# Cloud Language Runtimes on IA - Challenges & Opportunities

**Suresh Srinivas**
Principal Engineer, Intel
suresh.srinivas@intel.com
@ssuresh

Contributors:
Uttam, Dunni, Ahmad, Pranitha, Florin, Sushma, Catalin, Vish

# Agenda

Cloud Runtime Usage & Introduction

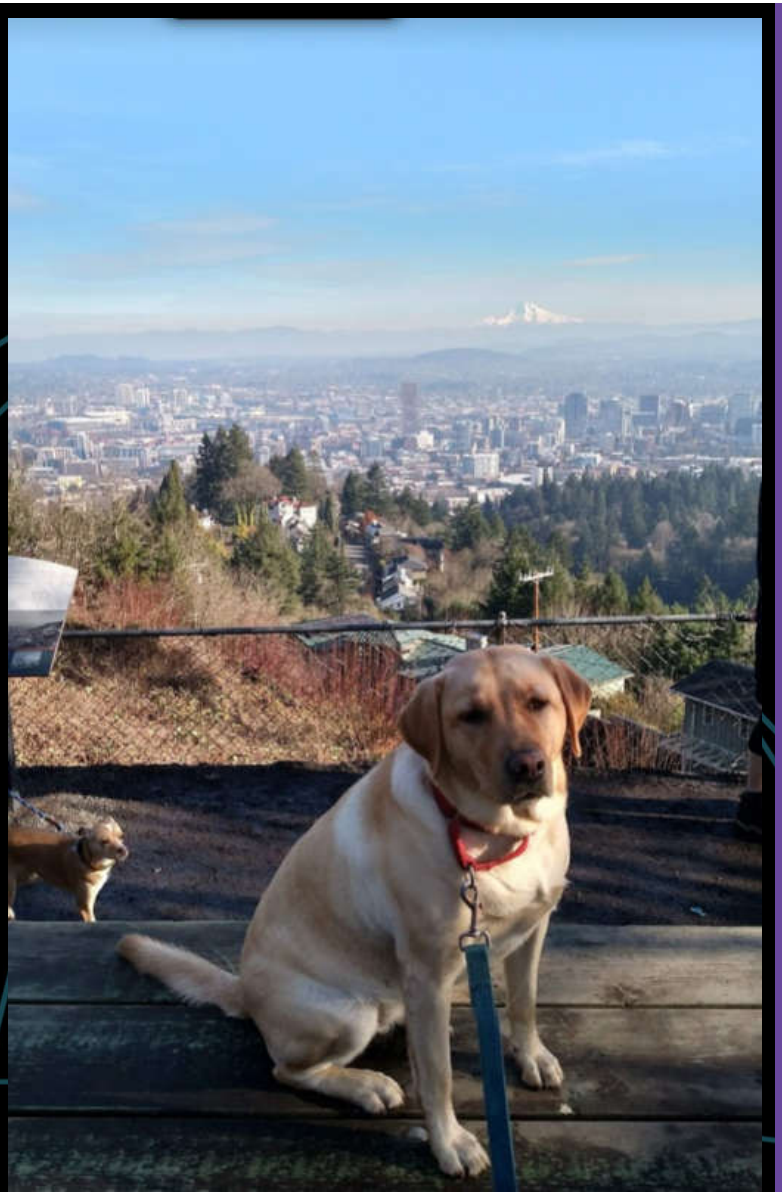
Cloud Runtime Challenges/Problems

Software/Hardware Optimization

Next Steps/Open Problems/Call To Action

# Cloud Runtimes

Usage/Introduction



Title: Luna at Pittock Mansion
Author: Suresh Srinivas
License: Creative Commons
Source: @sweetlunatheyellowlab on Instagram

