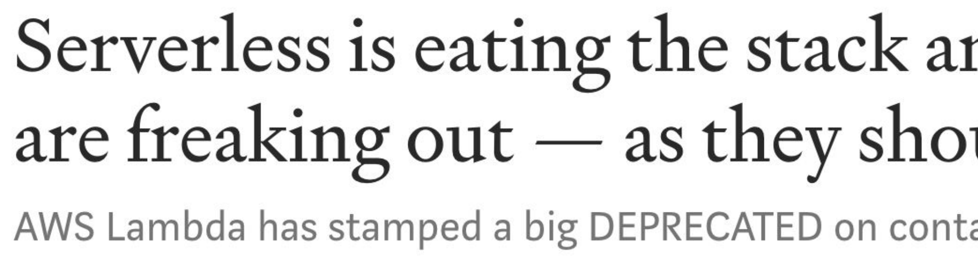
Serverless State

What comes next for serverless

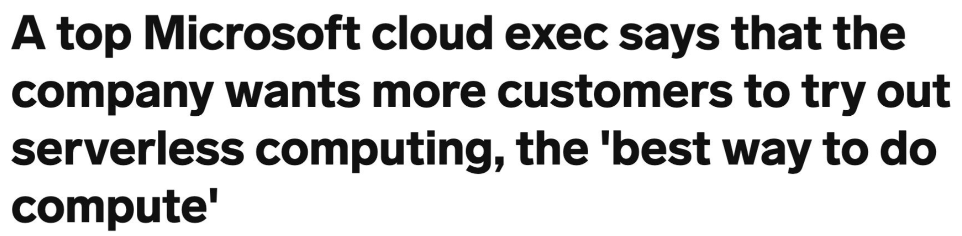
Dr. Tim Wagner

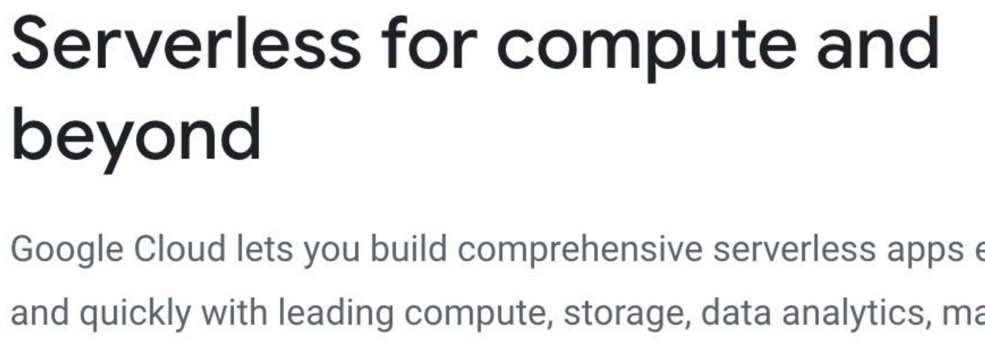
ServerlessConf NY 2019

“You’ll ~~never~~ change how people write applications”


Serverless is eating the stack ar
are freaking out — as they shou

AWS Lambda has stamped a big DEPRECATED on conta


A top Microsoft cloud exec says that the company wants more customers to try out serverless computing, the 'best way to do compute'


Serverless for compute and beyond

Google Cloud lets you build comprehensive serverless apps e
and quickly with leading compute, storage, data analytics, ma

STACKERY

serverless
framework

Gartner names IOpipe Cool Vendor in Performance Analysis

epsagon

“You’d ~~have to~~ build an ecosystem”

“The whole cloud would ~~have to~~ change”

AWS Lambda
Amazon S3 events
Amazon DynamoDB streams
Amazon Kinesis-to-Lambda
Amazon Aurora Serverless
AWS Event Hub
Amazon Serverless Repository
Amazon API Gateway
AWS Step Functions

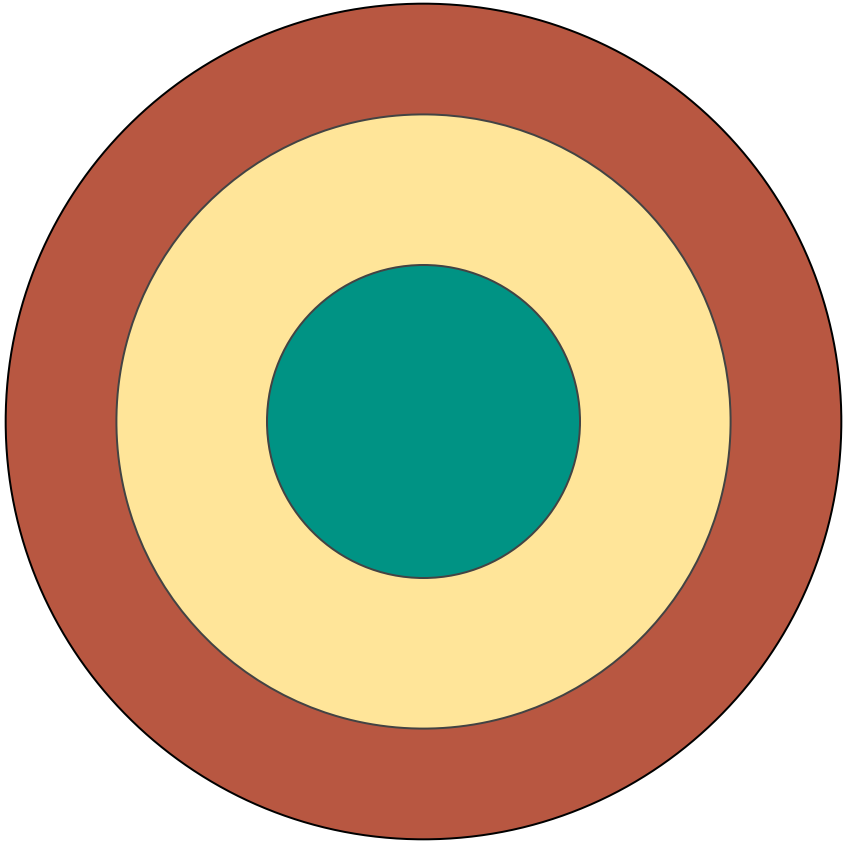
Azure Serverless Functions
Azure Serverless Kubernetes
Azure Logic Apps
Azure API Management
Azure Event Grid
Azure Cloud Events
Azure Service Bus

