

TCSS 562: SOFTWARE ENGINEERING FOR CLOUD COMPUTING

Serverless Computing

Wes J. Lloyd
Institute of Technology
University of Washington - Tacoma



OBJECTIVES

- Questions on:
 - Tutorial #2
 - Tutorial #3
- Presentations Schedule
- Presentations Format
- Group Project Check-in – Sunday May 6
- Midterm Wednesday 5/9
- Serverless Computing

May 2, 2018	TCSS562: Software Engineering for Cloud Computing [Spring 2018] Institute of Technology, University of Washington - Tacoma	L10.2
-------------	---	-------

PRESENTATIONS SCHEDULE

▪ WEEK 8

May 14 - Web Architecture

- Team 1 - Amazon Well Architected Framework (Paper)
- Team 2 - AWS Elastic Beanstalk (Technology)

May 16 - Serverless Computing I

- Team 3 - The Serverless Trilemma (Paper)
- Team 4 - Azure Functions
- Team 5 - Code Transformations to AWS Lambda (Paper)

May 2, 2018

TCS562: Software Engineering for Cloud Computing [Spring 2018]
Institute of Technology, University of Washington - Tacoma

L10.3

PRESENTATIONS SCHEDULE - 2

▪ WEEK 9

May 21 - NoSQL DBs

- Team 6 - Choosing the right NoSQL DB (paper)
- Team 7 - DynamoDB

May 23 - Serverless Computing II - Open Source Frameworks

- Team 8 - Open Lambda (paper)
- Team 9 - Oracle Fn
- Team 10 - Apache OpenWhisk

May 2, 2018

TCS562: Software Engineering for Cloud Computing [Spring 2018]
Institute of Technology, University of Washington - Tacoma

L10.4

PRESENTATIONS FORMAT

- Cloud Technology Sharing Presentation:
▪ http://faculty.washington.edu/wlloyd/courses/tcss562/assignments/TCSS562_s2018_A1A.pdf
- Cloud Research Paper Presentation:
▪ http://faculty.washington.edu/wlloyd/courses/tcss562/assignments/TCSS562_s2018_A1B.pdf

May 2, 2018

TCSS562: Software Engineering for Cloud Computing [Spring 2018]
Institute of Technology, University of Washington - Tacoma

L10.5

GROUP PROJECT CHECK IN: MAY 6

- Each group should submit a PDF file to Canvas
- Provides written status-update of group project

Please include on the "Project Checkin":

- 1. Names of the group members
- 2. Project Title/Topic
- 3. Answer the following questions:
 - Q1 - How has the group decided to divide the project work?
What technologies and/or aspects is each group member responsible for?
What aspects of the project have a shared responsibility?

May 2, 2018

TCSS562: Software Engineering for Cloud Computing [Spring 2018]
Institute of Technology, University of Washington - Tacoma

L10.6

GROUP PROJECT CHECK IN: MAY 6 - 2

- Q2 - Describe progress to date:

CODE:

- What code has been developed or located to support the cloud services evaluation?
- What languages are being used?
- What is the size of the code found/developed (lines of code, size KB)?
- Is there a shared GitHub repository? If so, please share a URL.

DATA:

- If the project involves testing a database, what data sources have been found or developed?
- How large are the data sets?
- What type of information do they describe?

May 2, 2018

TCS562: Software Engineering for Cloud Computing [Spring 2018]
Institute of Technology, University of Washington - Tacoma

L10.7

GROUP PROJECT CHECK IN: MAY 6 - 3

PLATFORMS/SERVICES:

- What technologies/services have been tested thus far?
- What initial results do you have?
- Performance data?
- Cost data?
- Q3 - Describe any road blocks or questions you may have.
- The instructor will review road blocks and questions.
- ***It is the team's responsibility to follow-up with the instructor regarding any ongoing road blocks.***

May 2, 2018

TCS562: Software Engineering for Cloud Computing [Spring 2018]
Institute of Technology, University of Washington - Tacoma

L10.8

FEEDBACK

■ ...

May 2, 2018	TCSS562: Software Engineering for Cloud Computing [Spring 2018] Institute of Technology, University of Washington - Tacoma	L10.9
-------------	---	-------

OUTLINE

- Background
- AWS Lambda Demo
- **Serverless Computing: An Investigation of Factors Influencing Microservice Performance**
 - Research Questions
 - Experimental Workloads
 - Experiments/Evaluation
 - Conclusions

May 2, 2018	TCSS562: Software Engineering for Cloud Computing [Spring 2018] Institute of Technology, University of Washington - Tacoma	L10.10
-------------	---	--------

SERVERLESS COMPUTING

Servers

(AAHHHHHHHHH!!)

How should my app withstand a server falling?

How can I tell if a server has been compromised?

How can I increase utilization of my servers?

Which OS should my servers run?

How much remaining capacity do my servers have?

How should I implement dynamic configuration changes on my servers?

When should I decide to scale up my servers?

What size servers are right for my budget?

How will I keep my server OS patched?

How can I control access from my servers?

Which packages should be baked into my server images?

How will the application handle server hardware failure?

How will new code be deployed to my servers?

Which users should have access to my servers?

Should I tune OS settings to optimize my application?

How many users create too much load for my servers?

When should I decide to scale out my servers?

What size server is right for my performance?

How many servers should I budget for?

ADVANTAGES OF SERVERLESS COMPUTING

- Pay for CPU/memory utilization
- High Availability
- Fault Tolerance
- Infrastructure Elasticity
- Function-as-a-Service (FAAS)