# Cloud Runtime: Usages/Language

## Multiple Environments

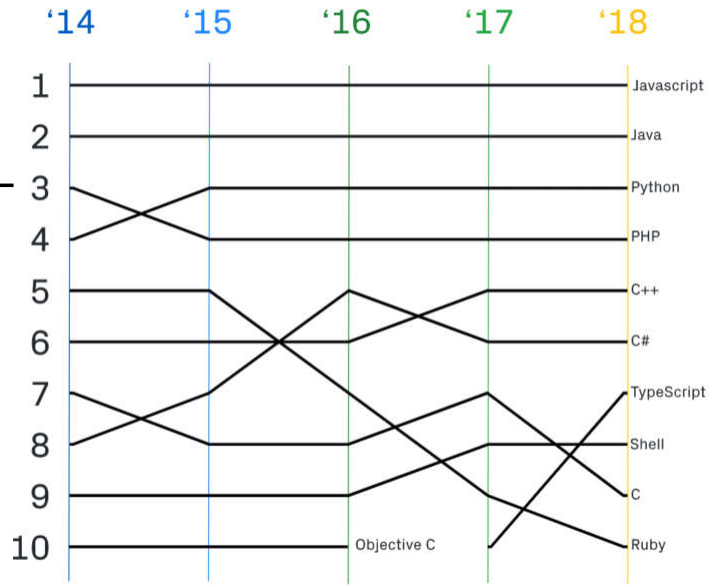
- IaaS->PaaS->FaaS

## Languages

- 4 of top 5 are Runtimes
- 3 are dynamically typed.

## Usages

- Function As Service
- WebTier
- Analytics

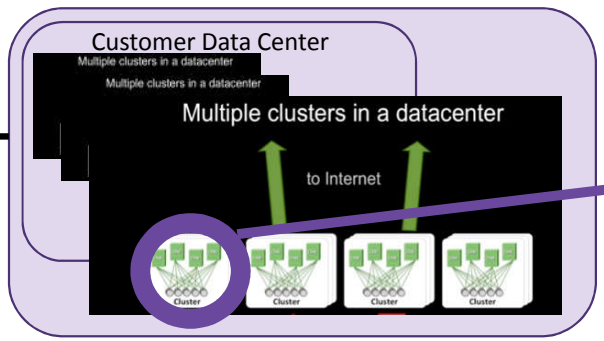



https://octoverse.github.com/projects#languages

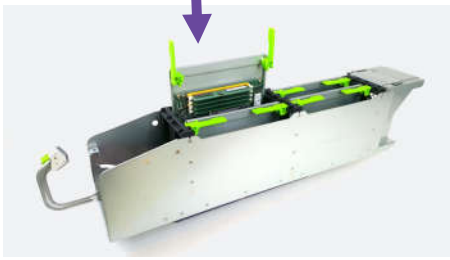
*Other names and brands may be claimed as the property of others.

# Cloud Runtime: Web Tier Scale Out Usage

- Each server running a Web Server
- Individual server can run at high utilization (> 90%)
- Each server also running Cloud Runtime
- Scale Out
  - Example: Cluster of Dense Computing Platform with Yosemite v2 (Four 1S servers)
- PHP, Python, Node.js …
- Very different from SPEC benchmarks (Large Scattered Code, JITting, Type Checks, …)







# Cloud Runtimes

Challenges/Problems

Title: Luna at Cannon Beach
Author: Suresh Srinivas
License: Creative Commons
Source: @sweetlunatheyellowlab on Instagram