Google Cloud Functions
Google Cloud Run
Firebase

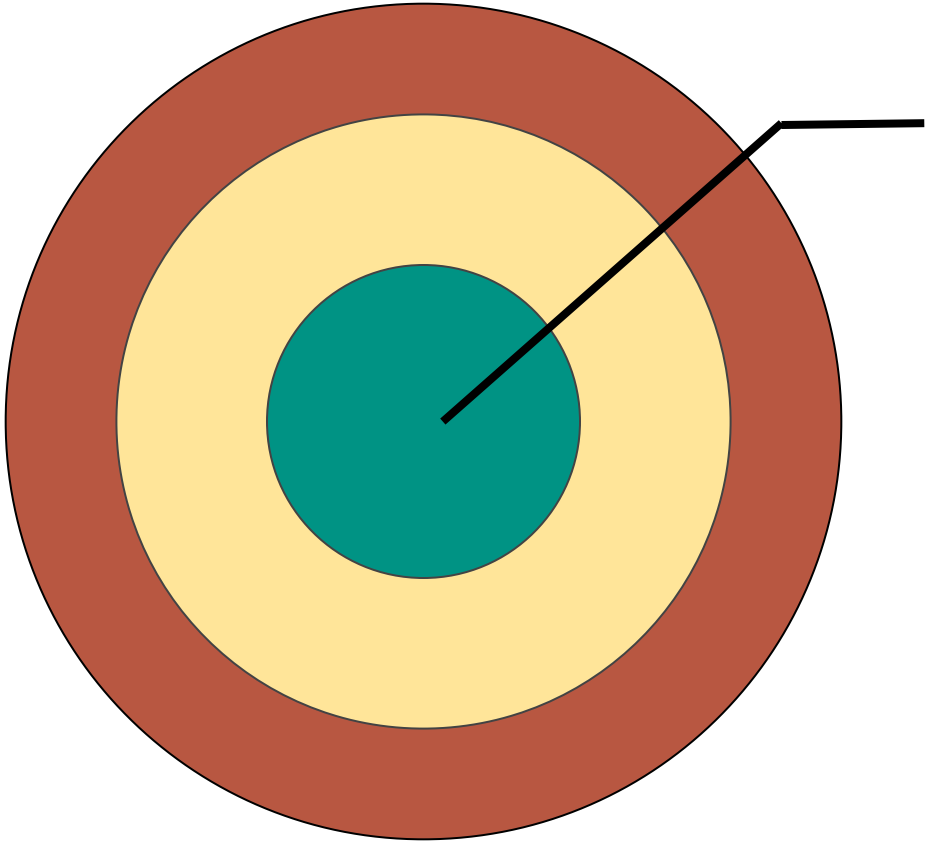
Apache OpenWhisk

We changed how the world writes software.

So, what’s next?



The Serverless Target



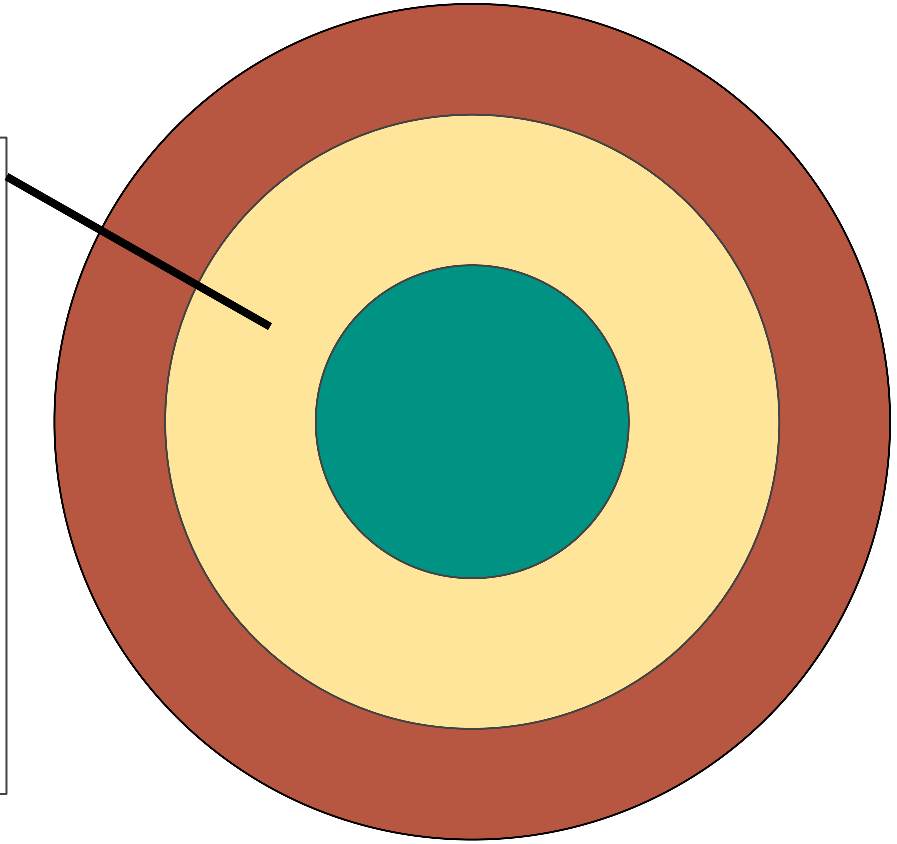
Event handlers

- Sync & async
- Reactive
- Automated scaling
- Integrated into vendor stacks
- **Way easier than what came before**

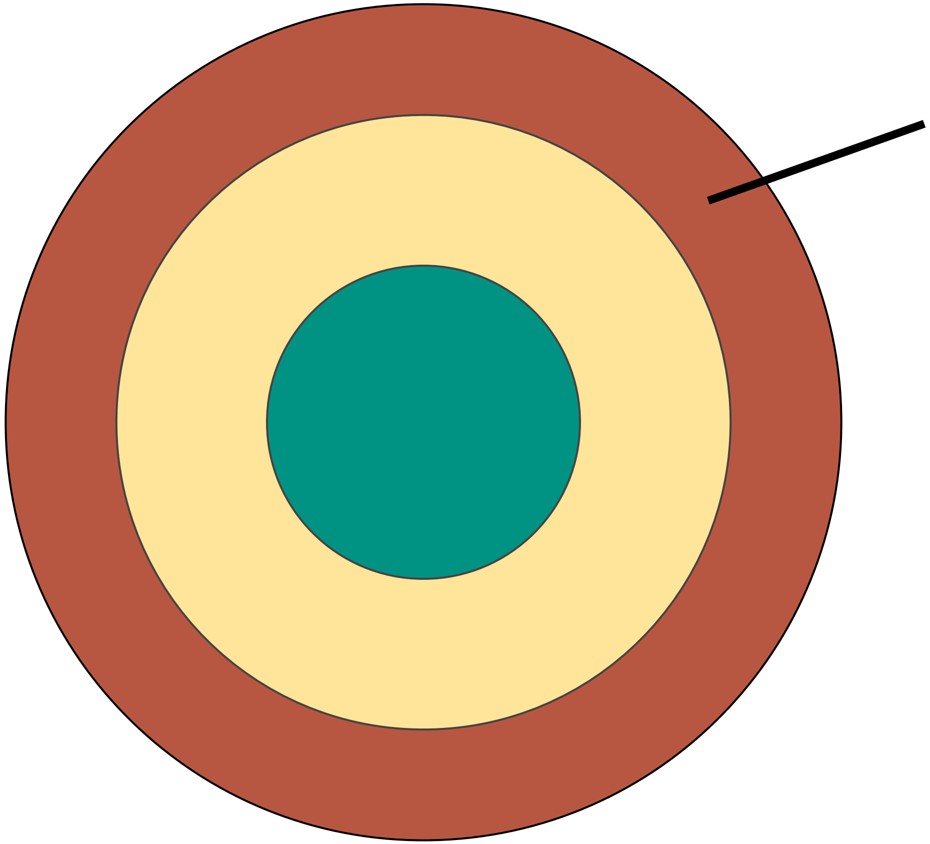
Event handling: **Bullseye!**

Mobile and Web Apps

- API-driven
- Microservice architecture
- Automated scaling
- Easier CRUD
- Legacy migration (sorta)
- Tools and frameworks (some)
- Managed GraphQL (yay)



Mobile and web backends: **Ok, needs more tooling support**



Serverless Supercomputing

- Big data
- Analytics
- Massively scaled simulations
- On-the-fly video transcoding
- Genetic algs
- ML/AI

Big data / distributed computing: **Still academic (pun intended)**

Conversation at AWS HQ, circa 2013...