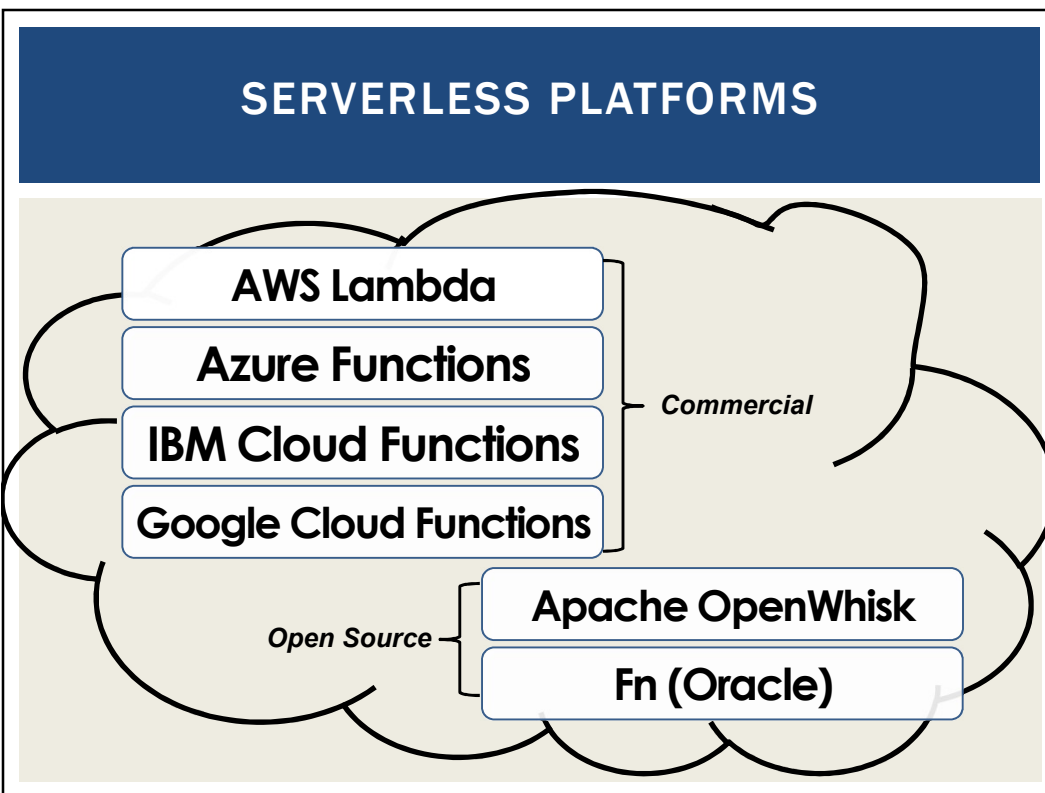
SERVERLESS COMPUTING

Why Serverless Computing?

**Many features of distributed systems,
that are challenging to deliver, are
provided automatically**

...they are built into the platform

May 2, 2018	TCSS562: Software Engineering for Cloud Computing [Spring 2018] Institute of Technology, University of Washington - Tacoma	L10.13
-------------	---	--------



Research Challenges

Serverless Computing

Deploy Applications Without
Fiddling With Servers

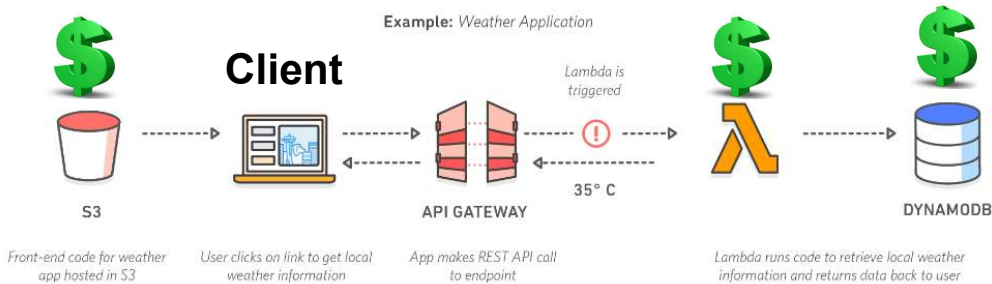


Image from: <https://mobisoftinfotech.com/resources/blog/serverless-computing-deploy-applications-without-fiddling-with-servers/>

15

VENDOR ARCHITECTURAL LOCK-IN

- Serverless software architecture requires external services/components



Images credit: aws.amazon.com

- Increased dependencies → increased hosting costs

May 2, 2018

TCCS562: Software Engineering for Cloud Computing [Spring 2018]
Institute of Technology, University of Washington - Tacoma

L10.16

SERVERLESS PRICING MODEL

- **EXAMPLE:** AWS Lambda Pricing
- **FREE TIER:** first 1,000,000 function calls/month → FREE
first 400,000 GB-sec/month → FREE
- **Obfuscated pricing:**
 - \$0.0000002 per request
 - \$0.000000208 to rent 128MB / 100-ms

May 2, 2018

TCSS562: Software Engineering for Cloud Computing [Spring 2018]
Institute of Technology, University of Washington - Tacoma

L10.17

WEBSERVICE HOSTING EXAMPLE

- Each service call: 100% of 1 CPU-core
100% of 4GB of memory
- Workload: 2 continuous client threads
- Duration: 1 month (30 days)

- VM: Amazon EC2 m4.large 2-vCPU VM
- Hosting cost: \$72/month
m4.large: 10¢/hour, 24 hrs/day x 30 days

- **How much would hosting this workload cost on AWS Lambda?**

May 2, 2018

TCSS562: Software Engineering for Cloud Computing [Spring 2018]
Institute of Technology, University of Washington - Tacoma

L10.18

PRICING OBFUSCATION

▪ Workload:	20,736,000 GB-sec
▪ FREE:	- 400,000 GB-sec
▪ C	
▪ M	Worst-case scenario = ~4.7x !
▪ I	AWS EC2: \$72.00
▪ F	
▪ C	AWS Lambda: \$339.23
▪ Calls:	<u>\$0.84</u>
▪ Total:	<u>\$339.23</u>


May 2, 2018	TCS562: Software Engineering for Cloud Computing [Spring 2018] Institute of Technology, University of Washington - Tacoma	L10.19
-------------	--	--------

PERFORMANCE IMPLICATIONS OF SERVERLESS COMPUTING PLATFORMS

- Infrastructure elasticity
- Load balancing
- Provisioning variation
- Infrastructure retention: COLD vs. WARM
- Memory reservation

May 2, 2018	TCS562: Software Engineering for Cloud Computing [Spring 2018] Institute of Technology, University of Washington - Tacoma	L10.20
-------------	--	--------


SERVERLESS COMPUTING MEMORY RESERVATION QUESTION...



- Lambda memory reserved for function:
- UI provides “slider bar” to set function’s memory allocation
- CPU power coupled to slider bar:
“every doubling of memory, doubles CPU...”
- But how much memory does code require?

▼ Basic settings

Memory (MB) Info
Your function is allocated CPU proportional to the memory configured.