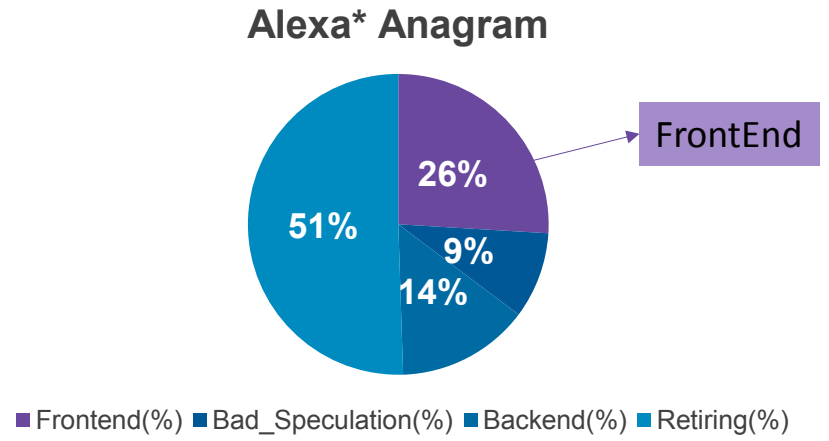
# FaaS JavaScript: Challenges

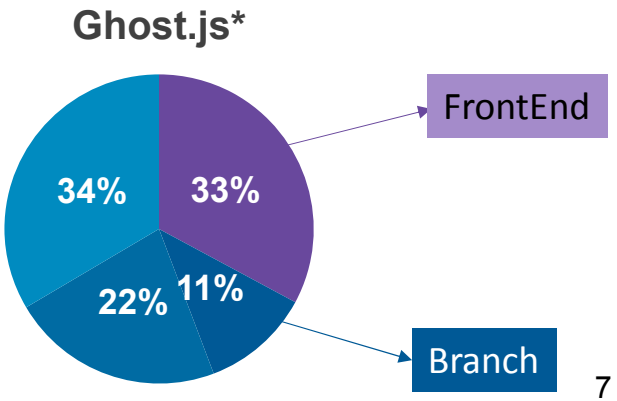
## High CPU bottlenecks ~35-45% stalled

- I-Cache Misses
- I-TLB Misses
- Branch Mispredicts

## Problem gets worse for bigger function

| Anagram | Ghost.js |
|---|---|
| 51% retiring | 34% retiring |
| 1.18% I-TLB misses | 6% I-TLB Misses |
| 1.25% I-$ misses | 10% I-$ Misses |



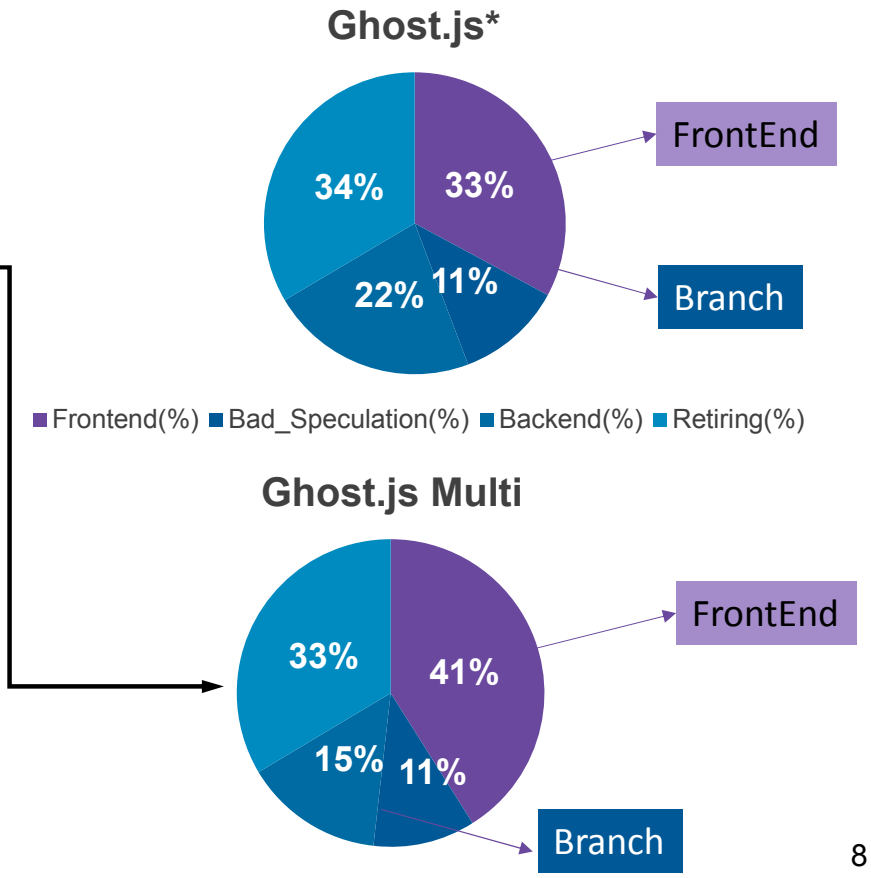




*Other names and brands may be claimed as the property of others.

# FaaS JavaScript: Characteristis & Challenges

## Problem gets worse

- Bigger Functions
- When multiple functions executing concurrently
- Frontend Stalls increase from **33% to 41%**
- Cycles/work much higher




*Other names and brands may be claimed as the property of others.

