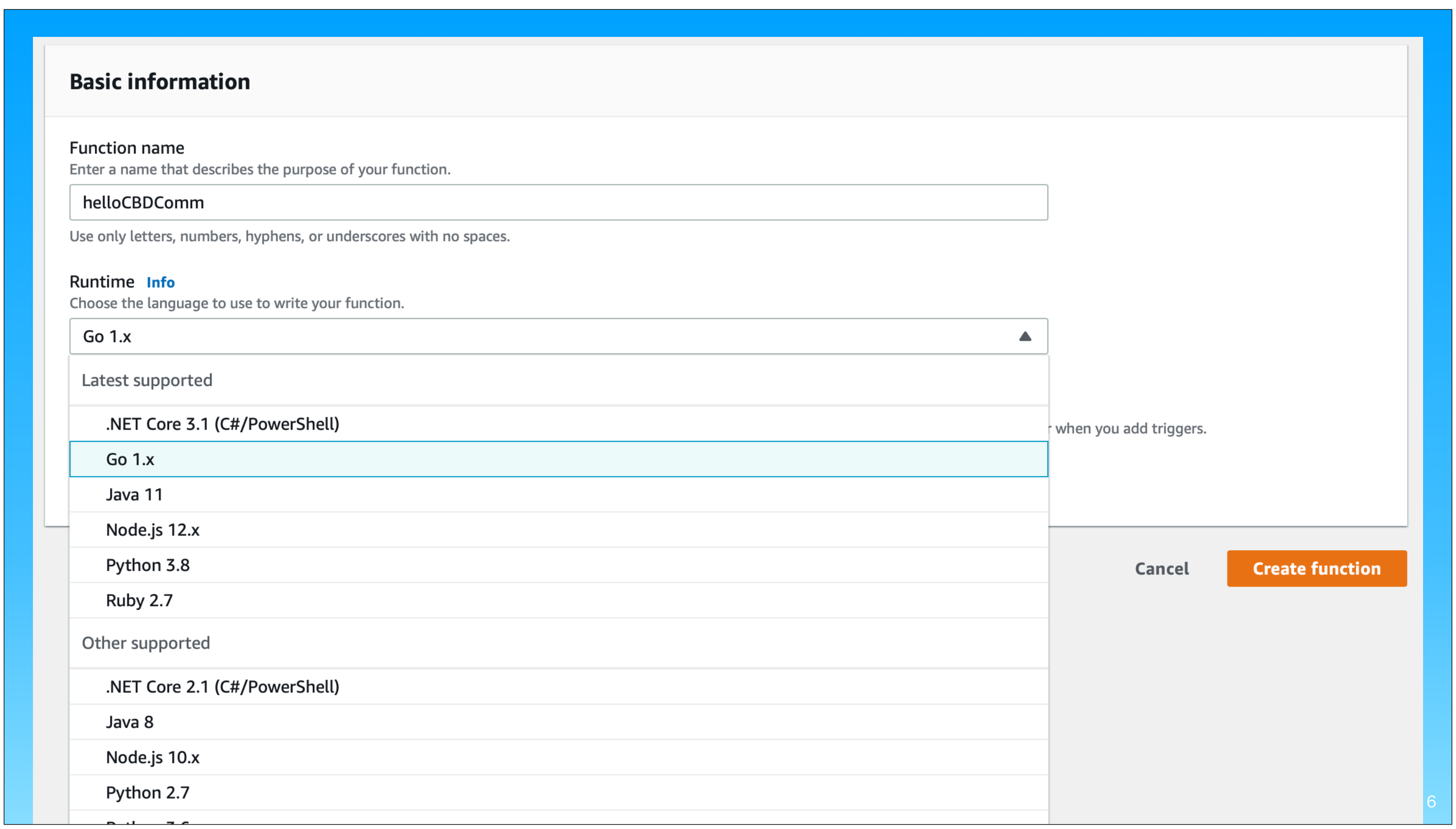
Serverless: Function-as-a-Service

- Developers create small applications called micro-services in a selection of supported languages by the cloud provider.
- Cloud providers automatically scale and manage cloud infrastructure instead of developers.

λ

The cost of FaaS:

- (Function Runtime) x (Memory Setting) x (Price)
- Billed only for runtime used.



Outline

- Background and Motivation
- Research Questions
- Serverless Application Analytics Framework (SAAF)
- TLQ Pipeline and Static Code Analysis
- Experiments and Results
- Conclusions

Research Questions

RQ-1: (Performance) How does the choice of programming language (Java, Go, Python, Node.js) impact the overall performance and throughput of a serverless data processing pipeline?

Research Questions

RQ-1: (Performance) How does the choice of programming language (Java, Go, Python, Node.js) impact the overall performance and throughput of a serverless data processing pipeline?

RQ-2: (Scalability) How does programming language choice impact the scalability of a serverless data processing pipeline when processing many concurrent data payloads?

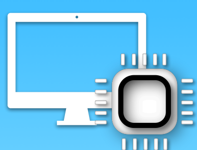
Research Questions

RQ-2: (Scalability) How does programming language choice impact the scalability of a serverless data processing pipeline when processing many concurrent data payloads?

RQ-3: (Infrastructure State) How does the choice of programming language impact cold FaaS performance compared to warm FaaS performance for a data processing pipeline?

Research Questions

RQ-3: (Infrastructure State) How does the choice of programming language impact cold FaaS performance compared to warm FaaS performance for a data processing pipeline?