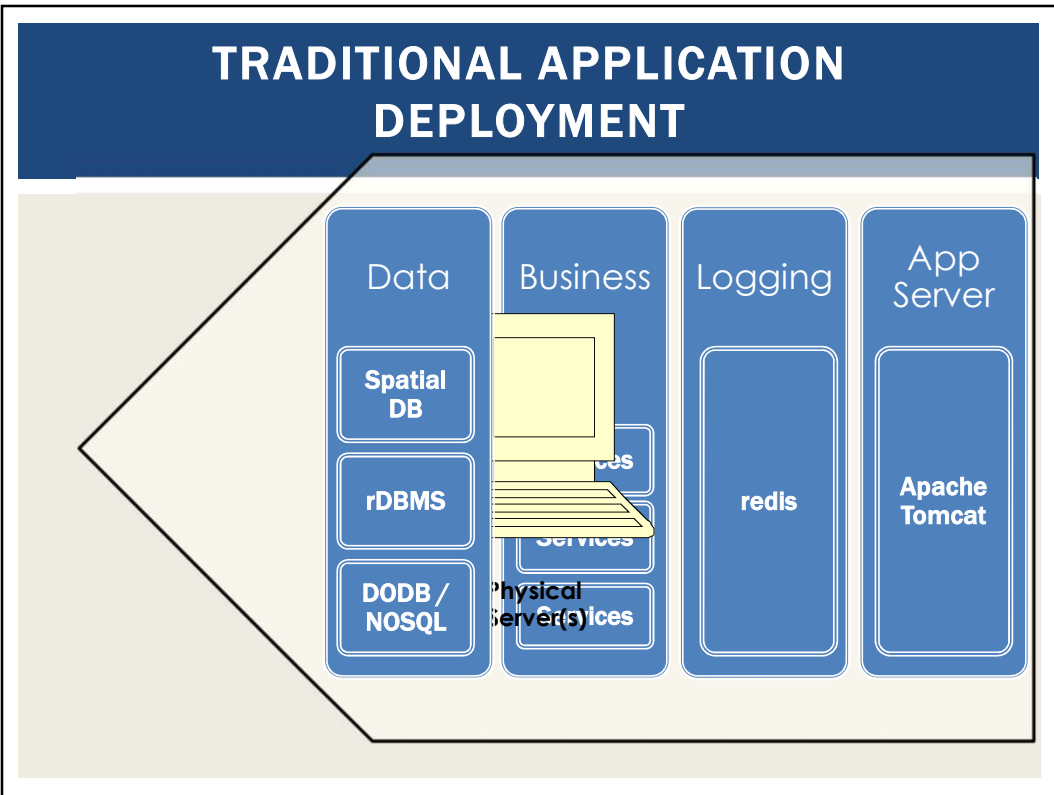
1536 MB

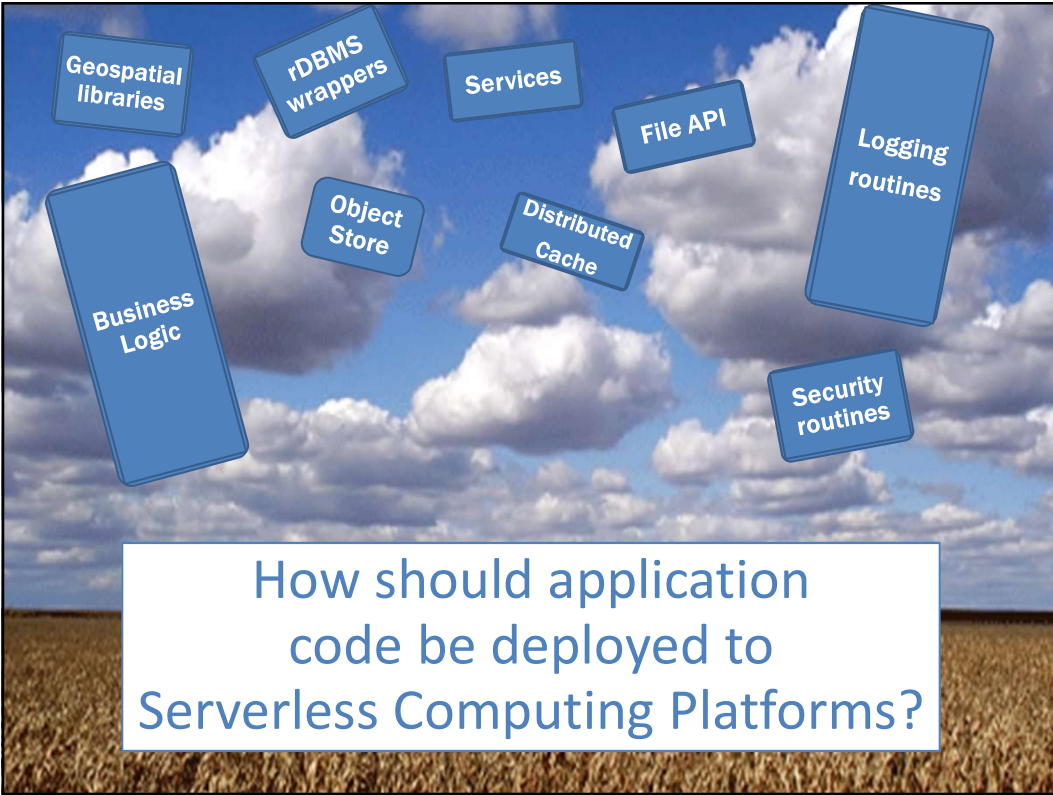
Timeout Info
3 min 0 sec

Description

Performance


May 2, 2018	TCSS562: Software Engineering for Cloud Computing [Spring 2018] Institute of Technology, University of Washington - Tacoma	L10.21
-------------	---	--------





CODE DISAGGREGATION


- How should legacy application code be decomposed into microservices?
- Lambda function limits:
 - All source files and libraries must fit into:
 - AWS Lambda: 64MB compressed, 256MB uncompressed
- What are the cost implications based on how we disaggregate code into individual functions?
- How does this impact # of invocations and memory utilization?



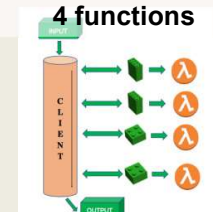
May 2, 2018	TCSS562: Software Engineering for Cloud Computing [Spring 2018] Institute of Technology, University of Washington - Tacoma	L10.24
-------------	---	--------

SERVICE COMPOSITION

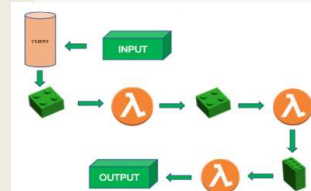
Monolithic




**Client flow control,
4 functions**



**Server flow control,
3 functions**




- Recommended practice:
Decompose code into many microservices
- Platform limits: code + libraries ~256MB
- How does composition impact number of invocations, and memory utilization?



Performance

FREEZE/THAW CYCLE

- Unused infrastructure is deprecated
 - *But after how long?*
- Infrastructure: VMs, “containers”
- Provider-COLD / VM-COLD
 - “Container” images - built/transferred to VMs
- Container-COLD
 - Image cached on VM
- Container-WARM
 - “Container” running on VM



Performance




Image from: Denver7 – The Denver Channel News

SERVERLESS COMPUTING RESEARCH CHALLENGES

- Vendor architectural lock-in
- Pricing obfuscation
- Memory reservation
- Service composition
- Infrastructure freeze/thaw cycle

May 2, 2018

TCSS562: Software Engineering for Cloud Computing [Spring 2018]
Institute of Technology, University of Washington - Tacoma

L10.27

OUTLINE

- Background
- **AWS Lambda Demo**
- **Serverless Computing: An Investigation of Factors Influencing Microservice Performance**
 - Research Questions
 - Experimental Workloads
 - Experiments/Evaluation
 - Conclusions

May 2, 2018

TCSS562: Software Engineering for Cloud Computing [Spring 2018]
Institute of Technology, University of Washington - Tacoma

L10.28

USING AWS LAMBDA

- **Supports many popular languages**
 - Node.js, Java, Python, C#
 - Can include libraries (native & custom)
- **Simple resource model**
 - Memory reservation to 3GB
 - CPU scaled accordingly
- **Flexible use**
 - Synchronous or asynchronous
 - Integration with other AWS services
- **Flexible authorization**
 - Access to resources, VPCs
 - Fine-grained access control

Based on AWS Lambda slide set – slides removed for copyright

May 2, 2018	TCS562: Software Engineering for Cloud Computing [Spring 2018] Institute of Technology, University of Washington - Tacoma	L10.29
-------------	--	--------

USING AWS LAMBDA - 2

- **Authoring functions**
 - Edit code in GUI, or upload code (jar, zip)
 - Third-party plugin support: Eclipse, Visual Studio
- **Monitoring and logging**
 - Amazon cloud watch logs
 - Metrics for requests, errors, throttles
- **Programming model**
 - Language specific: processes, threads, sockets
 - Access to 500MB /tmp space
- **Stateless**
 - Persist data using external storage
 - No affinity or access to underlying infrastructure