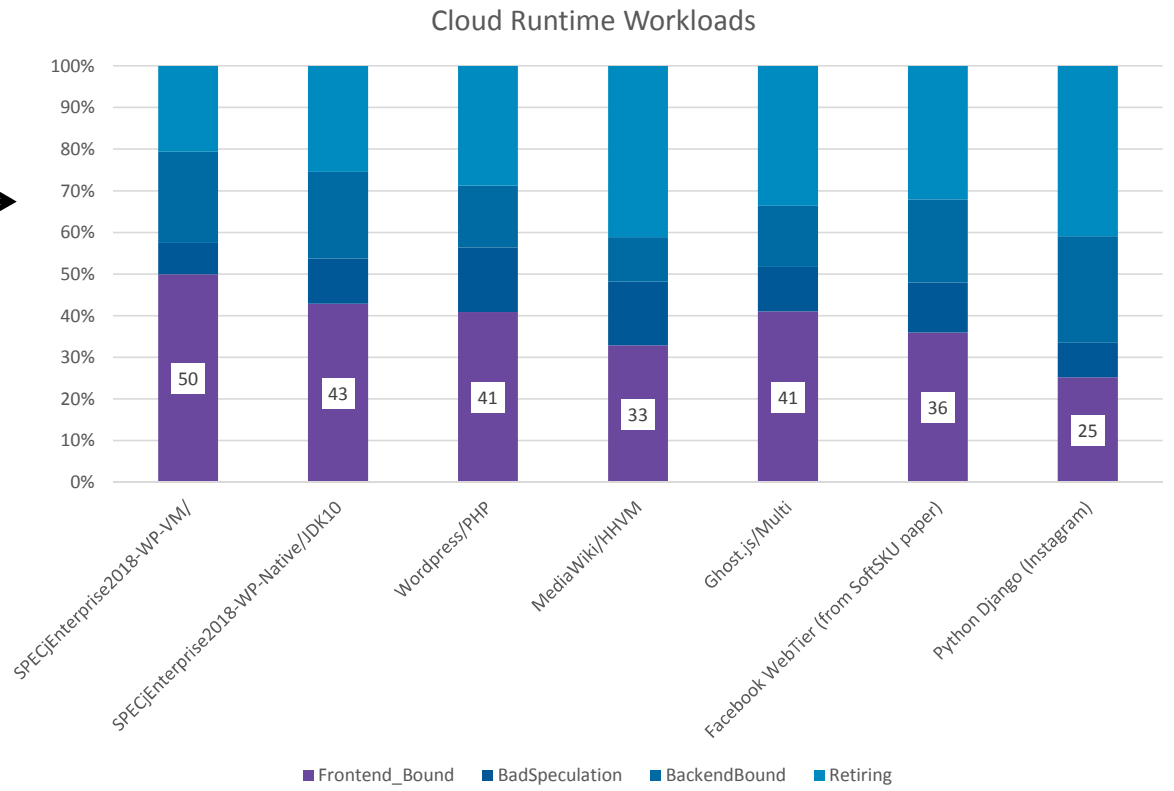
# WebTier: Characteristics & Challenges

## Runtime with High CPU bottlenecks

- Front End Bottlenecks: I-Cache Misses, I-TLB Misses
- Interpreters with high Branch Mispredicts
- Accounting for 50% of Stalls

## Long Pathlength

- 10s of millions of instructions executed per request




*Other names and brands may be claimed as the property of others.

# Cloud Runtimes

## Hardware & Software Optimization

Title: Luna on bed of flowers
Author: Suresh Srinivas
License: Creative Commons
Source: @sweetlunatheyellowlab on Instagram

# Cloud Runtime: Hardware

## Hardware Improvements

- 17% Improvement in Retirement
- 2M Code Pages in shared 2nd level structure
- Larger L2 helps with I$ Misses
  - 40% L2 miss reduction
  - 25% Frontend Improvement

| | MediaWiki BDX-D | MediaWiki SKX-D |
|---|---|---|
| Topdown Frontend bound (%) | 43 | 32 |
| Topdown Retiring (%) | 35 | 41 |
| metric_L2 misses per txn (includes code+data+rfo w/ prefetches) | 1,139,373 | 674,635 |
| RPS | 1.0 | 1.40 |
| Cores | 16 | 18 |
| Package Power | 70W | 85W |

*Other names and brands may be claimed as the property of others.

