UNIVERSITY of WASHINGTON

Serverless Computation with OpenLambda

Raaghavi Sivaguru
Ramya Kumar
Sindhuja Chandran
Sujanasree Ratakonda

Talk Outline

- Paper Overview
- Introduction to serverless Computing
- Background / Research Challenges
- Summary of new technology, approach or benchmarks
- Architecture
- Author's Evaluation and conclusion
- Critique: Strengths
- Critique: Weakness
- Critique: Evaluation
- Future Work

Paper Overview

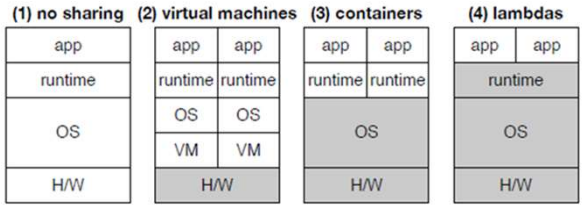
- Open Lambda - Open source serverless computing model
- Proposal - Building an RPC-aware storage engine that runs alongside serverless-computing platforms such as Google Cloud Functions, AWS Lambda.
- Facilitates work in research avenues of serverless computing
- Lambda Bench - benchmarking suite

Introduction

- Problem
  - Noisy neighbor
  - Security leaks
  - Startup latency
- Solution:
  - Sharing as much as possible
  - Isolation

Introduction

- Evolution of sharing

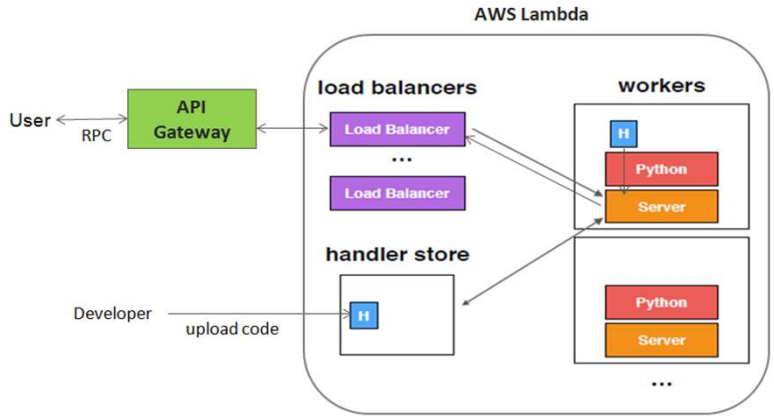


Introduction

- Lambda Model
  - No Server Management for developers
  - Auto-scaling & Elasticity
  - High Availability
  - Functions share the runtime environment
  - Minimize startup latency

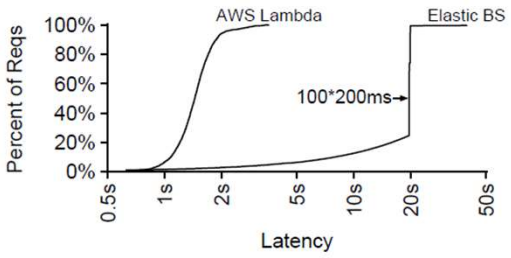
Background/Related Work

- Programming Model in Lambda



Background/Related Work

- Containers Vs Lambda (Response times)



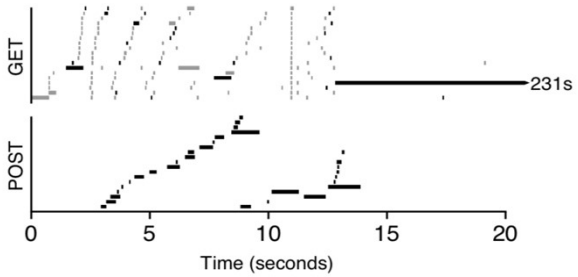
- Median 1
- Median 1

Background/Related Work

- Advantages of Lambda
  - Resource management is handled by cloud provider
  - Automatically scales without any configuration
  - Responds faster
  - Pay-as-you-go
  - Offers millisecond level billing granularity

Lambda Workload

- Example client-to-server pattern
  - Google gmail
- Trace RPC calls using chrome extension
- Long polling



Design Implications

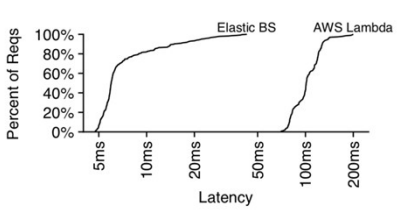
- RPCs shorter than 100ms
- AWS Lambda charges are 3.7x more
- Idle handlers dominate cost of application

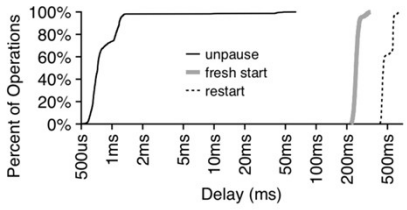
Research challenges

- Execution Engine
- Interpreted Languages
- Package Support
- Cookies and Sessions
- Databases
- Data Aggregators
- Load Balancers
- Cost Debugging
- Legacy Decomposition