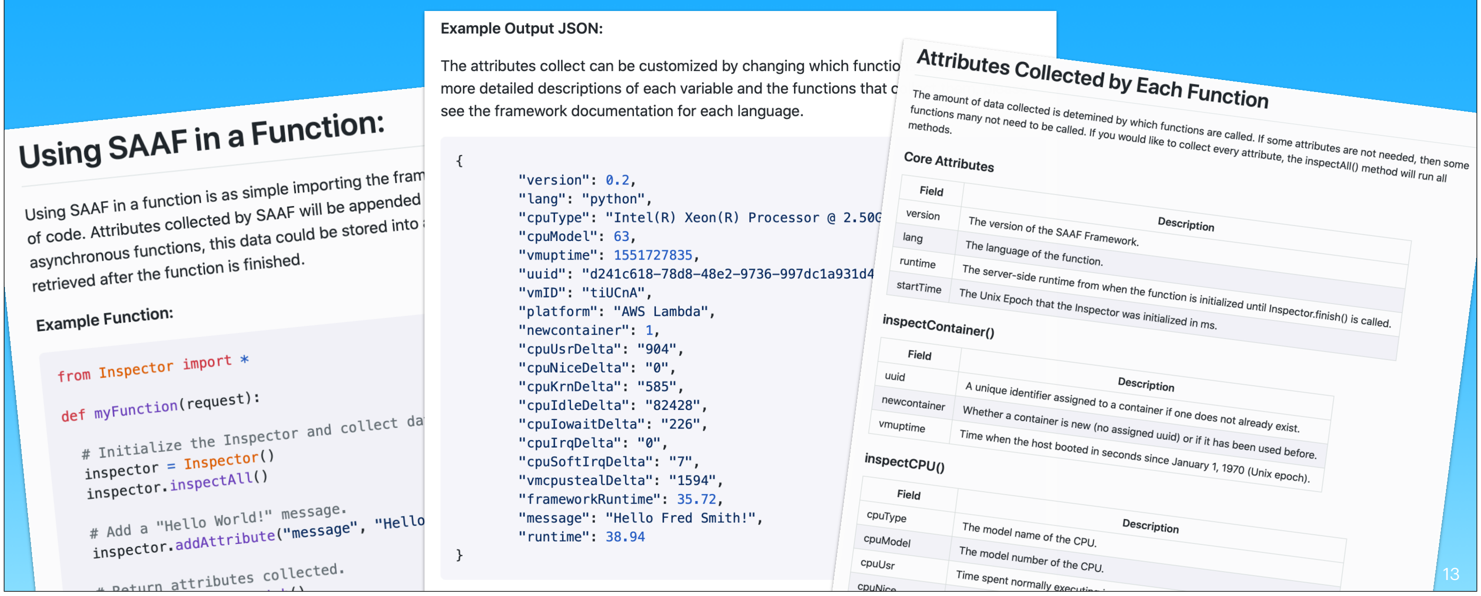


RQ-4: (Memory/Cost) How does performance vary for a serverless data processing pipeline across alternate memory settings for implementations in different languages.

Outline

- Background and Motivation
- Research Questions
- → Serverless Application Analytics Framework (SAAF)
- TLQ Pipeline and Static Code Analysis
- Experiments and Results
- Conclusions

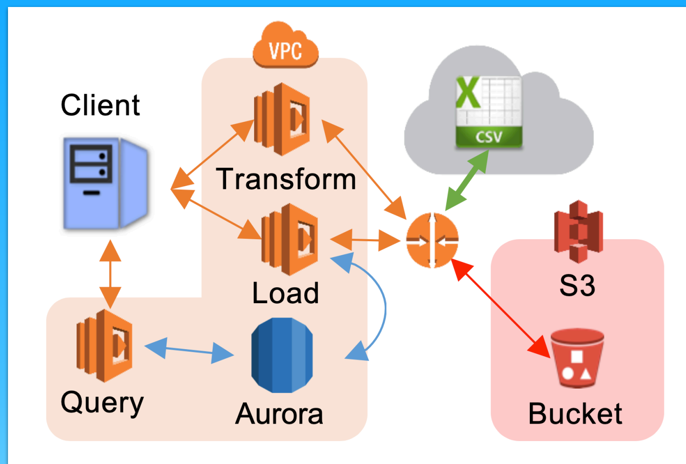
Serverless Application Analytics Framework
(SAAF)



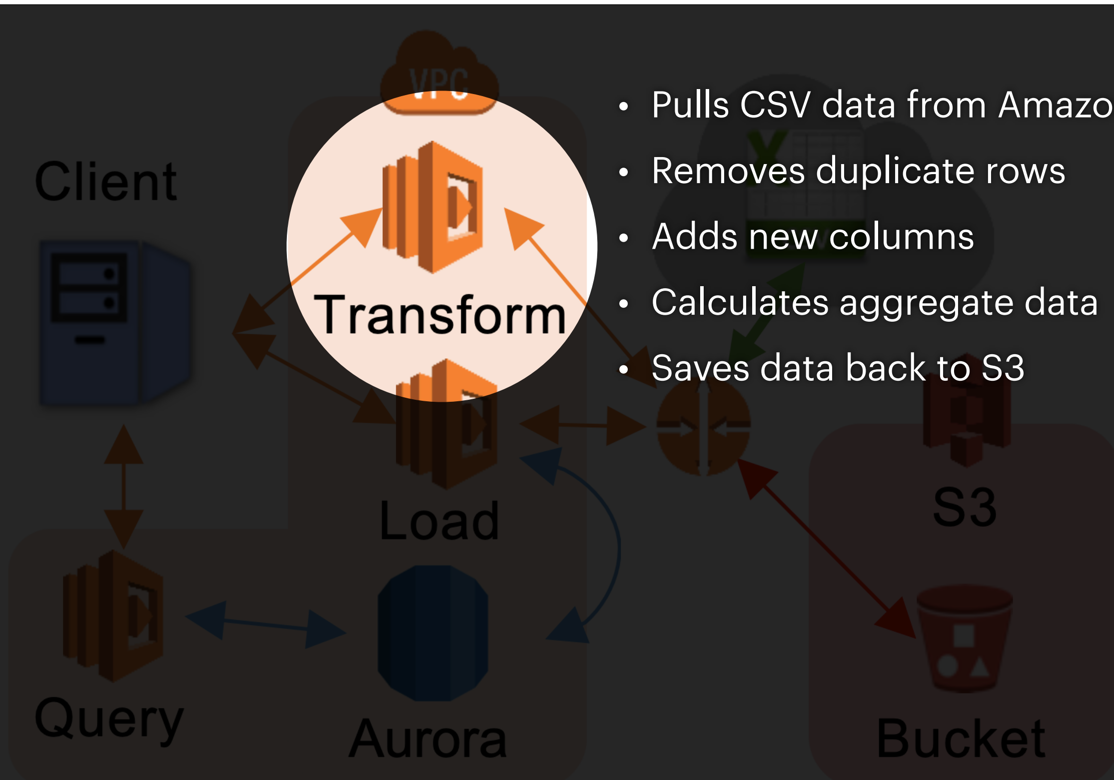
Outline

- Background and Motivation
- Research Questions
- Serverless Application Analytics Framework (SAAF)
- TLQ Pipeline and Static Code Analysis
- Experiments and Results
- Conclusions