Adam Selipsky, now CEO of Tableau:

"So Tim, what kind of application would Lambda never be able to handle?"

Tim Wagner, aspiring AWS Lambda inventor:

"Video transcoding. It will never be any good for video transcoding."

Encoding, Fast and Slow: Low-Latency Video Processing Using Thousands of Tiny Threads

Sadjad Fouladi, Riad S. Wahby, and Brennan Shacklett, *Stanford University;*
Karthikeyan Vasuki Balasubramaniam, *University of California, San Diego;*
William Zeng, *Stanford University;* Rahul Bhalerao, *University of California, San Diego;*
Anirudh Sivaraman, *Massachusetts Institute of Technology;*
George Porter, *University of California, San Diego;* Keith Winstein, *Stanford University*

https://www.usenix.org/conference/nsdi17/technical-sessions/presentation/fouladi

This paper is included in the Proceedings of the 14th USENIX Symposium on Networked Systems Design and Implementation (NSDI ’17).

March 27–29, 2017 • Boston, MA, USA

ISBN 978-1-931971-37-9

If the world's fastest
video transcoder
is serverless...

*...what else are we
missing out on???*

Occupy the Cloud: Distributed Co

Eric Jonas, Qifan Pu, Shivaram Venkataraman, I
University of California, Be
jonas, qifan, shivaram, istoica, brecht}@

target on-p

Outsourcing Everyday Jobs to Thousands of Cloud Functions with gg

SADJAD FOULADI, FRANCISCO ROMERO, DAN ITER, QIAN LI, ALEX OZDEMIR, SHUVO CHATTERJEE, MATEI ZAHARIA, CHRISTOS KOZYRAKIS, AND KEITH WINSTEIN

Understanding Ephemeral Storage for Serverless Analytics

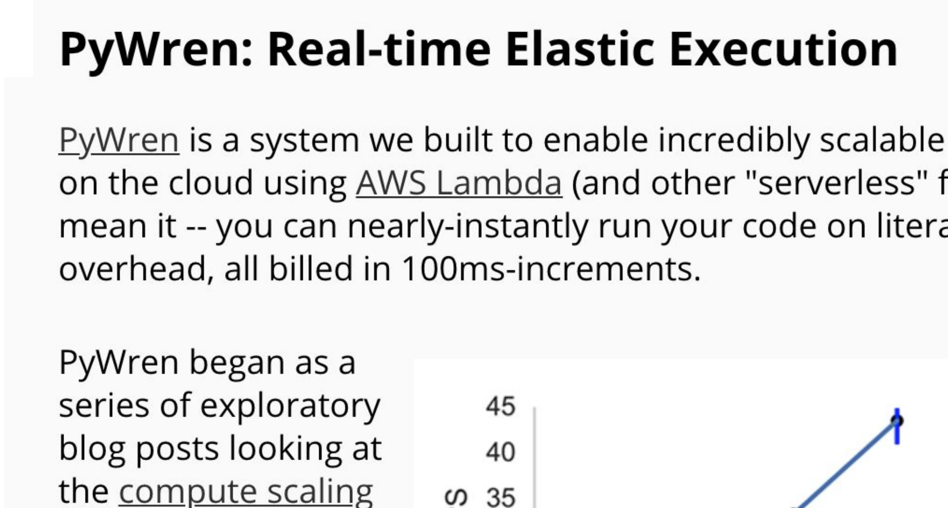
Distributed computi
users, in spite of ma
mercial offerings. W

Pocket: Elastic Ephemeral Sto for Serverless Analytics

Numpywren: Serverless Linear Algebra

Cloud Programming Simplified: A Berkeley View on Serverless Computing

UC Berkeley Paper Heralds Rise of Serverless Computing in the Cloud


PyWren: Real-time Elastic Execution

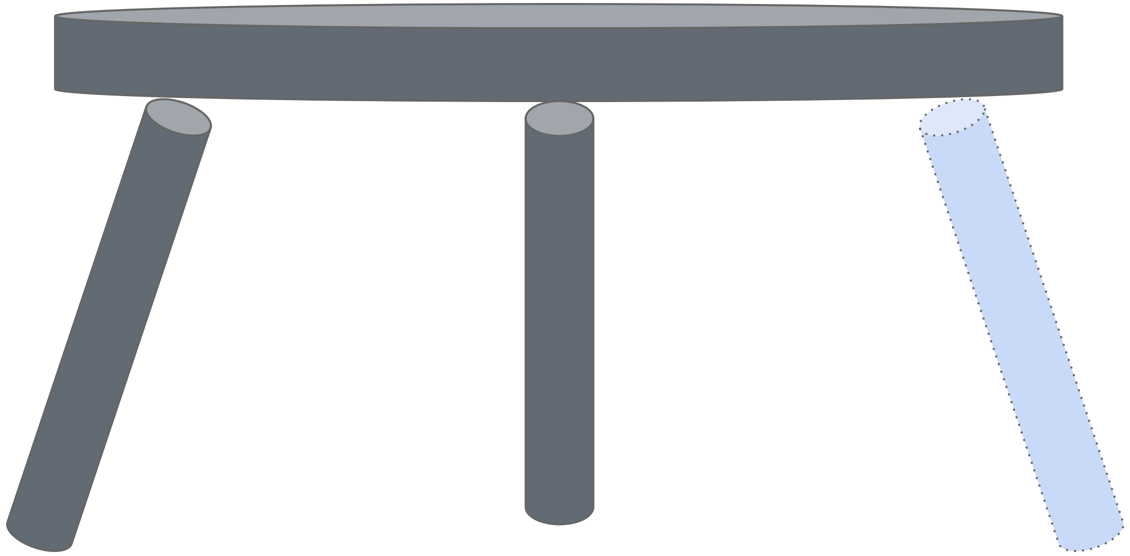
PyWren is a system we built to enable incredibly scalable
on the cloud using AWS Lambda (and other "serverless" f
mean it -- you can nearly-instantly run your code on litera
overhead, all billed in 100ms-increments.

PyWren began as a
series of exploratory
blog posts looking at
the compute scaling

This research also tells us something important about platform maturity:

You know a new platform has hit critical mass when interesting innovation is happening outside of the vendor.

BUT…the Serverless Supercomputer(™) is still missing some parts…



The Serverless Supercomputer Stack

The Serverless Supercomputer Stack

Cloud Vendors: Functions, APIs, Managed Service Portfolios

The Serverless Supercomputer Stack

Applications: Big data, analytics, ML/AI, simulations, r/t video transcoding, ...