# Cloud Runtime: Software

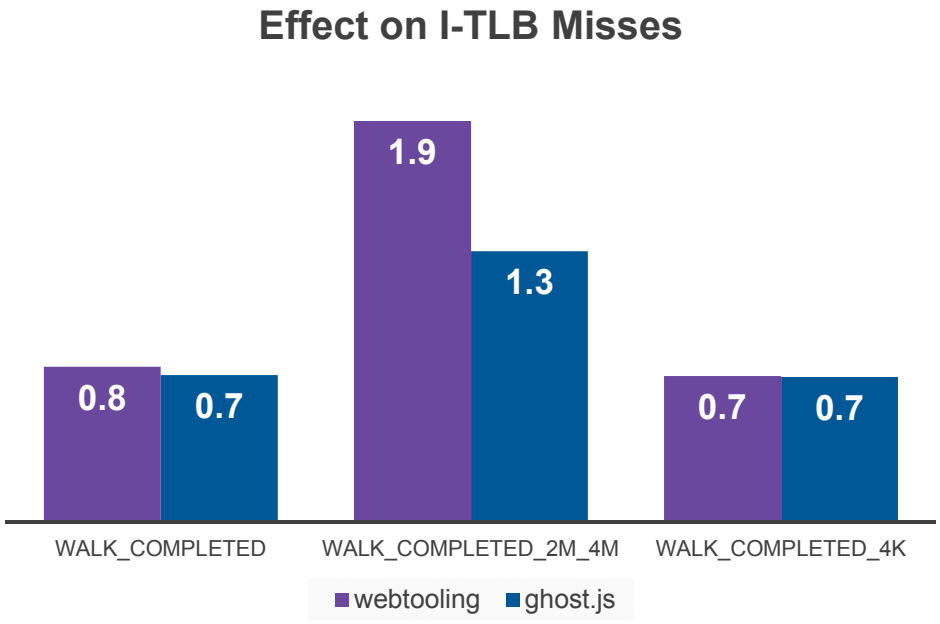
## Reduce I-TLB misses and cycles by using large pages for code

- Remap a subset of .text segment to 2M pages
- Relies on THP through `madvise`
- Reduce 4K walks and Reduce cycles
  - 30% Reduction in 4K Walks

## Benefits longer running workloads

- Ghost.js*: 5%, WebTooling: 3%

**Using 2M Pages reduces I-TLB misses 20-30% & improves perf 3-5%**



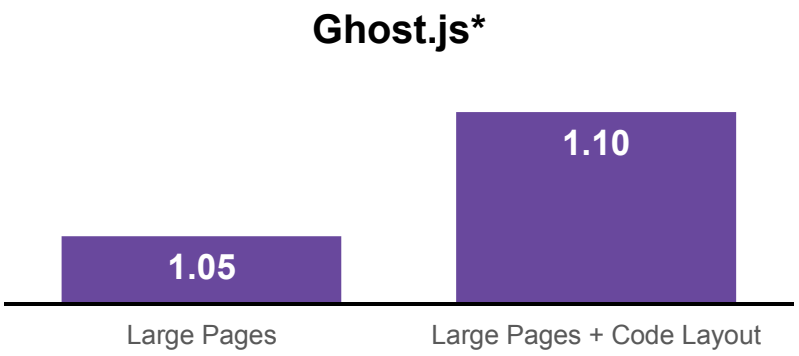

*Other names and brands may be claimed as the property of others.

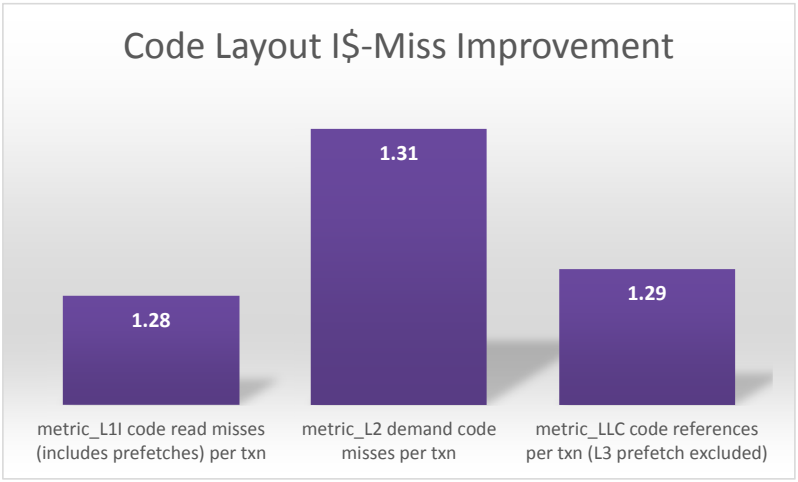
# Cloud Runtime: Software

## Reduce I-Cache Misses through Code Layout Optimization

- Intel CPUs have a feature called last branch records (LBR) where the CPU can continuously log branches to a set of model-specific registers (MSRs).
- Profile guided reordering of the static code in .text reduces L1-I, L2, LLC Code Misses.
- 5-7% Performance improvement