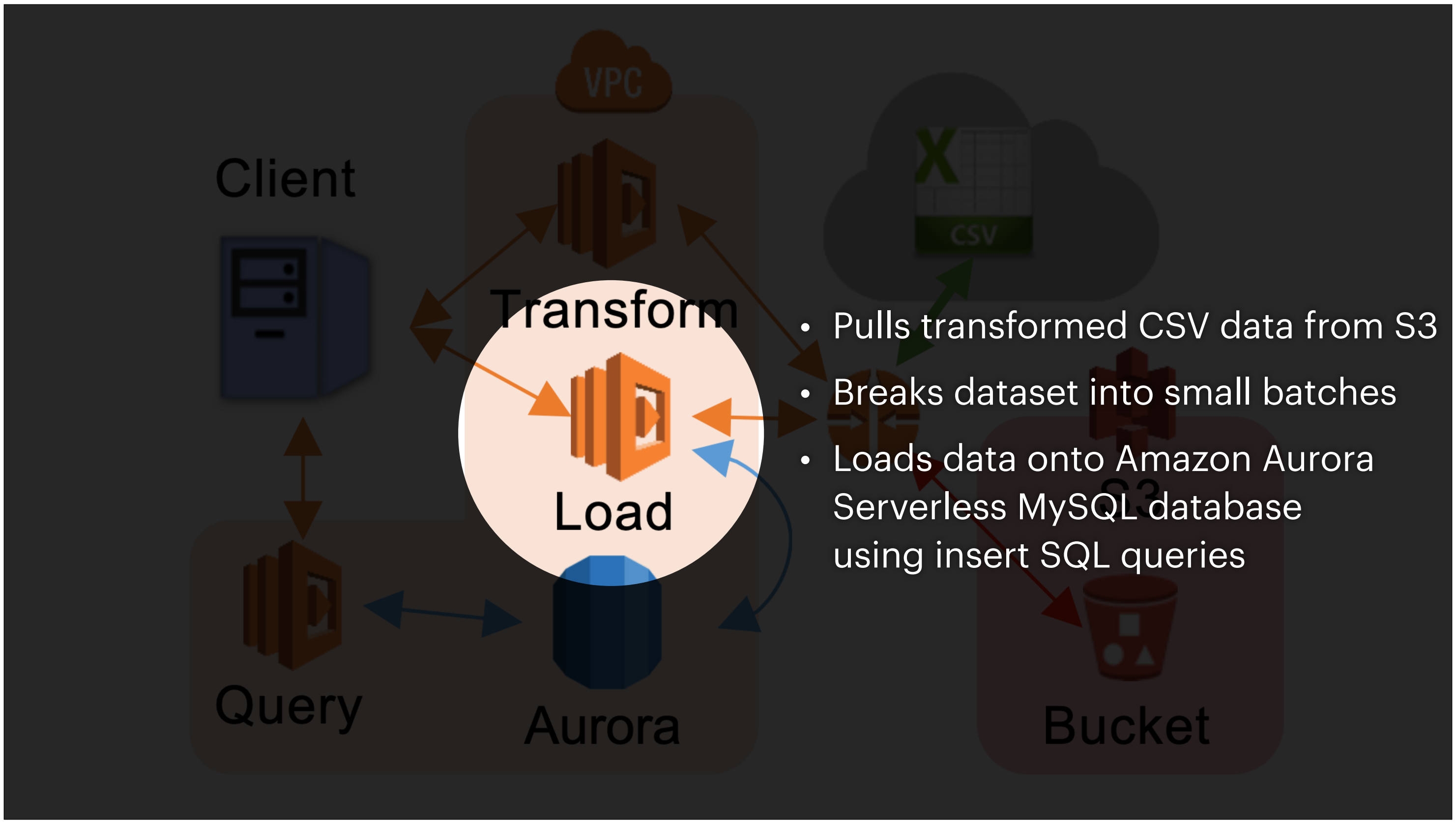
Transform-Load-Query Pipeline



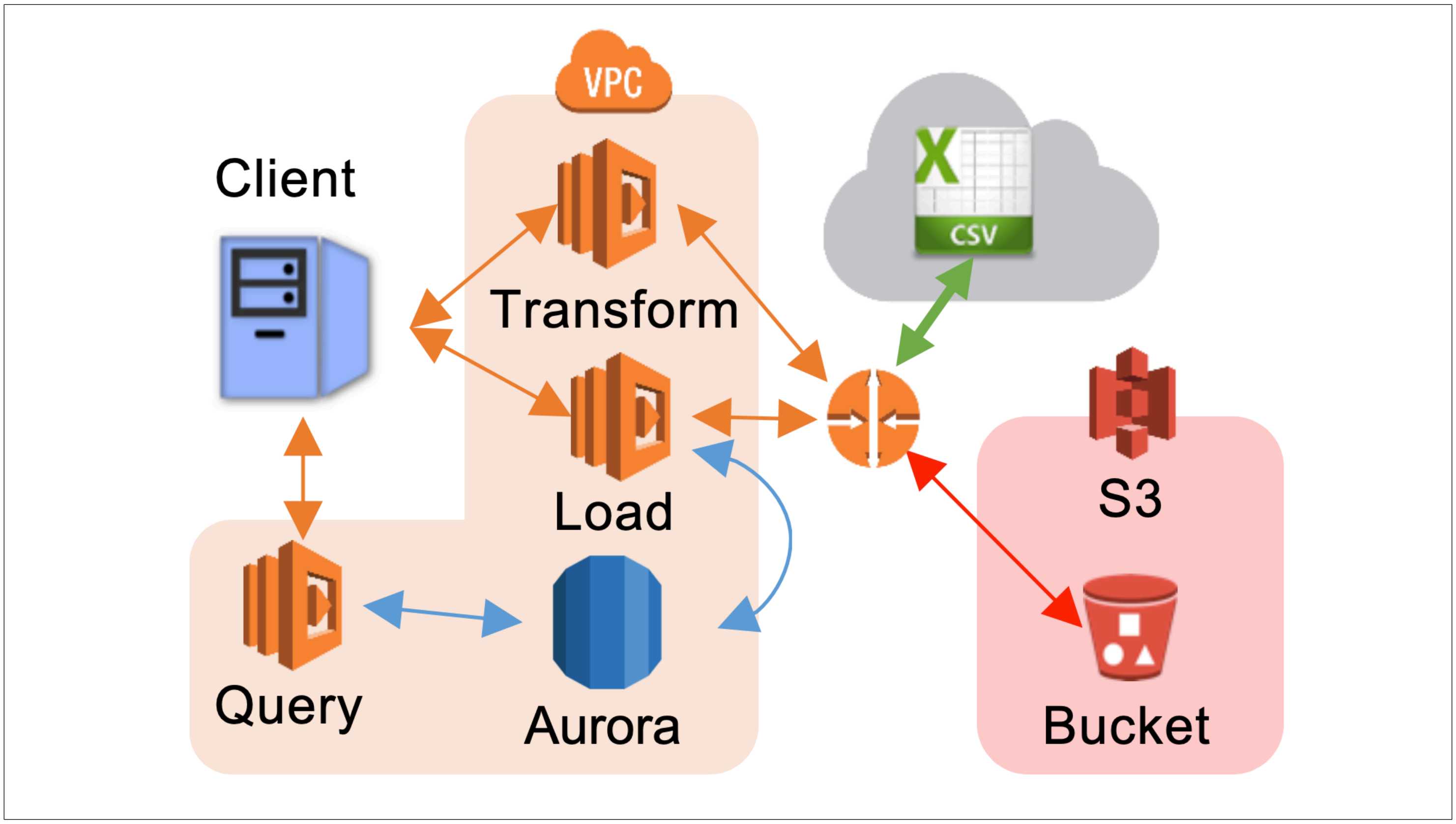
We developed a three-function data processing pipeline creating functionally identical versions in Java, Go, Node.js, and Python.