Based on AWS Lambda slide set – slides removed for copyright

May 2, 2018	TCS562: Software Engineering for Cloud Computing [Spring 2018] Institute of Technology, University of Washington - Tacoma	L10.30
-------------	--	--------

AWS LAMBDA TRIGGERS

- **Data stores**
 - S3
 - Dynamo DB
 - Kinesis
 - Cognito
- **Configuration repositories**
 - AWS Cloud Formation
 - AWS Cloud Front
 - AWS Code Commit
 - AWS CloudWatch
- **End points**
 - API Gateway
 - AWS IoT
 - AWS Step Functions
 - Amazon Alexa
- **Event/message services**
 - Simple email service (SES)
 - Simple notification service (SNS)
 - Cron Events

Based on AWS Lambda slide set – slides removed for copyright

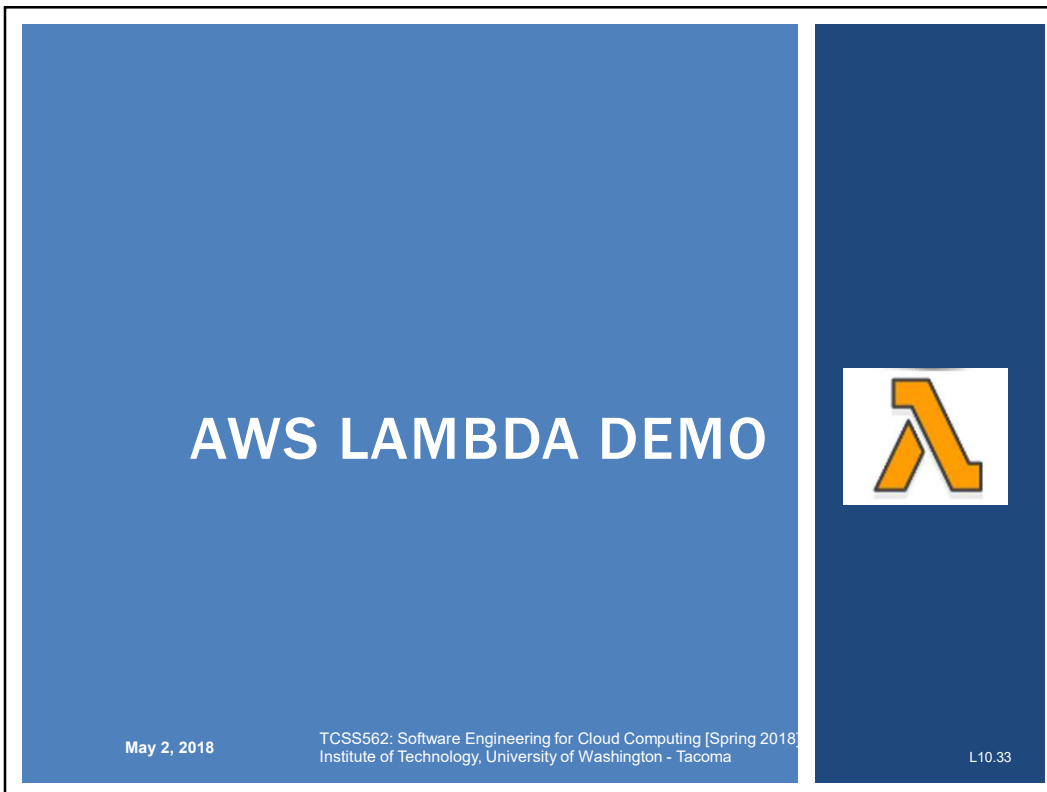
May 2, 2018	TCSS562: Software Engineering for Cloud Computing [Spring 2018] Institute of Technology, University of Washington - Tacoma	L10.31
-------------	---	--------

AWS LAMBDA USE CASES


- **Common/suggested use cases**
 - Web applications
 - Backends
 - Data processing
 - Chatbots
 - Amazon Alexa
 - IT Automation

Based on AWS Lambda slide set – slides removed for copyright

May 2, 2018	TCSS562: Software Engineering for Cloud Computing [Spring 2018] Institute of Technology, University of Washington - Tacoma	L10.32
-------------	---	--------



AWS LAMBDA DEMO



May 2, 2018

TCSS562: Software Engineering for Cloud Computing [Spring 2018]
Institute of Technology, University of Washington - Tacoma

L10.33



**SERVERLESS COMPUTING:
AN INVESTIGATION OF FACTORS
INFLUENCING MICROSERVICE
PERFORMANCE**



**Wes Lloyd, Shruti Ramesh,
Swetha Chinthalapati,
Lan Ly, Shrideep Pallickara**

April 20, 2018

Institute of Technology,
University of Washington, Tacoma, Washington USA
**IC2E 2018: IEEE International Conference
on Cloud Engineering**

OUTLINE

- Background
- AWS Lambda Demo
- **Serverless Computing: An Investigation of Factors Influencing Microservice Performance**
 - **Research Questions**
 - Experimental Workloads
 - Experiments/Evaluation
 - Conclusions

May 2, 2018

TCSS562: Software Engineering for Cloud Computing [Spring 2018]
Institute of Technology, University of Washington - Tacoma

L10.35

RESEARCH QUESTIONS

RQ1: What are the performance implications of infrastructure *elasticity* for serverless computing?
(e.g. *COLD* vs. *WARM* performance)

RQ2: How does *load balancing* vary in serverless computing? How do computational requests impact load balancing, and ultimately performance?

May 2, 2018

TCSS562: Software Engineering for Cloud Computing [Spring 2018]
Institute of Technology, University of Washington - Tacoma

L10.36

RESEARCH QUESTIONS - 2

- RQ3:** What performance implications result from provisioning variation of container infrastructure?
- RQ4:** What are the impacts on infrastructure retention based on microservice/function utilization?
- RQ5:** What performance implications result from microservice memory reservation size? How does memory reservation size impact container placement?

May 2, 2018

TCSS562: Software Engineering for Cloud Computing [Spring 2018]
Institute of Technology, University of Washington - Tacoma

L10.37

OUTLINE

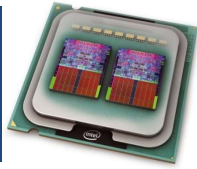
- Background
- AWS Lambda Demo
- **Serverless Computing: An Investigation of Factors Influencing Microservice Performance**
 - Research Questions
 - Experimental Workloads
 - Experiments/Evaluation
 - Conclusions

May 2, 2018



TCSS562: Software Engineering for Cloud Computing [Spring 2018]
Institute of Technology, University of Washington - Tacoma