Cloud Vendors: Functions, APIs, Managed Service Portfolios

The Serverless Supercomputer Stack

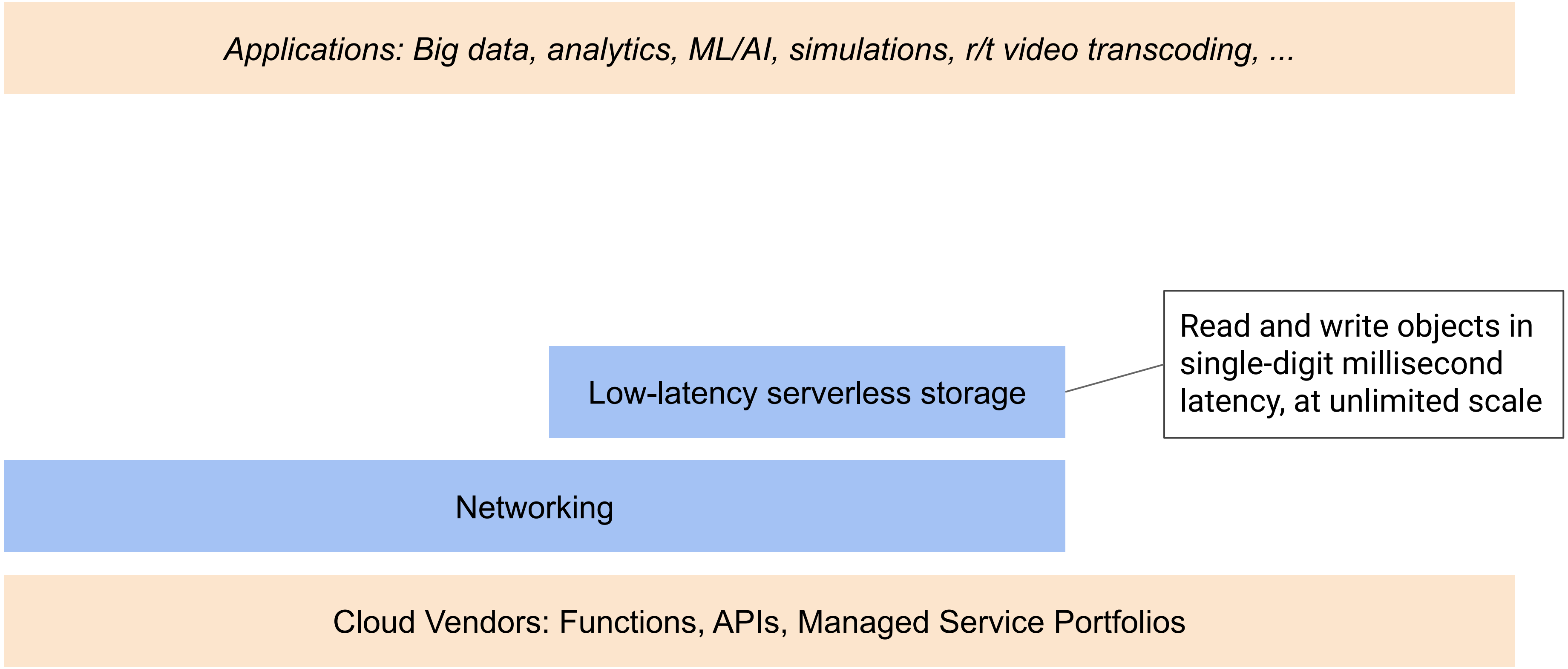
Applications: Big data, analytics, ML/AI, simulations, r/t video transcoding, ...