- Pulls CSV data from Amazon S3
- Removes duplicate rows
- Adds new columns
- Calculates aggregate data
- Saves data back to S3


VPC
Client
Transform
CSV
Load
Query
Aurora
Bucket
Pulls transformed CSV data from S3
Breaks dataset into small batches
Loads data onto Amazon Aurora Serverless MySQL database using insert SQL queries


VPC
Client
Transform
CSV
Load
S3
Query
Aurora
Bucket
Executes 5 aggregate queries
Results combined with JOIN query
Saves results back to S3
Additionally executes SELECT * on all data



Static Code Analysis

Service	Lang	Funcs	Vars	SLOC	Loops	Cloud Service Usage
Transform	Java	3	40	86	2	S3 Get/Put
Transform	Python	3	28	64	3	S3 Get/Put
Transform	Go	3	30	77	1	S3 Get/Put
Transform	Node.js	3	24	96	1	S3 Get/Put
Load	Java	3	25	77	2	S3 Get, DB Conn x1
Load	Python	3	21	57	3	S3 Get, DB Conn x1
Load	Go	3	15	65	1	S3 Get, DB Conn x1
Load	Node.js	4	18	83	1	S3 Get, DB Conn x1
Query	Java	4	36	111	7	S3 Put, DB Conn x2
Query	Python	5	44	96	9	S3 Put, DB Conn x2
Query	Go	4	34	104	8	S3 Put, DB Conn x2
Query	Node.js	5	17	74	1	S3 Put, DB Conn x2

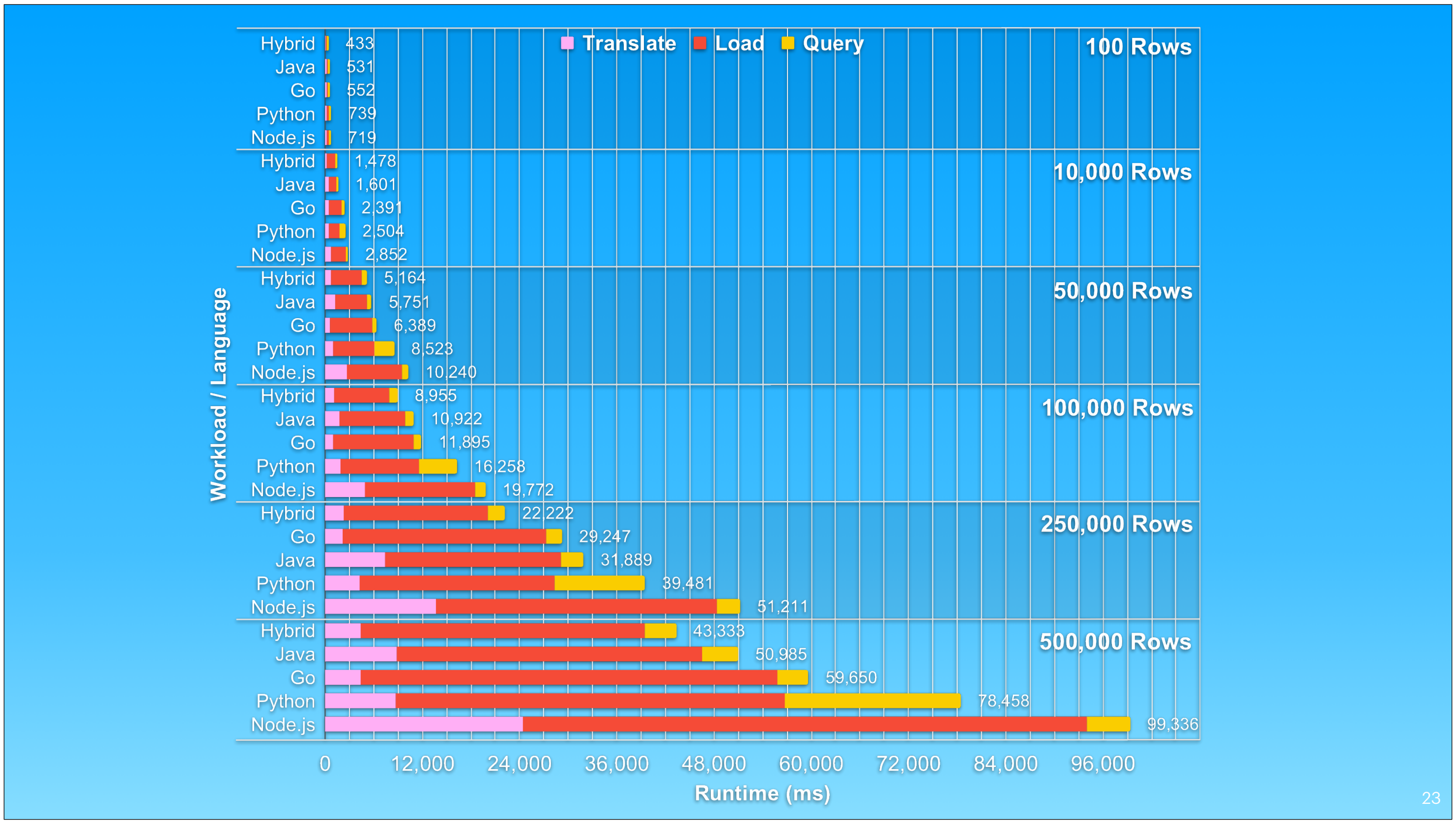
Code Available at github.com/wlloyduw/FaaSProgLangComp