L10.38

AWS LAMBDA COMPUTE BOUND TEST SERVICE

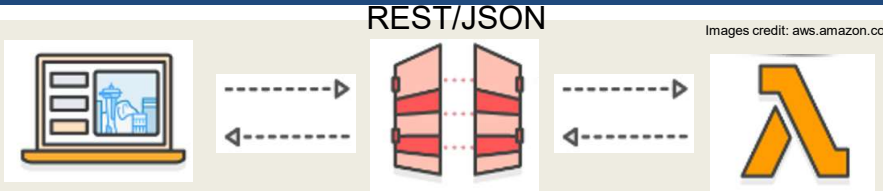


- Increasing stress levels 1 (none) → 9 (high) (*non-linear*)
- Parameters:
 - Operand array size and number of calculation loops (0, 20, 100, 1,000, 10,000, 25,000, 100,000)
 - Operands stored in random array locations
 - Induces page faults when seeking random locations
 - Number of function calls per loop (0, 20, 1,000, 100,000, 300,000)
- Control CPU time of function as input parameter
- Goal: observe impact of CPU time on infrastructure scaling, provisioning variation, retention, and service performance


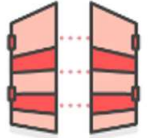




AWS LAMBDA TESTING

REST/JSON



Images credit: aws.amazon.com

 <p>Client: c4.2xlarge</p> <p>BASH: GNU Parallel Multi-thread client "partest"</p> <p>Up to 100 concurrent synchronous requests</p> <p>Results of each thread traced individually</p>	 <p>API GATEWAY</p> <p>Fixed-availability zone: EC2 client / Lambda server us-east-1e</p>	 <p>CPU-bound Test Function</p> <p>Max service duration: < 30 seconds</p> <p>Memory: 128 to 1536MB</p>
---	--	---

May 2, 2018	TCCS562: Software Engineering for Cloud Computing [Spring 2018] Institute of Technology, University of Washington - Tacoma	L10.40
-------------	---	--------

AWS LAMBDA TESTING

The diagram illustrates the AWS Lambda testing architecture. On the left, a laptop icon represents the **Client: c4.2xlarge**. A dashed arrow points from the client to the **API GATEWAY**, which is represented by a server rack icon. Another dashed arrow points from the API Gateway to the **CPU-bound Test Function**, represented by the AWS Lambda logo. The text **REST/JSON** is positioned above the API Gateway. A small note **Images credit: aws.amazon.com** is located above the Lambda logo.

Automatic Metrics Collection:

- New vs. Recycled Containers/VMs
 - # of requests per container/VM
- Avg. performance per container/VM
 - Avg. performance workload
 - Standard deviation of requests per container/VM

Container Identification
UUID → /tmp file

VM Identification
btime → /proc/stat

Linux CPU metrics

AZURE FUNCTIONS TESTING

- Http-triggered function app, written in C#
- Logs to Azure Table storage (*similar to Dynamo DB*)
 - Unique app service instance IDs
 - Current worker process ID
- Consumption plan → auto-scaled infrastructure
 - vs. app service plan (*deployment to dedicated VMs*)
- Performance testing:
Visual Studio Team System (VSTS)

May 2, 2018	TCSS562: Software Engineering for Cloud Computing [Spring 2018] Institute of Technology, University of Washington - Tacoma	L10.42
-------------	---	--------

OUTLINE

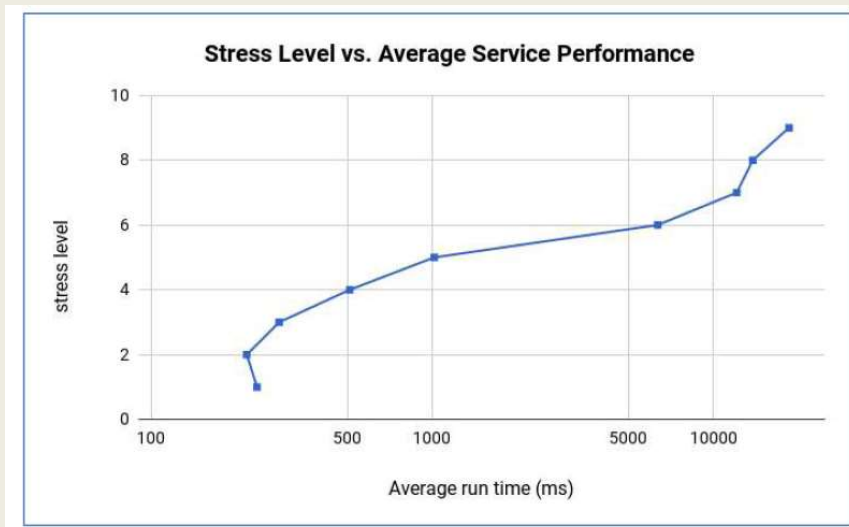
- Background
- AWS Lambda Demo
- **Serverless Computing: An Investigation of Factors Influencing Microservice Performance**
 - Research Questions
 - Experimental Workloads
 - Experiments/Evaluation
 - Conclusions

May 2, 2018

TCSS562: Software Engineering for Cloud Computing [Spring 2018]
Institute of Technology, University of Washington - Tacoma

L10.43

CPU-BOUND LAMBDA TEST SERVICE WARM PERFORMANCE



RQ-1: ELASTICITY

- What are the performance implications of infrastructure elasticity for serverless computing?

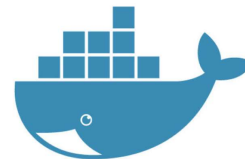
(e.g. *COLD* vs. *WARM* performance)

May 2, 2018

TCS562: Software Engineering for Cloud Computing [Spring 2018]
Institute of Technology, University of Washington - Tacoma

L10.45

RQ-1: AWS LAMBDA LATENCY EVALUATION



- AWS Lambda Simulation
- Harness c4.8xlarge 36 vCPU VM instance
 - Intel Xeon E5-2666v3 CPU - *same as Lambda*
- Lambda JAR file deployed Docker container(s)
 - Set memory: `docker run "-m <ram in MB>"`
 - Set CPUs: `docker run "--cpus <VCPU>"`
- Compare: 1 and 12 concurrent runs
 - Avg VM tenancy ~12.3 of all tests
- How does Lambda scale CPU power?

Literal Estimates:

Memory (MB)	Expected CPU%
128	16.6%
256	33.3%
384	50.0%
512	66.6%
640	83.3%
768	100.0%
896	116.7%
1024	133.3%
1152	150.0%
1280	166.60%
1408	183.30%
1536	200.00%

May 2, 2018

TCS562: Software Engineering for Cloud Computing [Spring 2018]
Institute of Technology, University of Washington - Tacoma

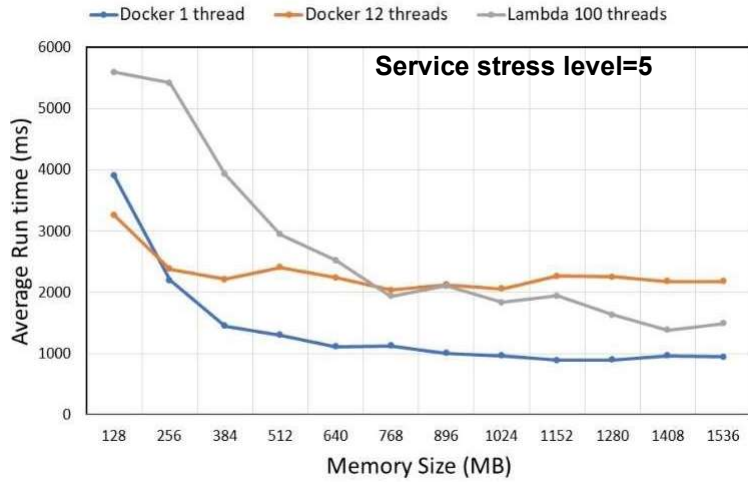
L10.46

RQ-1

Assumed tenancy of ~12 service requests per container for Lambda: *average across many tests*