## Combining Optimization:

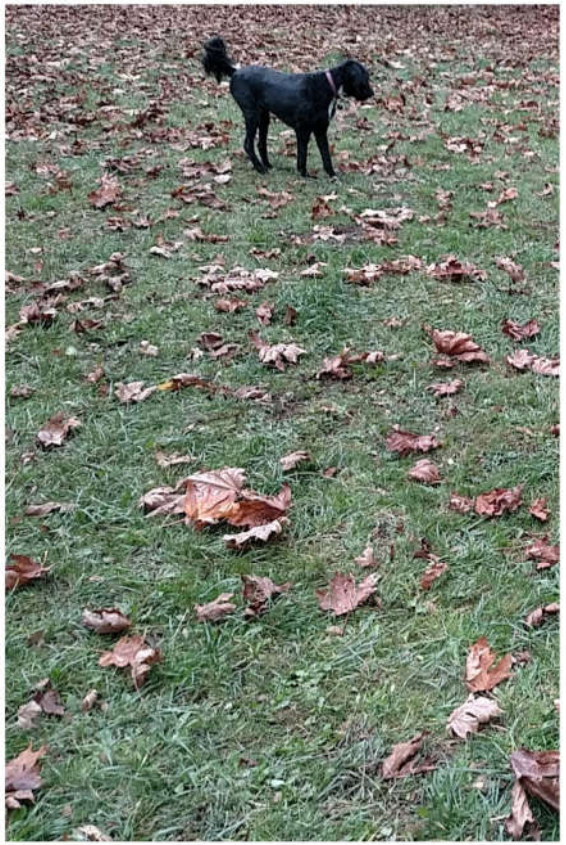


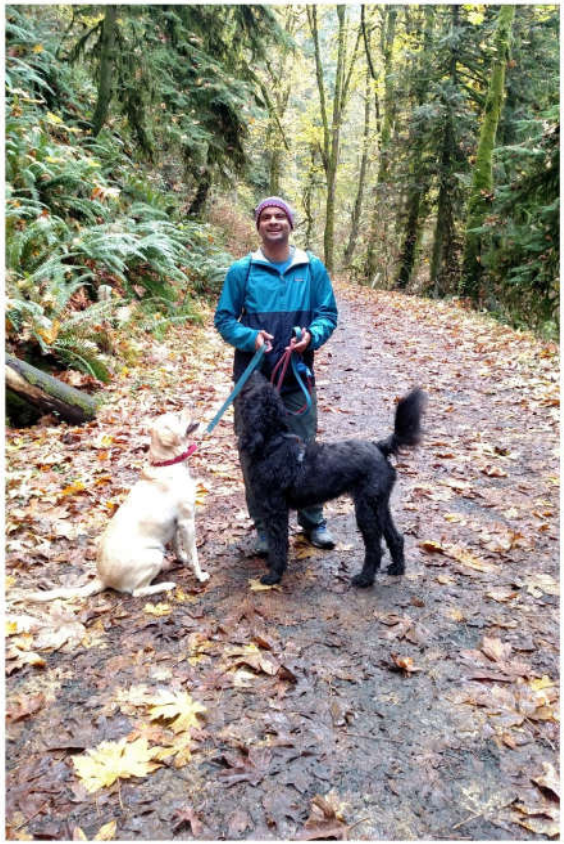




*Other names and brands may be claimed as the property of others.

# Cloud Runtimes

Next Steps,
Open Problems
Call to Action

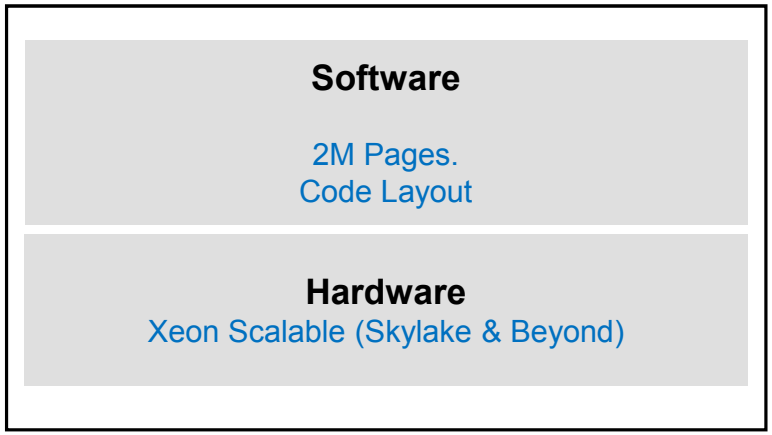




# Next Steps

## Intel Focus:

- Continue to optimize Cloud Runtimes
- Collaborate to address Runtime Bottlenecks
- Enhance our Products
- Develop runtime performance optimization blue print(s)