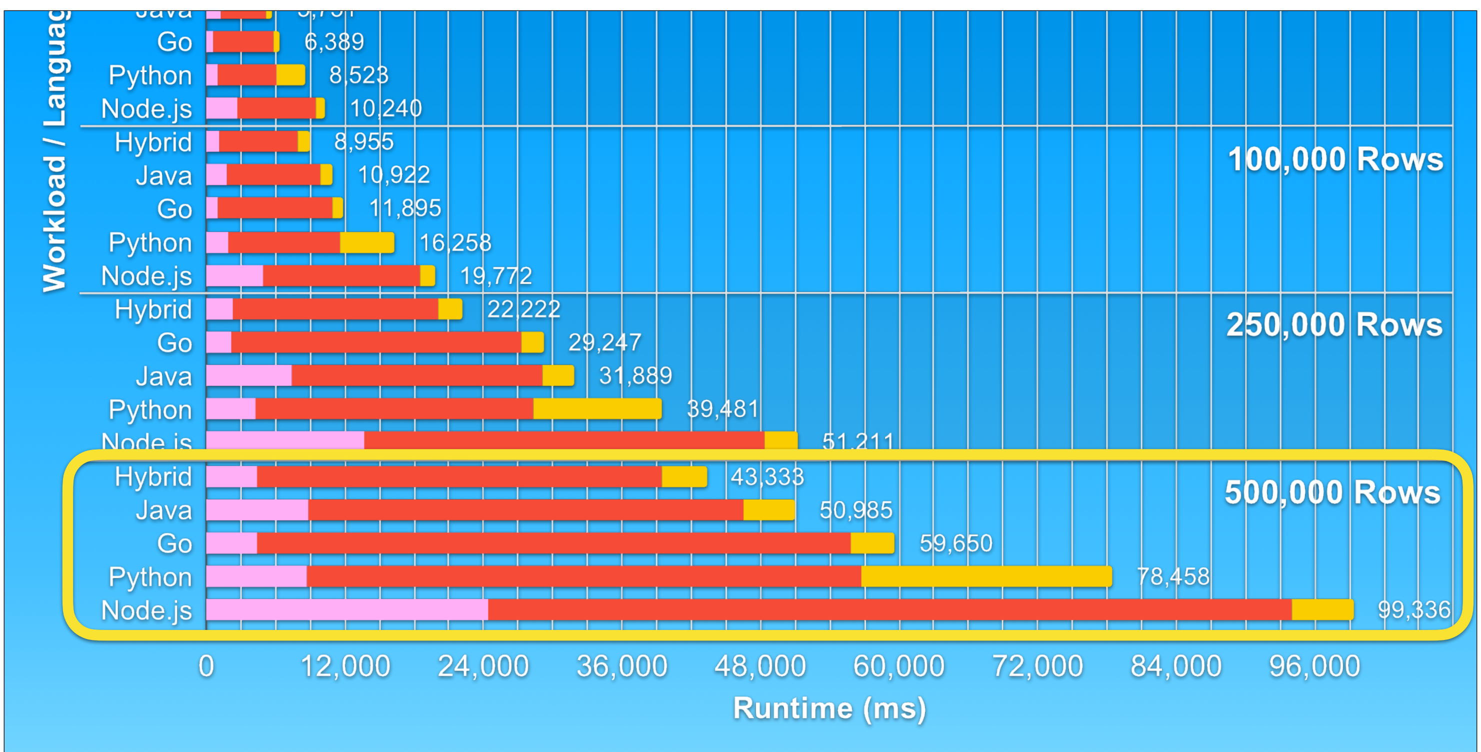
Outline

- Background and Motivation
- Research Questions
- Serverless Application Analytics Framework (SAAF)
- TLQ Pipeline and Static Code Analysis
- ➡ Experiments and Results
- Conclusions

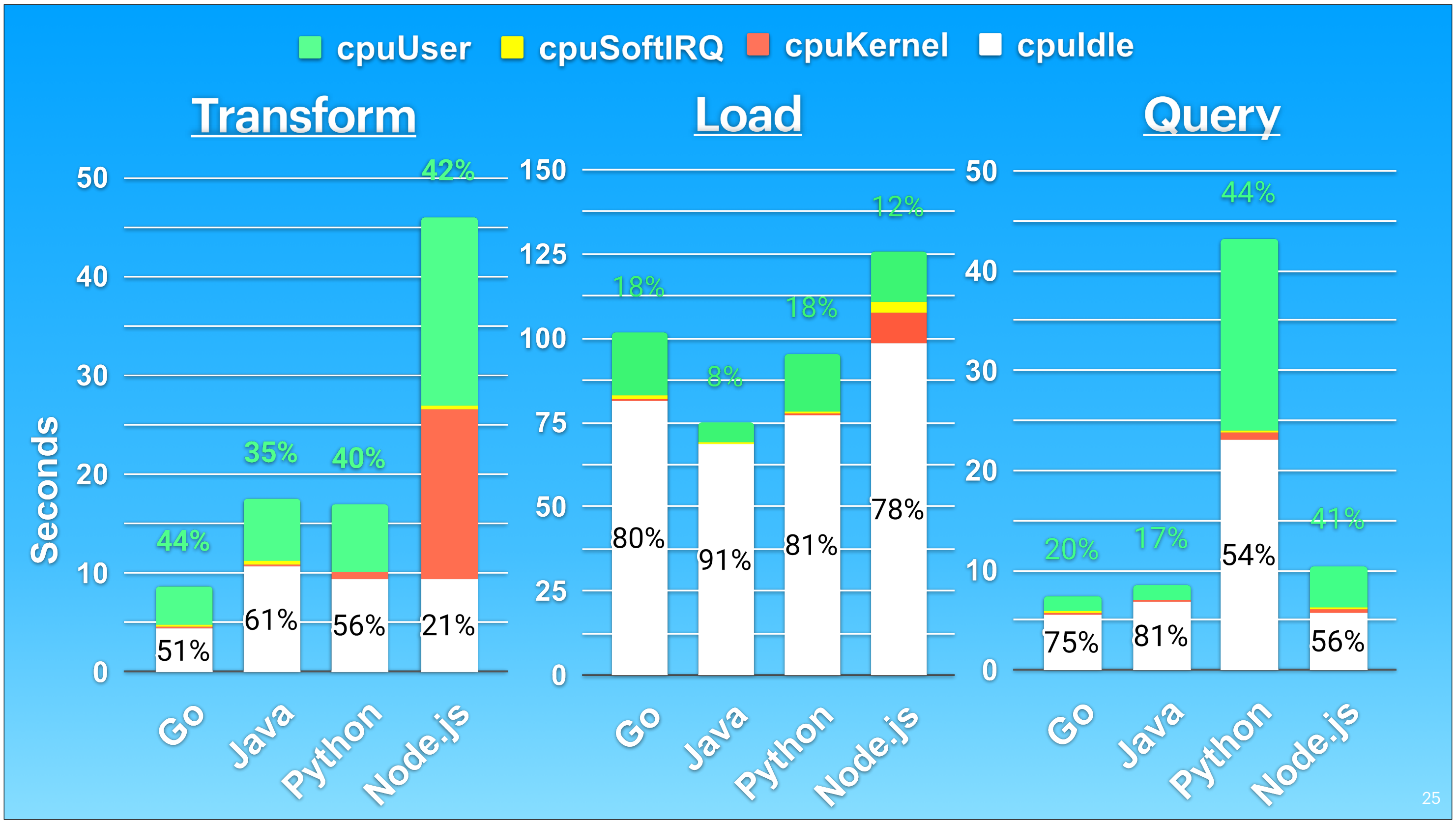
Experiment 1: Overall Performance Comparison

Compare function runtime across different workload sizes.