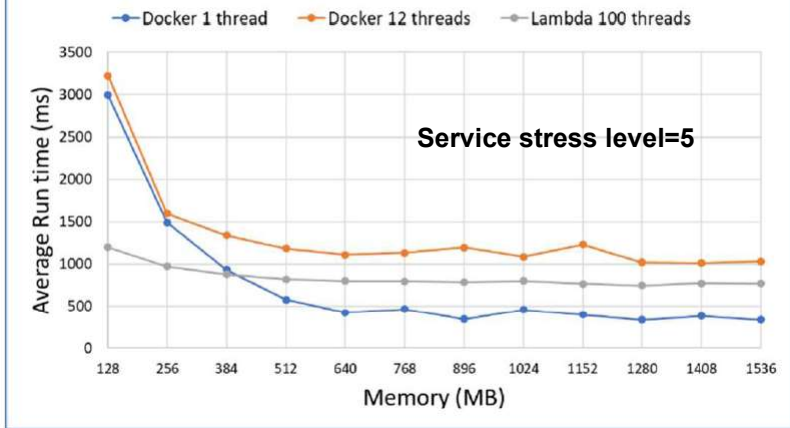
CE

Cold Run Performance - Docker-Machine vs. Lambda



**RQ-1: EC2/DOCKER VS. LAMBDA PERFORMANCE
 INTEL XEON E5-2666 V3 - WARM**

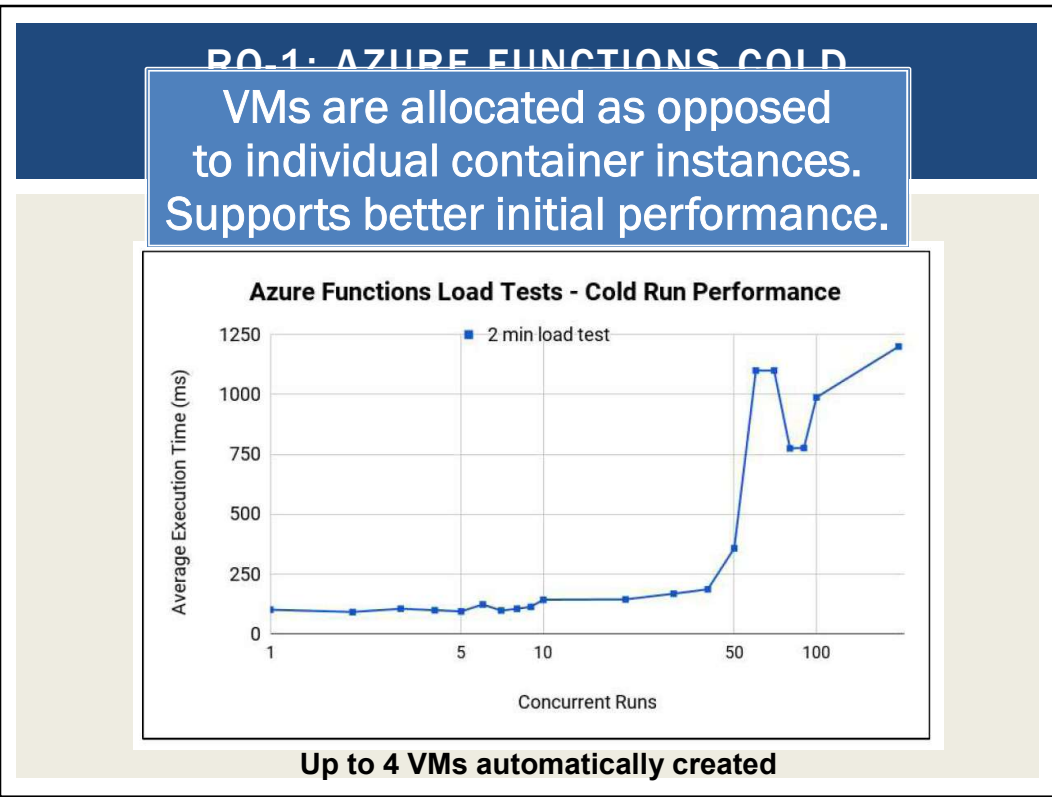
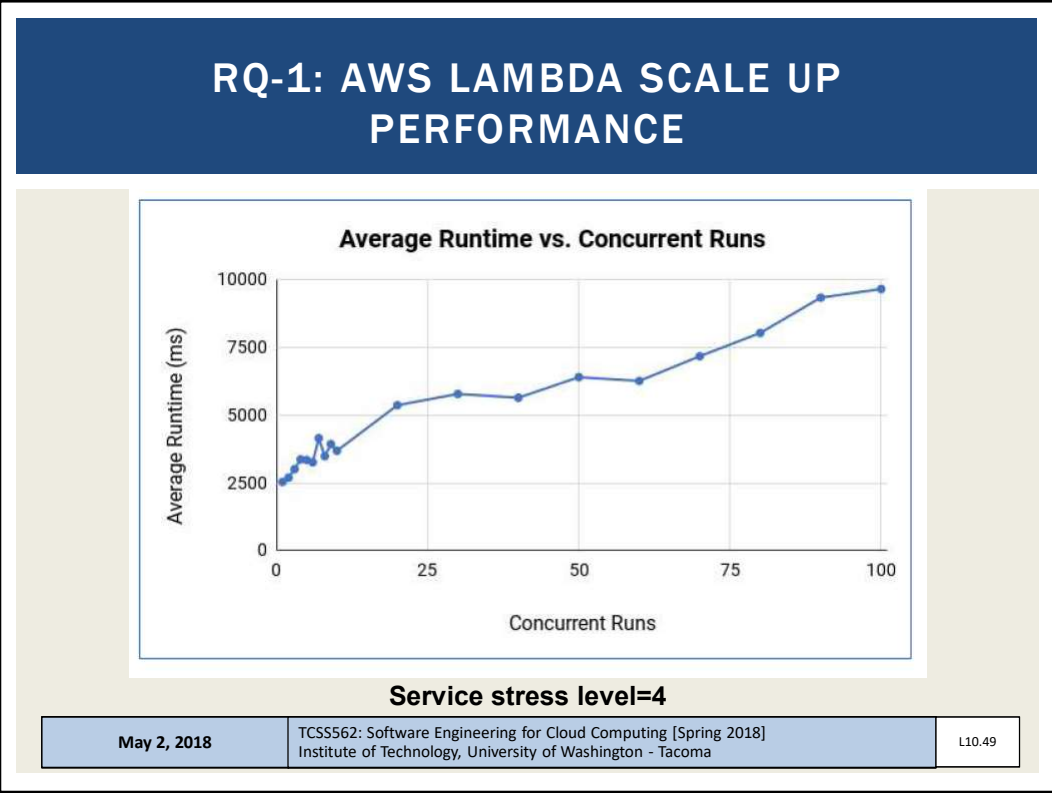
Warm Run Performance - Docker-Machine vs. Lambda