Execution Engine

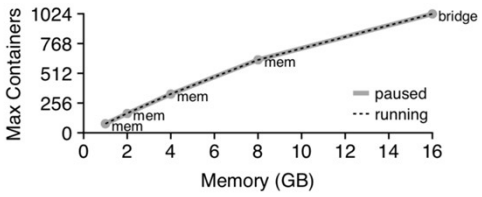
- Sandbox for executing handlers
- Trade Offs happens





Execution Engine

- High memory cost
- Memory is main bottleneck
- Prevents starting new nodes



Interpreted Language

- Just-in-time compilers
- Trade offs
- Alternative
  - Dynamic optimization

Package Support

- Rapidly spin up
- Bundle third-party libraries
- Increase startup latency
- Alternative
  - Package aware
  - Eg: npm-Node.js, pip-Python

Cookies and Sessions

- Short-lived and stateless
- Shared view of cookie state
- Maintains TCP connections
- Alternative
  - Management of TCP connections

Database

- Handler waiting for DB
- Change feed Batching

Data Aggregation

- Parallelism over data shares
- Lambda - Tree Structure
- Solution - Custom data store to coordinate with Lambdas
- Challenge - Different platforms for pre-processing

Load Balancing

Schedulers must consider

- Session Locality
- Code Locality
- Data locality

Cost

- Based on each RPC call/page
- Database operation performed by each lambda

Legacy System

Decompose web applications into Lambda based services

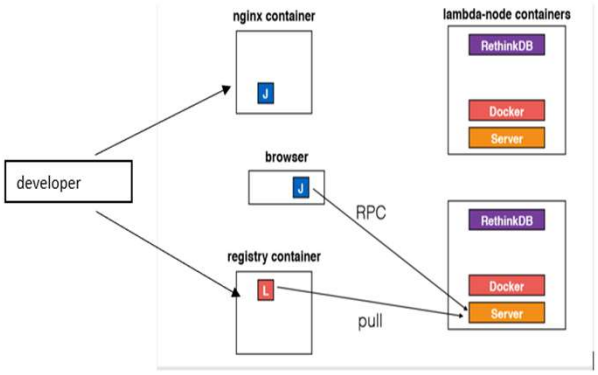
Open Lambda

- OpenLambda is an Apache-licensed serverless computing project, written in Go and based on Linux containers.
- There are several opportunities for tight integration:
  - Client RPC calls introduce consistency boundaries
  - RPC schedulers aware of database replicas
  - Coordinating handler pausing with database change batches.
- Database will coordinate with handler workers via gRPC calls.

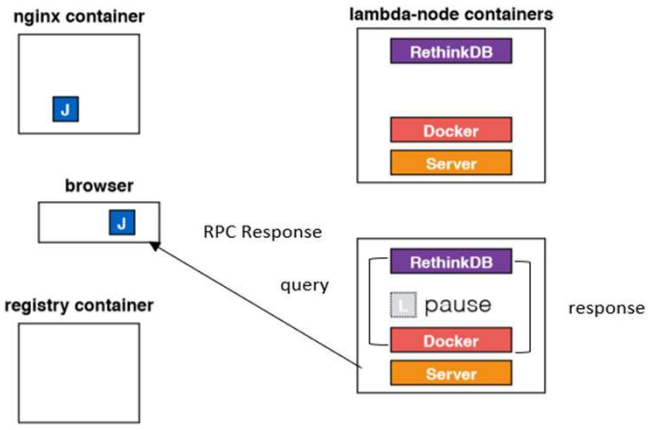
Components

- worker: Lambda server that executes handlers
- nginx: load balancer
- lambda-generator: old script for generating Python Lambdas
- node: container with worker, rethinkdb, and docker
- util: scripts for starting/stopping local cluster
- applications: OpenLambda applications
- testing: initial unit test environment

Workflow



Workflow



Development

- Online Project : http://www.open-lambda.org.
- Code - Open source – https://github.com/open-lambda/open-lambda#architecture
- Slack Development Channel: https://open-lambda.slack.com

Authors Evaluation & Conclusion

- Detailed overview of serverless computation and how sharing between applications evolved
- Overview of research challenges with serverless computing platform

Critiques: Strengths

- Base to new lambda
- Details on Limitations
- Lambda Efficiency
  - Sharing runtime seems to be compromising
- Study on Load burst and steady Light Load
- Project is funded by NSF and donations from top companies

Critiques: Weakness

- In paper, open lambda architecture is not being explained in detail
- Related interactions are not possible
- Complexity and usability of open lambda

Identify GAPS

- Only RPC trigger events evaluated
- Dynamic profiling is not possible
- No benchmark tools
- Open lambda is not materialized

FutureWork

- Reduced Memory Cost
- Supporting more runtime
- Coordination between lambdas to support data aggression

References

- http://www.cs.cmu.edu/~coda/docdir/s11.pdf
- https://aws.amazon.com/lambda/
- https://leveros.readme.io/
- https://serverless.com/

Q&A