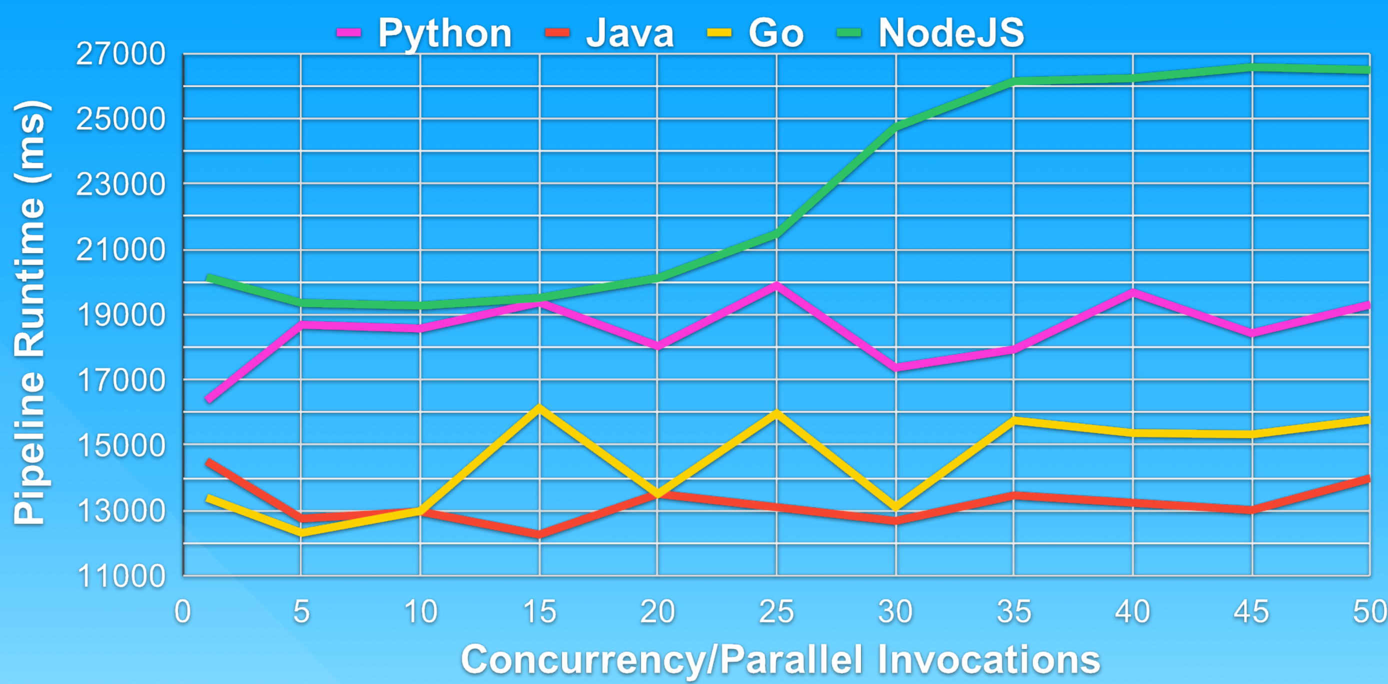


Hybrid Pipeline outperformed Java by 17%, Go by 37%, Python by 81%, and Node.js by 129%.



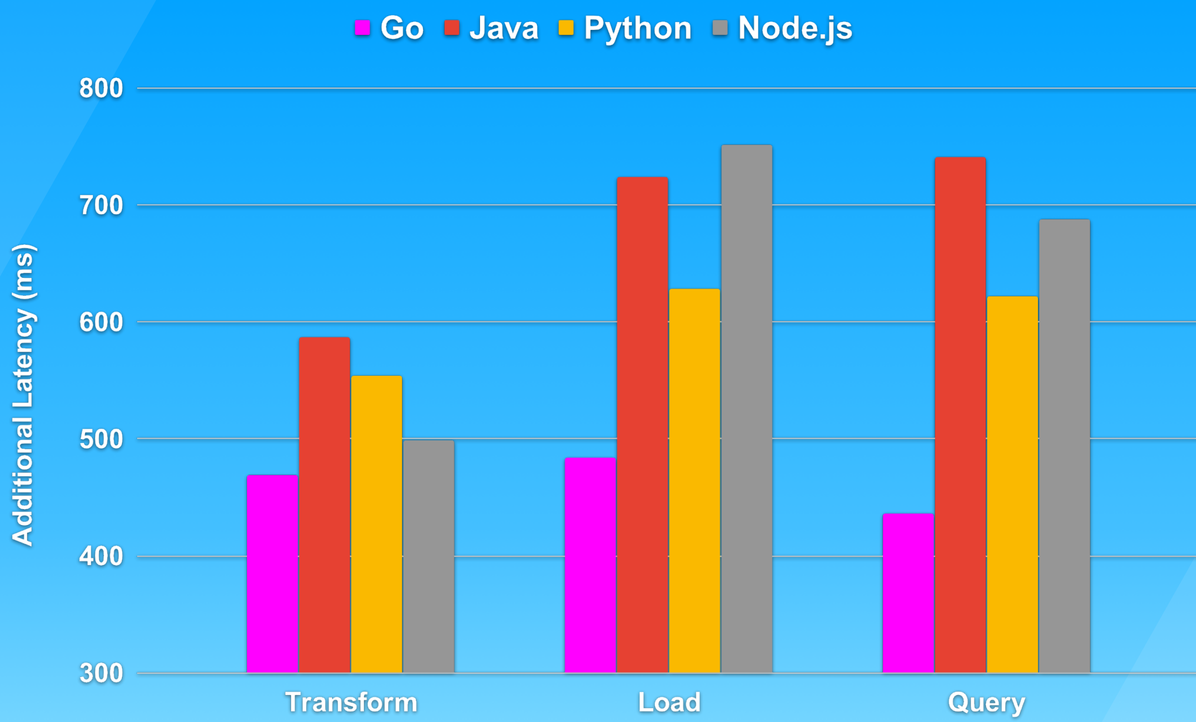
Experiment 2: Scalability Performance Testing

Compare function runtime as the number of concurrent calls is increased.