May 2, 2018

TCS562: Software Engineering for Cloud Computing [Spring 2018]
 Institute of Technology, University of Washington - Tacoma

L10.48



RQ-2: LOAD BALANCING

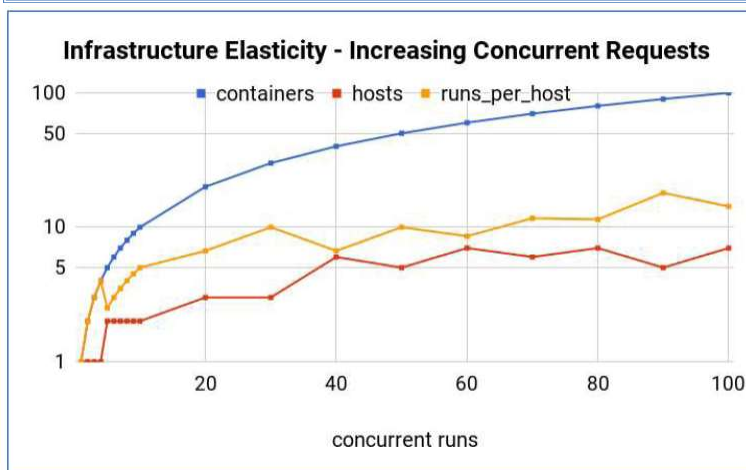
- How does load balancing vary in serverless computing?
- How do computational requests impact load balancing, and ultimately performance?

May 2, 2018

TCSS562: Software Engineering for Cloud Computing [Spring 2018]
Institute of Technology, University of Washington - Tacoma

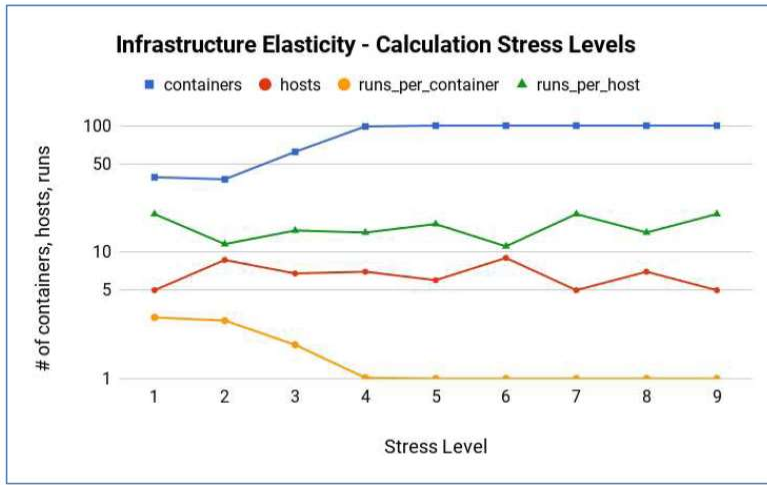
L10.51

COLD service requests receive separate container instances to amortize startup overhead



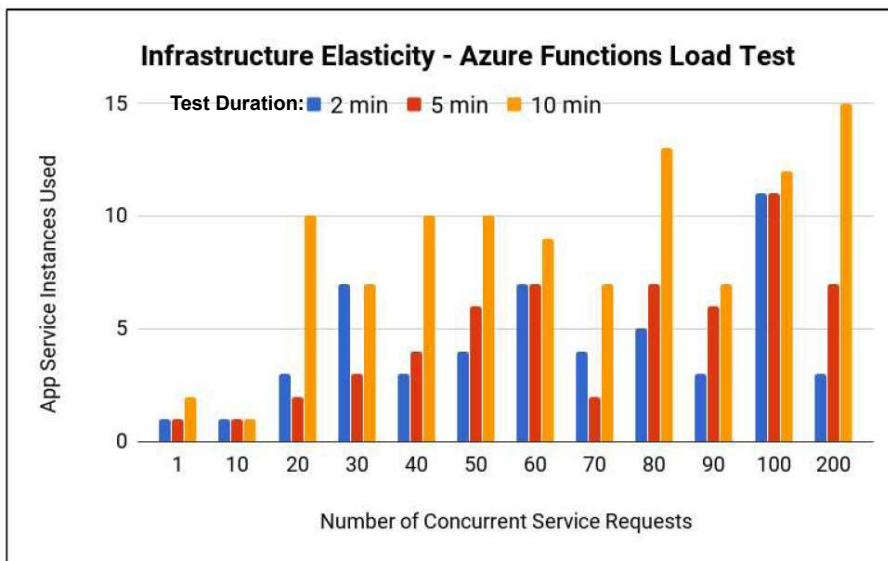
Service stress level=4

WARM service requests share container instances unless CPU requirements are increased



Average for 100 runs

RQ-2: COLD AZURE FUNCTIONS INFRASTRUCTURE FOR SCALING



RQ-3: PROVISIONING VARIATION

- What performance implications result from provisioning variation of container infrastructure?

May 2, 2018

TCSS562: Software Engineering for Cloud Computing [Spring 2018]
Institute of Technology, University of Washington - Tacoma

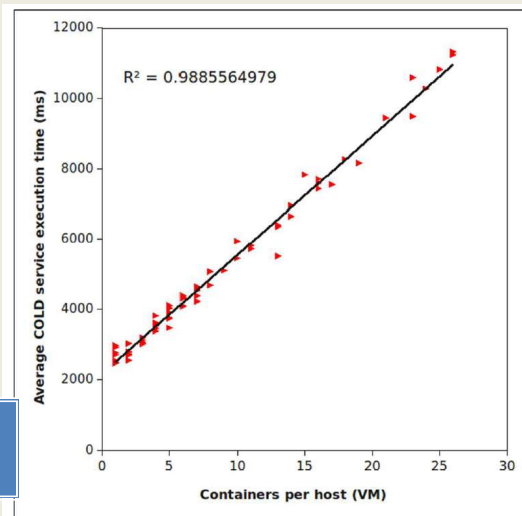
L10.55

RQ-3: COLD LAMBDA SERVICE PERFORMANCE VS. CONTAINER PLACEMENT

When more containers were placed on the same VMs for COLD service requests, Lambda Performance suffered up to 5x !

The impact of tenancy vs. performance is quite clear.

Service stress level=4



RQ-4: INFRASTRUCTURE RETENTION

- What are the impacts on *infrastructure retention* based on microservice/function utilization?