Fast, reliable *function-to-function* communication paths

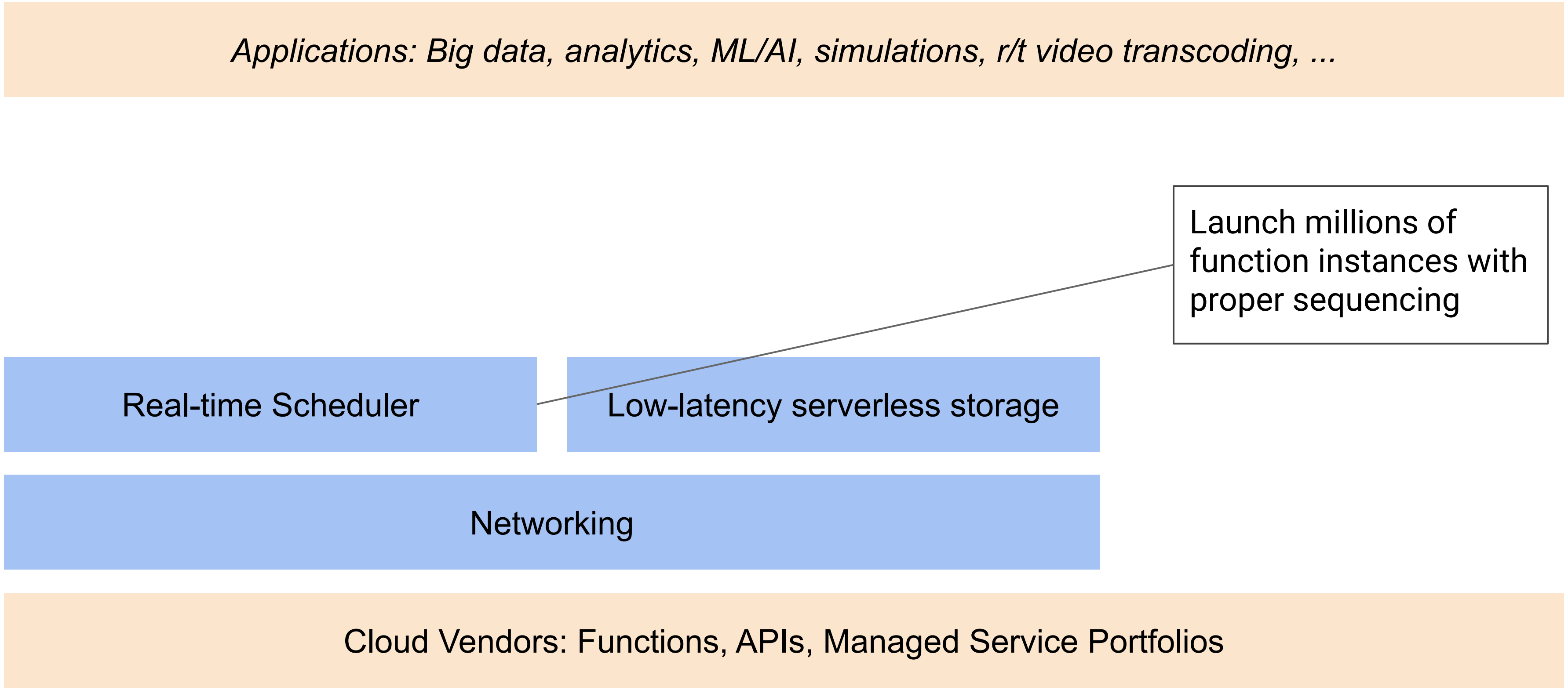
Networking

Cloud Vendors: Functions, APIs, Managed Service Portfolios

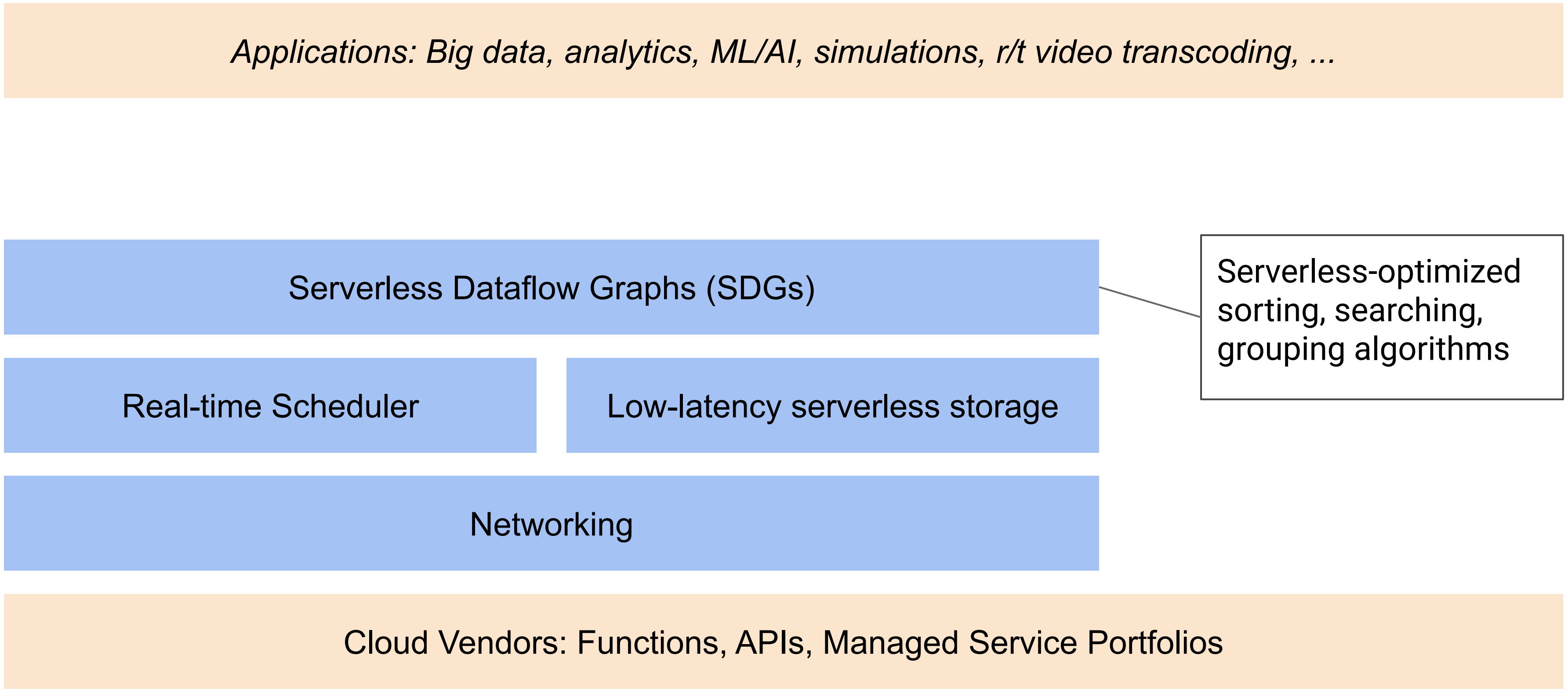
The Serverless Supercomputer Stack


Applications: Big data, analytics, ML/AI, simulations, r/t video transcoding, ...
Low-latency serverless storage
Read and write objects in single-digit millisecond latency, at unlimited scale
Networking
Cloud Vendors: Functions, APIs, Managed Service Portfolios

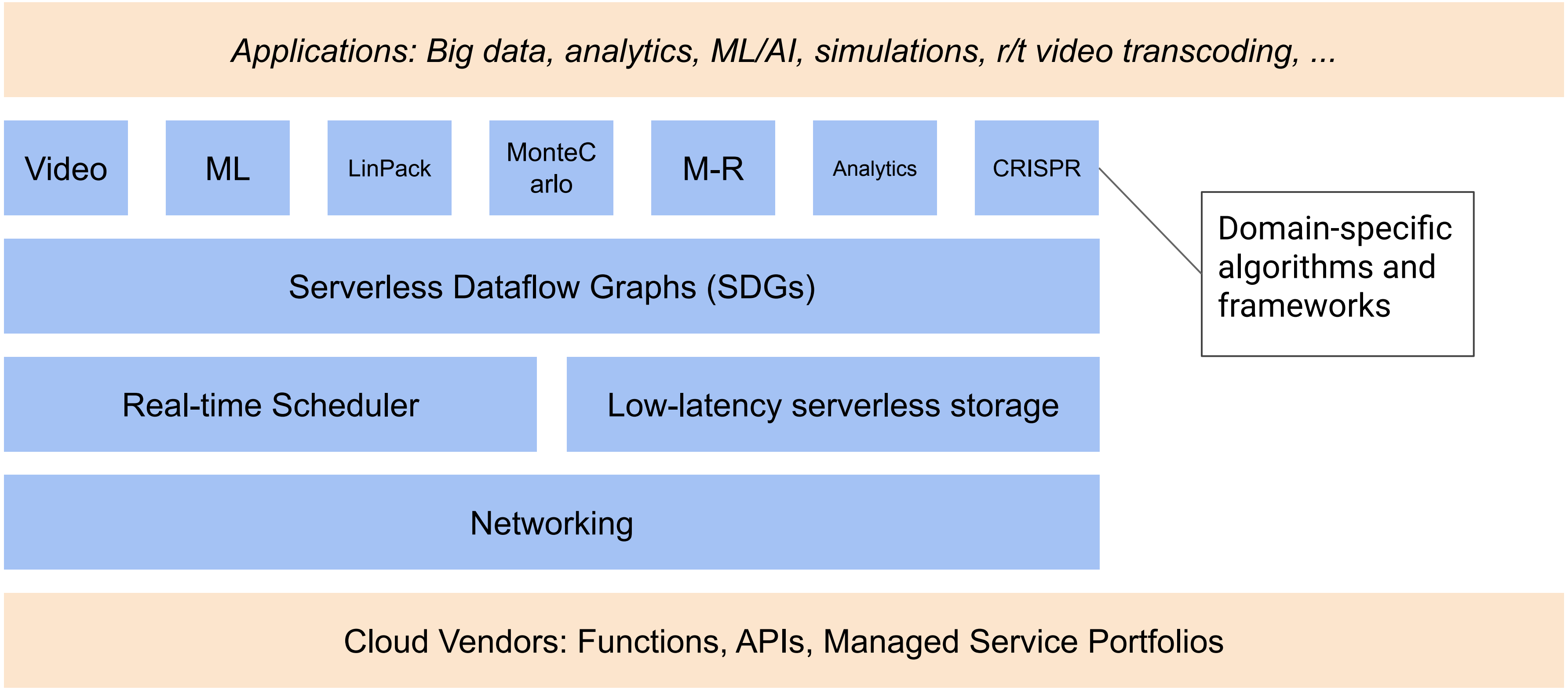
The Serverless Supercomputer Stack


Applications: Big data, analytics, ML/AI, simulations, r/t video transcoding, ...
Launch millions of function instances with proper sequencing
Real-time Scheduler
Low-latency serverless storage
Networking
Cloud Vendors: Functions, APIs, Managed Service Portfolios