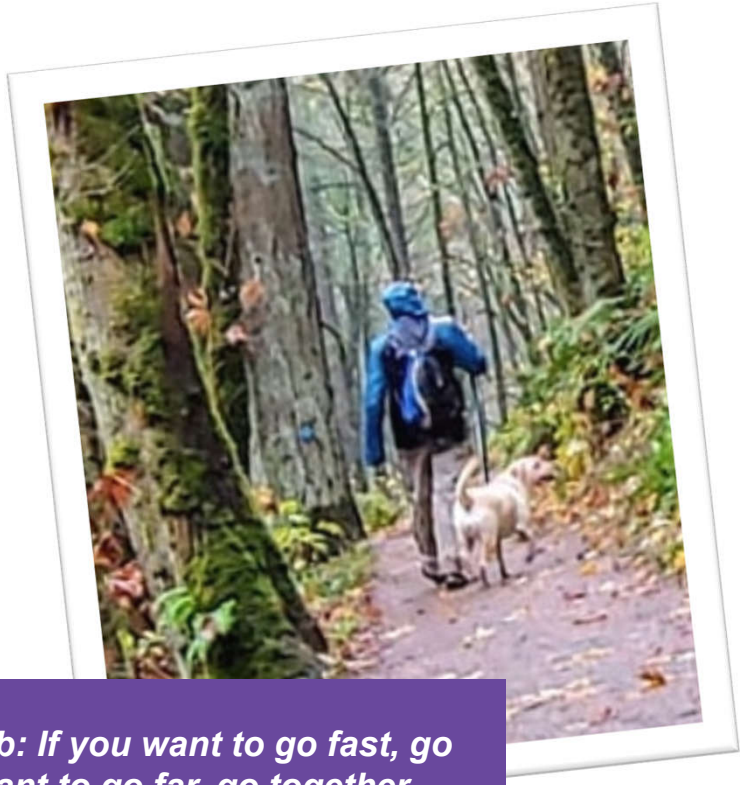
# Open Problems

- Address Frontend problems (I$, I-TLB) in JITs
- Better locality of Native & JIT code
- Better use of SIMD & u-architecture
- Sharing dynamic code across JIT instances
- HW/SW Communication
  - SW Controlled Instruction Fetch
  - SW Control for Indirect Branches

# Call to Action

- Diagnose uarch bottlenecks
- Apply known solutions
- Collaborate on Open Problems

- Contact me suresh.srinivas@intel.com



*African Proverb: If you want to go fast, go alone. If you want to go far, go together.*

Title: Going Far in Forest Park
Author: Suresh Srinivas
License: Creative Commons
Source: @sweetlunatheyellowlab on Instagram

## Collaborate to Solve these Cloud Runtime problems on Intel Architecture

*Other names and brands may be claimed as the property of others.

# Q&A

# Disclaimers

No license (express or implied, by estoppel or otherwise) to any intellectual property rights is granted by this document.

Intel disclaims all express and implied warranties, including without limitation, the implied warranties of merchantability, fitness for a particular purpose, and non-infringement, as well as any warranty arising from course of performance, course of dealing, or usage in trade.

This document contains information on products, services and/or processes in development. All information provided here is subject to change without notice. Contact your Intel representative to obtain the latest forecast, schedule, specifications and roadmaps.

The products and services described may contain defects or errors known as errata which may cause deviations from published specifications. Current characterized errata are available on request. No product or component can be absolutely secure.

Copies of documents which have an order number and are referenced in this document may be obtained by calling 1-800-548-4725 or by visiting www.intel.com/design/literature.htm

Intel, the Intel logo, Intel Atom, Intel Optane, and Intel Xeon are trademarks of Intel Corporation or its subsidiaries in the U.S. and/or other countries.

*Other names and brands may be claimed as the property of others.

© Intel Corporation.