May 2, 2018

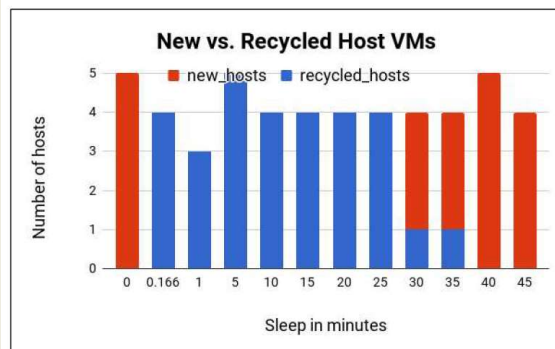
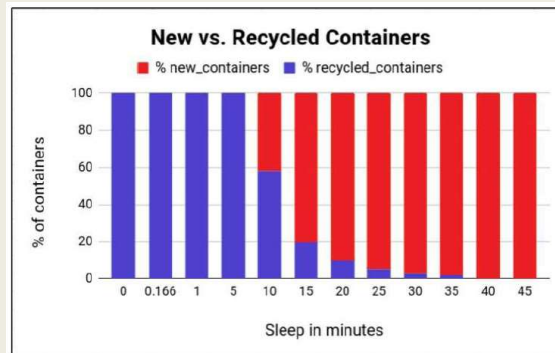
TCSS562: Software Engineering for Cloud Computing [Spring 2018]
Institute of Technology, University of Washington - Tacoma

L10.57

**Lambda
Container
Recycling**

Service stress level=4

**Lambda
Virtual Machine
Recycling**



RQ-5: MEMORY RESERVATION

- What performance implications result from microservice memory reservation size?
- How does memory reservation size impact container placement?

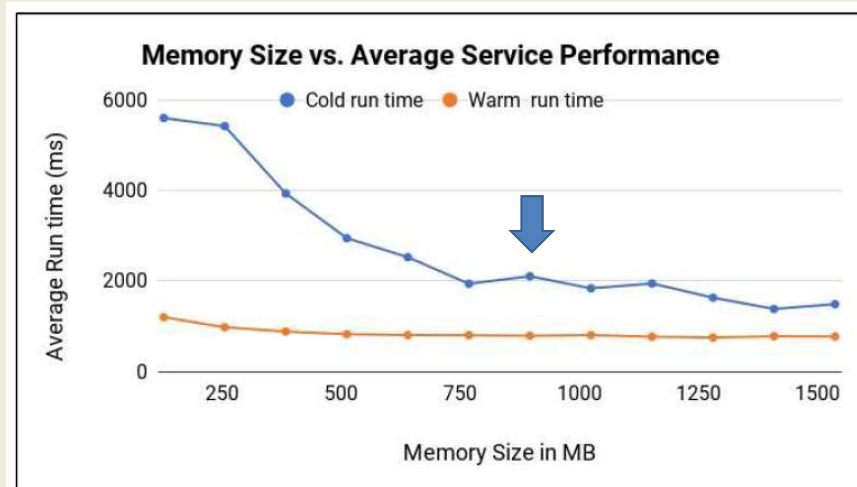
May 2, 2018

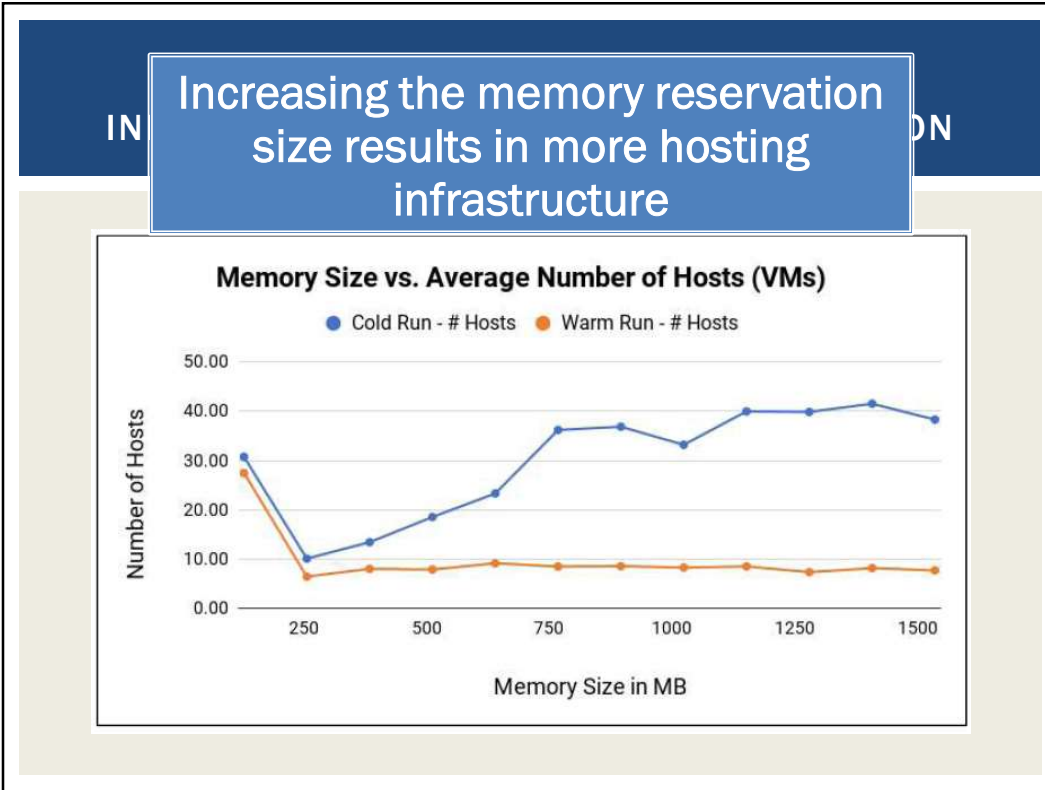
TCSS562: Software Engineering for Cloud Computing [Spring 2018]
Institute of Technology, University of Washington - Tacoma

L10.59

RQ-5: SLIDER BAR TEST: MEMORY VS. CPU POWER

Service stress level=4






OUTLINE

- Background
- AWS Lambda Demo
- **Serverless Computing: An Investigation of Factors Influencing Microservice Performance**
 - Research Questions
 - Experimental Workloads
 - Experiments/Evaluation
 - Conclusions

May 2, 2018	TCSS562: Software Engineering for Cloud Computing [Spring 2018] Institute of Technology, University of Washington - Tacoma	L10.62
-------------	---	--------

CONCLUSIONS




- **RQ-1 Elasticity:** Extra infrastructure is provisioned to compensate for initialization overhead of “container” startup
 - VM COLD: up to ~20x slower than WARM
 - Container COLD: ~5x slower than WARM

- **RQ-2 Load Balancing:** Better when COLD. WARM runs only use all original infrastructure when CPU-bound execution time is similar to container initialization execution time
 - **Must increase stress level to harness available infrastructure**

May 2, 2018	TCS562: Software Engineering for Cloud Computing [Spring 2018] Institute of Technology, University of Washington - Tacoma	L10.63
-------------	--	--------

CONCLUSIONS - 2



- **RQ-3 Provisioning Variation:** Bad placement can lead to ~4.6x degradation in COLD service performance

- **RQ-4 Infrastructure Retention:**
3 distinct performance states:
VM COLD, Container COLD, WARM
 - Containers begin to disappear after 10 minutes
 - VM hosts deprecated after ~40 minutes

- **RQ-5 Memory Reservation:**
 - For non memory-bound service, performance improves up to ~512-640MB

May 2, 2018	TCS562: Software Engineering for Cloud Computing [Spring 2018] Institute of Technology, University of Washington - Tacoma	L10.64
-------------	--	--------

QUESTIONS

May 2, 2018

TCSS562: Software Engineering for Cloud Computing [Spring 2018]
Institute of Technology, University of Washington - Tacoma

L10.65