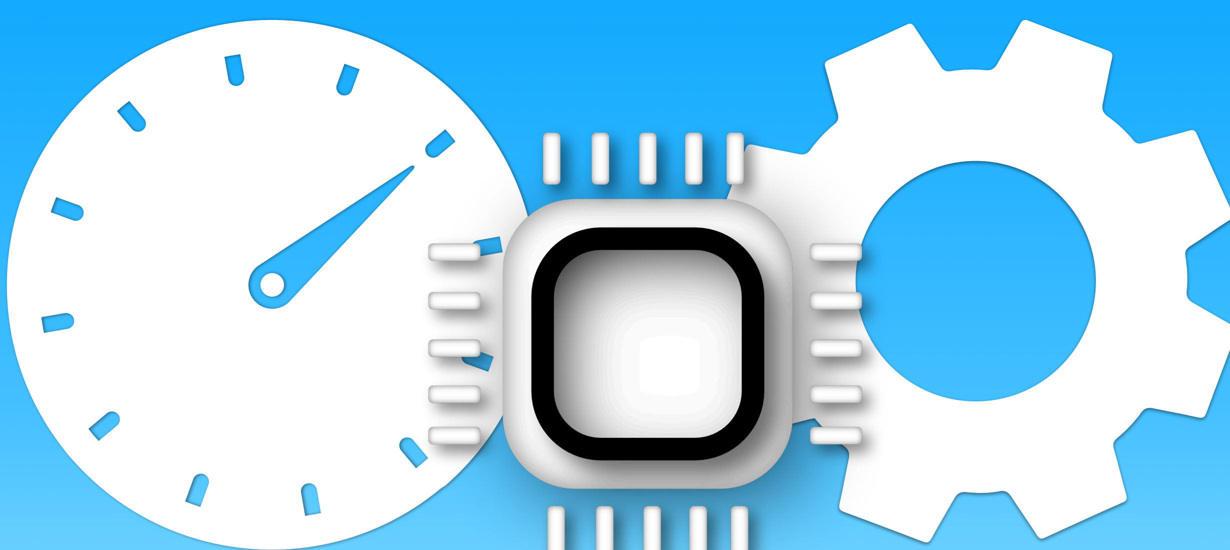
Experiment 3: Cold/Warm Performance

Compare function latency between cold and warm FaaS Infrastructure.