# Cloud Runtime: FaaS

## Evolution

- IaaS->PaaS->FaaS
- FaaS business model vs. IaaS business model

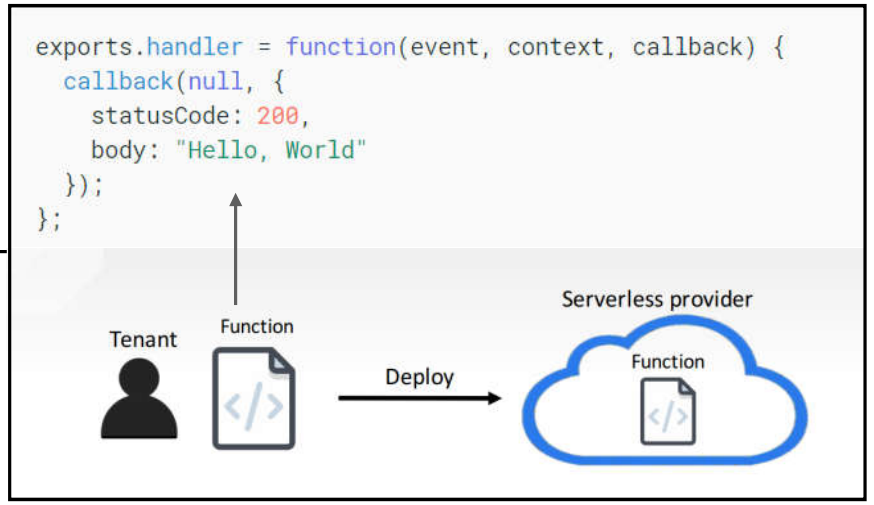
## Pay for Useful Work (Duration of Execution)

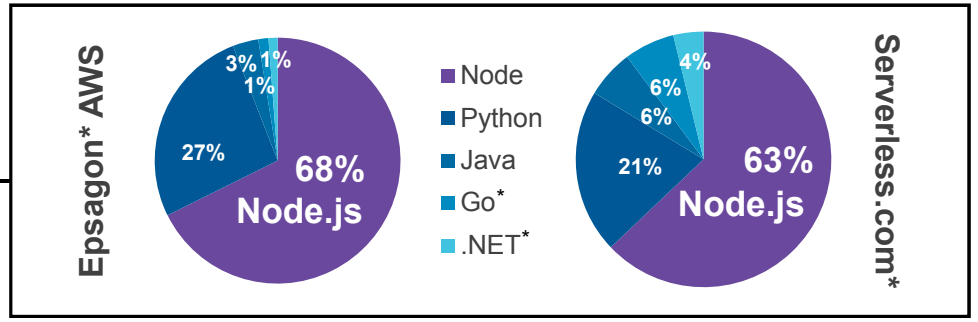
- No effort for server management
- Autoscaling & CSP determines platform for execution

## Function

- From 5 lines to multiple megabytes
- Triggered from an event source

## JavaScript* (with Node.js*) and Python* are leading FaaS languages

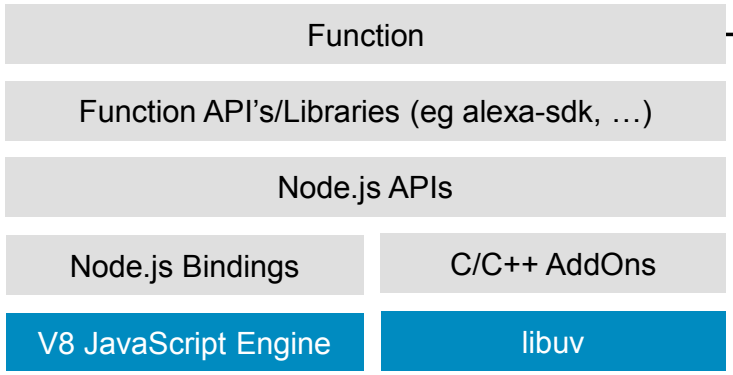



*Other names and brands may be claimed as the property of others.

# JavaScript* on Server

- Node.js* is a JavaScript Server runtime built on top of the Chrome* v8 JavaScript engine
- FaaS Function built with Function APIs/Libraries (eg. Alexa-sdk, others)




**Node.js* Stack**

```
const Alexa = require('alexa-sdk');
const anagrams = require('./anagrams');
var handlers = {
     getAnagrams: function() {… }, …
};

export.handler = function (event, c, l) {
 var alexa = Alexa.handler(event, c, l);
 alexa.registerHandlers(handlers);
 alexa.execute();
}
```

**FaaS Alexa* Anagram Skill**
**https://github.com/evanchiu/alexa-anagram**

*Other names and brands may be claimed as the property of others.

# Common Runtime Challenges

- **Runtimes need high performance primitives**
  - Memory & String Operations (memcpy, memchr, …)
  - Compression
  - IA Encoder/Decoder
  - Several rely on system (glibc: In 2019 people are still using glibc 2.17 from 2014)
- **Runtimes need tooling and common infrastructure**
  - LBR Samples decorated to IR
  - Processor-Trace -> JIT
- **Runtimes Libraries don’t expose optimized native**
  - JavaScript/PHP dont expose IA optimized libs