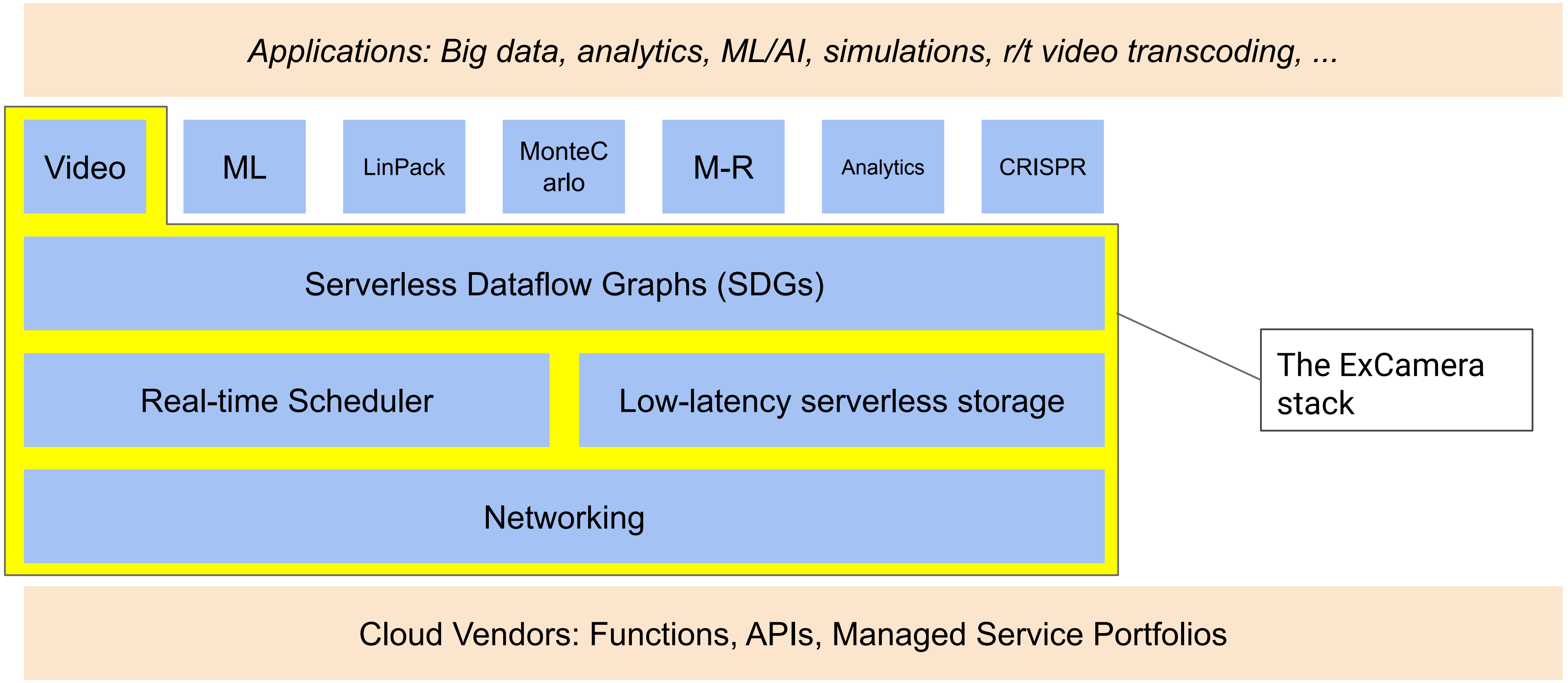
The Serverless Supercomputer Stack


Applications: Big data, analytics, ML/AI, simulations, r/t video transcoding, ...
Serverless Dataflow Graphs (SDGs)
Serverless-optimized sorting, searching, grouping algorithms
Real-time Scheduler
Low-latency serverless storage
Networking
Cloud Vendors: Functions, APIs, Managed Service Portfolios

The Serverless Supercomputer Stack


Applications: Big data, analytics, ML/AI, simulations, r/t video transcoding, ...
Video
ML
LinPack
MonteCarlo
M-R
Analytics
CRISPR
Domain-specific algorithms and frameworks
Serverless Dataflow Graphs (SDGs)
Real-time Scheduler
Low-latency serverless storage
Networking
Cloud Vendors: Functions, APIs, Managed Service Portfolios

The Serverless Supercomputer Stack


Applications: Big data, analytics, ML/AI, simulations, r/t video transcoding, ...
Video
ML
LinPack
MonteC arlo
M-R
Analytics
CRISPR
Serverless Dataflow Graphs (SDGs)
Real-time Scheduler
Low-latency serverless storage
Networking
The ExCamera stack
Cloud Vendors: Functions, APIs, Managed Service Portfolios

The Serverless Supercomputer Stack

Applications: Big data, analytics, ML/AI, simulations, r/t video transcoding, ...

TODAY:

Strictly DIY - academics and entrepreneurs

“No fly zone” for enterprises

Cloud Vendors: Functions, APIs, Managed Service Portfolios

Why this zone is so hard:

It’s stateful *and* serverless!

But…it’s also the most exciting area of computer science innovation today!

For its entire history, distributed computing research modeled *capacity* as fixed but *time* as unlimited.

With serverless *time* is limited, but *capacity is effectively infinite*.

This only changes everything :).

Let’s get started!

First up: Serverless Networking

Serverless Networking on AWS:
What works versus what's needed

Works today

- Lambda invocation through its built-in web service front end
- Fully managed APIs (CDN, regional, private)
- Event wireup from other managed services
- Call public Internet endpoints
- Call services inside a VPC (if so configured)

Don't want to duplicate this!

Missing today

- Anonymous pipes (parent-child comm)
- Easily share / create data (no socket.listen)
- Ephemeral service ownership and discovery
- Message-based orchestration (wait for something to arrive)
- Exchange files or content among Lambda functions, except by paying for storage in another AWS service (Amazon S3, etc.)

Let's build this!

Launching in beta today:

Serverless peer-to-peer networking for AWS Lambda

Serverless networking beta release

- ***Reliable* communication between Lambda functions**
  - Enables a Lambda to “serve” content to EC2 or other systems
  - Built on top of udt, a *reliable* high-speed UDP transport protocol
  - Enterprise friendly open source licensing (BSD)
  - Python 3.7 and C++ language bindings available today
  - Distributed as a public Lambda Layer, with sample code in the Amazon Serverless Application Repository
  - Available in us-east-1 today; more regions coming soon!

- ***Fully managed* NAT lookup service (no need to run a STUN server)**
  - Uses Amazon API Gateway serverless websockets
  - Free of charge during beta, modest pay-per-use pricing thereafter

What can you build with serverless networking?

Out of the box you can…

- Reliably send messages, in-memory buffers, and files between Lambda functions
- Invoke another Lambda with an anonymous bidirectional pipe between parent and child
- Use a Lambda function as a simple data server

...*without* paying for other AWS services or API calls to ship your data!

What can you build with serverless networking?