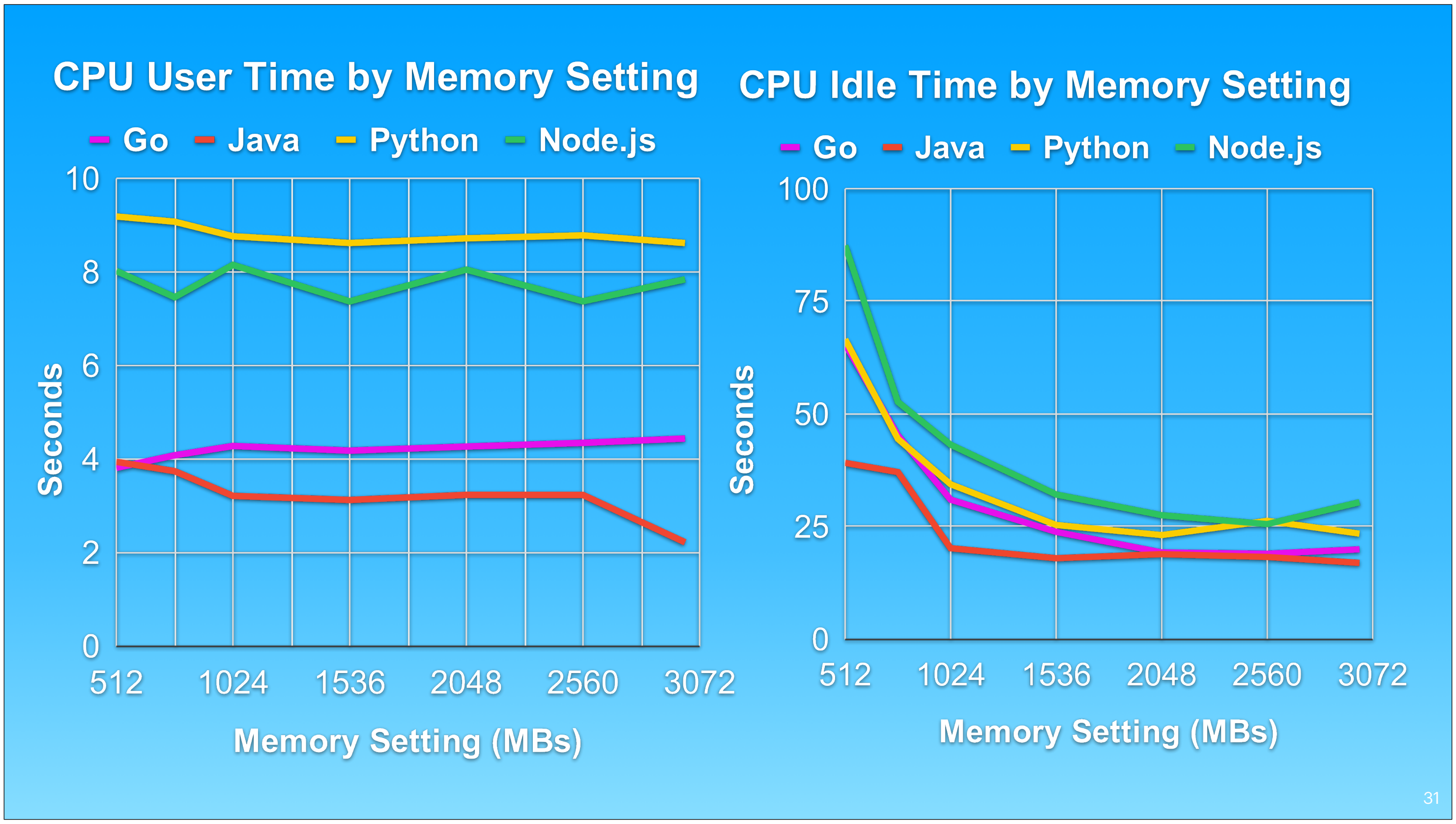


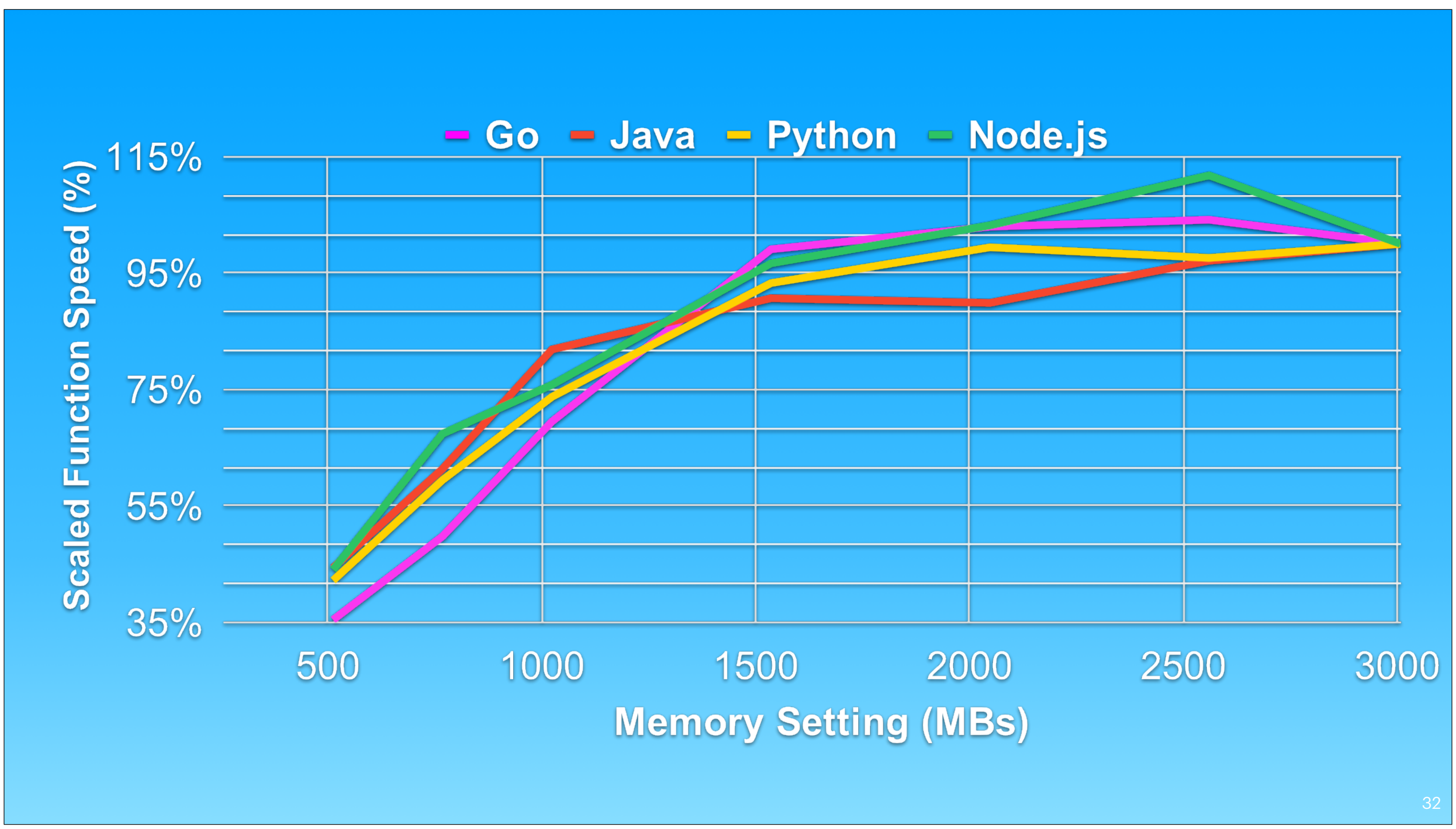
Go: 463 ms, Java: 684 ms, Python 602 ms, Node.js 645 ms

Experiment 4: Memory Configuration Comparison



Compare FaaS performance scaling as memory setting is changed.

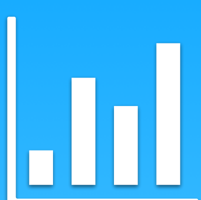

CPU User Time by Memory Setting
Go Java Python Node.js
Seconds
10
8
6
4
2
0
512 1024 1536 2048 2560 3072
Memory Setting (MBs)
CPU Idle Time by Memory Setting
Go Java Python Node.js
Seconds
100
75
50
25
0
512 1024 1536 2048 2560 3072
Memory Setting (MBs)
31


Go Java Python Node.js
Scaled Function Speed (%)
115%
95%
75%
55%
35%
500 1000 1500 2000 2500 3000
Memory Setting (MBs)
32

Outline

- Background and Motivation
- Research Questions
- Serverless Application Analytics Framework (SAAF)
- TLQ Pipeline and Static Code Analysis
- Experiments and Results
- ➡ Conclusions

Conclusions



RQ-1: (Performance) How does the choice of programming language (Java, Go, Python, Node.js) impact the overall performance and throughput of a serverless data processing pipeline?

For a single language, Java offered the best performance, outperforming Node.js by 94%. The fastest pipeline used a hybrid combination of both Go and Java functions.

Conclusions

RQ-2: (Scalability) How does programming language choice impact the scalability of a serverless data processing pipeline when processing many concurrent data payloads?

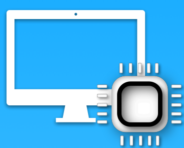
All languages performed similarly with Node.js performing negatively for workloads with higher concurrency.

Conclusions

RQ-3: (Infrastructure State) How does the choice of programming language impact cold FaaS performance compared to warm FaaS performance for a data processing pipeline?

Java, Python, and Node.js had similar latency, while Go had about 33% less latency than Java.

Conclusions



RQ-4: (Memory/Cost) How does performance vary for a serverless data processing pipeline across alternate memory settings for implementations in different languages.

Performance scaled approximately linearly for memory sizes up to 1.5 GBs for all pipelines. Beyond 1.5 GB, no major performance improvements were observed.

Thank You for Watching

Questions or comments?

Please email:

rcording@uw.edu or wlloyd@uw.edu

Download Serverless Application Analytics Framework

github.com/wlloyduw/saaf

This research is supported by NSF Advanced Cyberinfrastructure Research Program (OAC-1849970), NIH grant R01GM126019, and the AWS Cloud Credits for Research program.