With a little work you can…

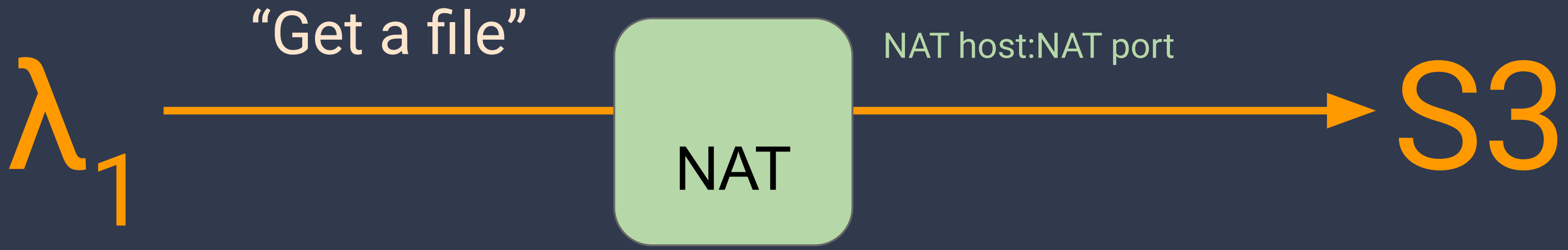
- Keep a pool of “warm Lambdas” and fire them off when you need them
- Use a Lambda instance as a (time bounded) file or model server
- Send additional arguments or get intermediate results from a function
- Change the code of a Lambda function on the fly, without re-invoking it
- Stream content to/from a Lambda in real time

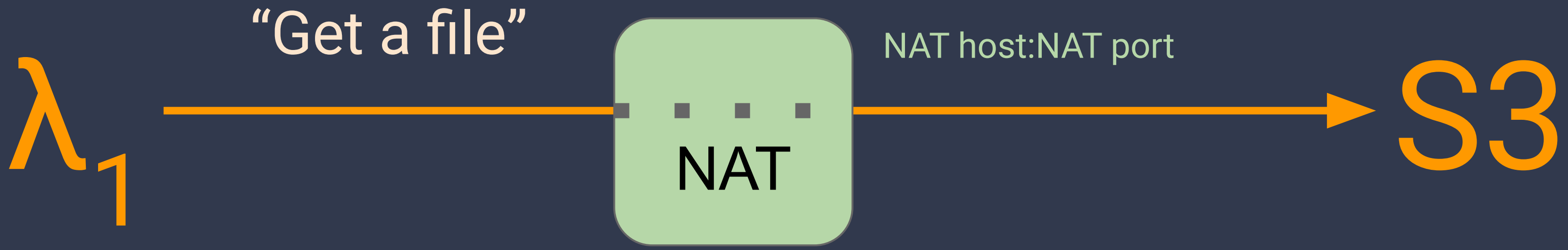
NAT punching sounds violent...
will the poor NATs be ok?

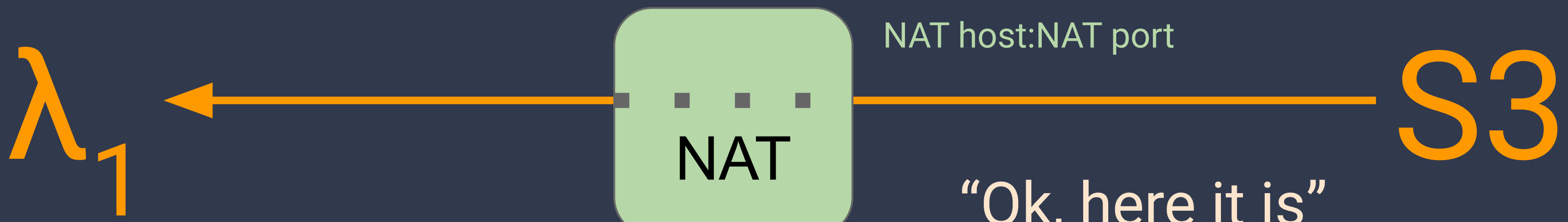
Never fear, the NATs are safe

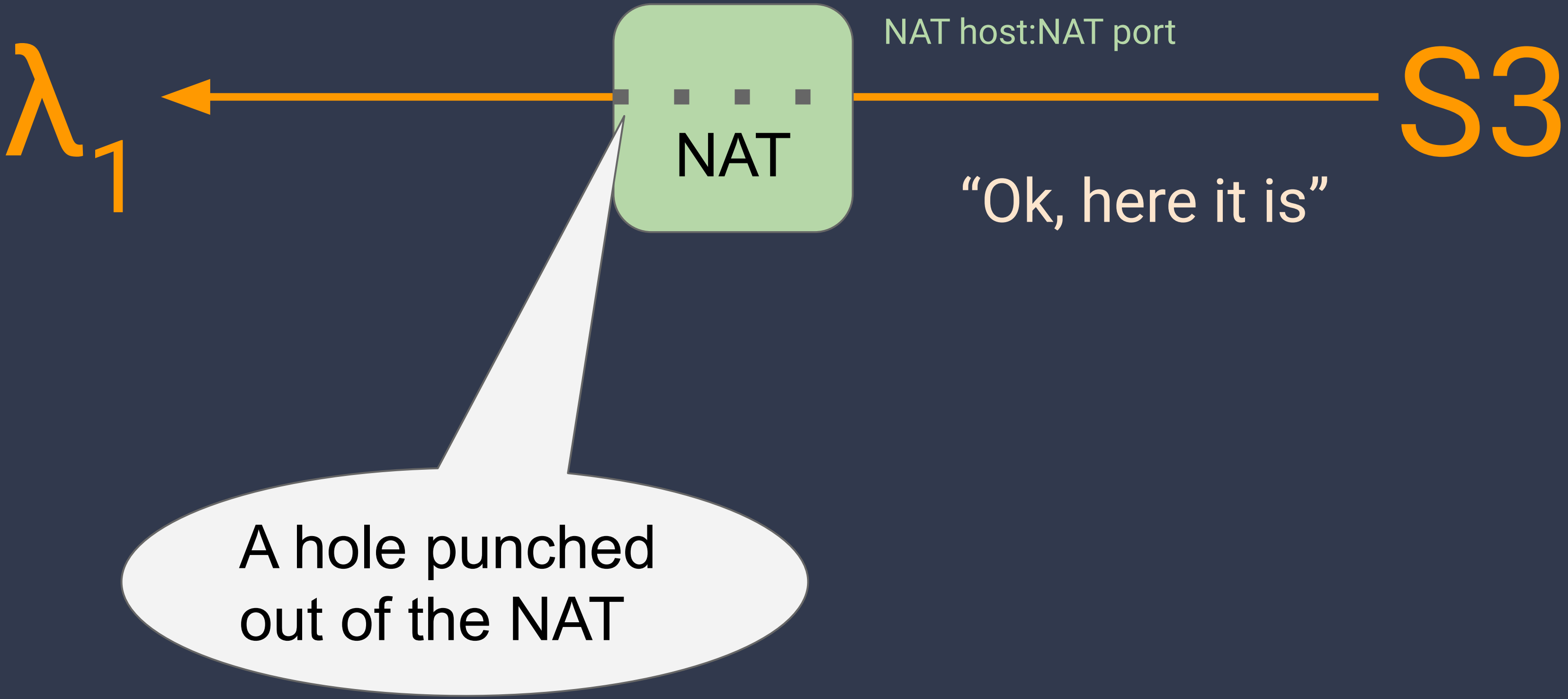
$$\begin{tikzcd} \lambda_1 \arrow[r, "\text{“Get a file”}"] & \text{S3} \end{tikzcd}$$

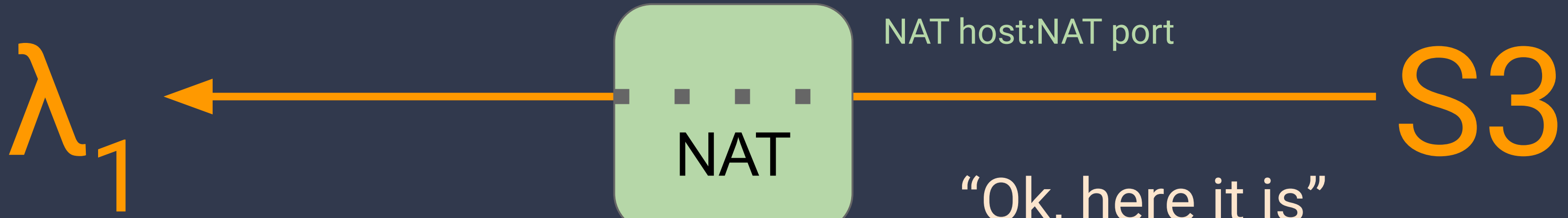
$$\begin{tikzcd}\lambda_1 & S3 \arrow[l, "\text{“Ok, here it is”}"']\end{tikzcd}$$


“Get a file”
λ_1
NAT
NAT host:NAT port
S3


λ1
"Get a file"
NAT
NAT host:NAT port
S3


λ1
NAT
NAT host:NAT port
S3
“Ok, here it is”


λ_1
NAT host:NAT port
S3
NAT
"Ok, here it is"
A hole punched out of the NAT

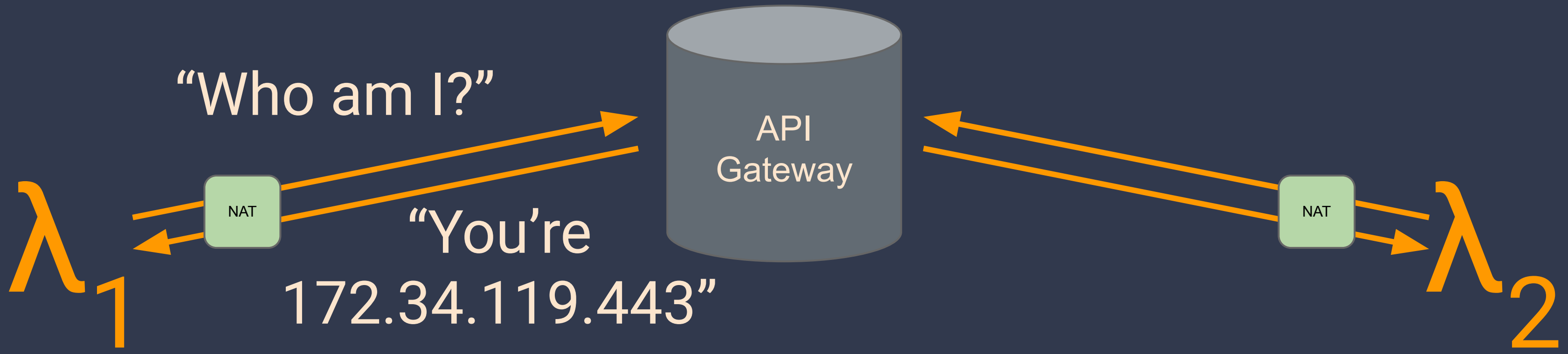


This is client-to-server NAT punching.

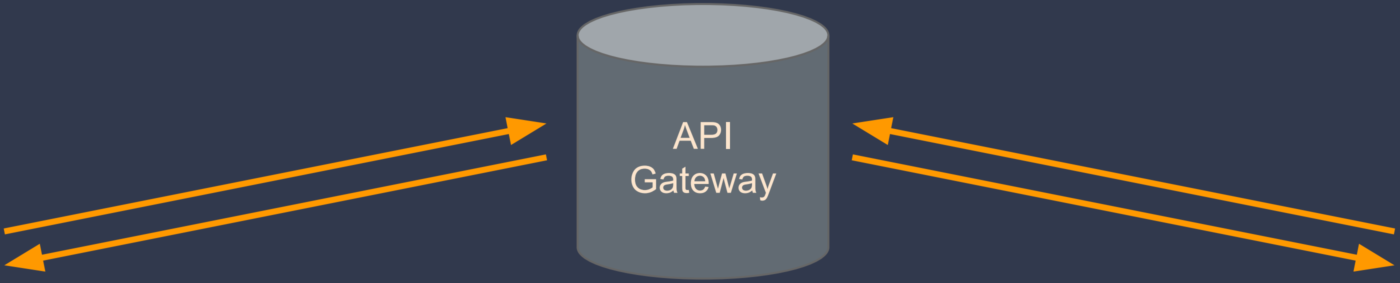
It's transparent and ubiquitous inside of a Lambda *unless* you turn on VPC without an IGW/NAT combo.

λ_1 λ_2

Peer-to-peer NAT punching is more complicated...



Step 1: The two sides use an intermediary to learn their "true" source IPs.

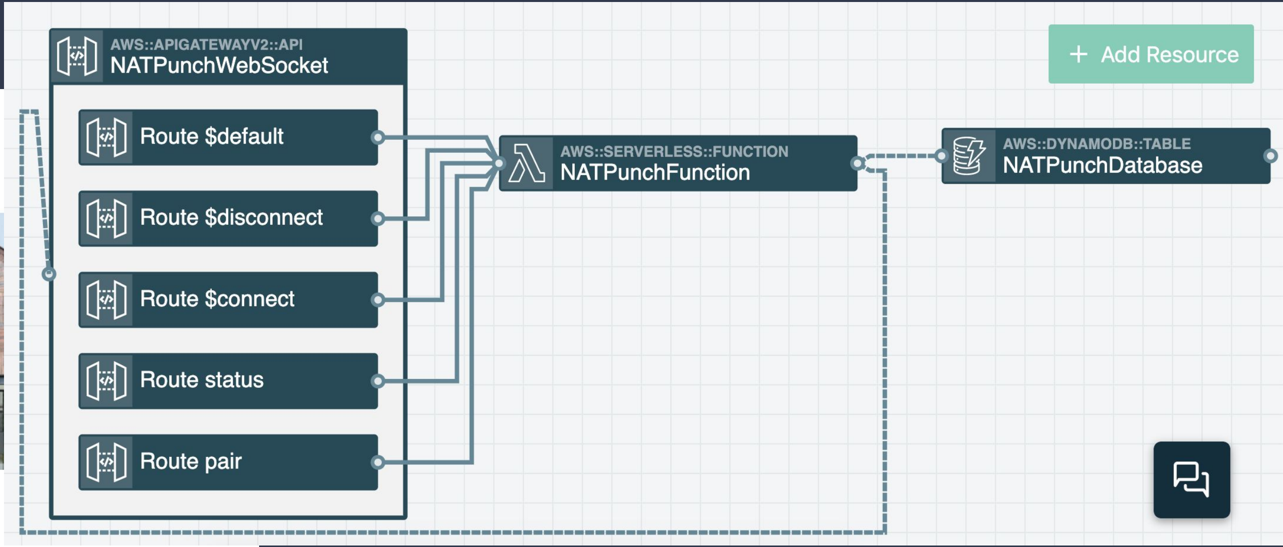


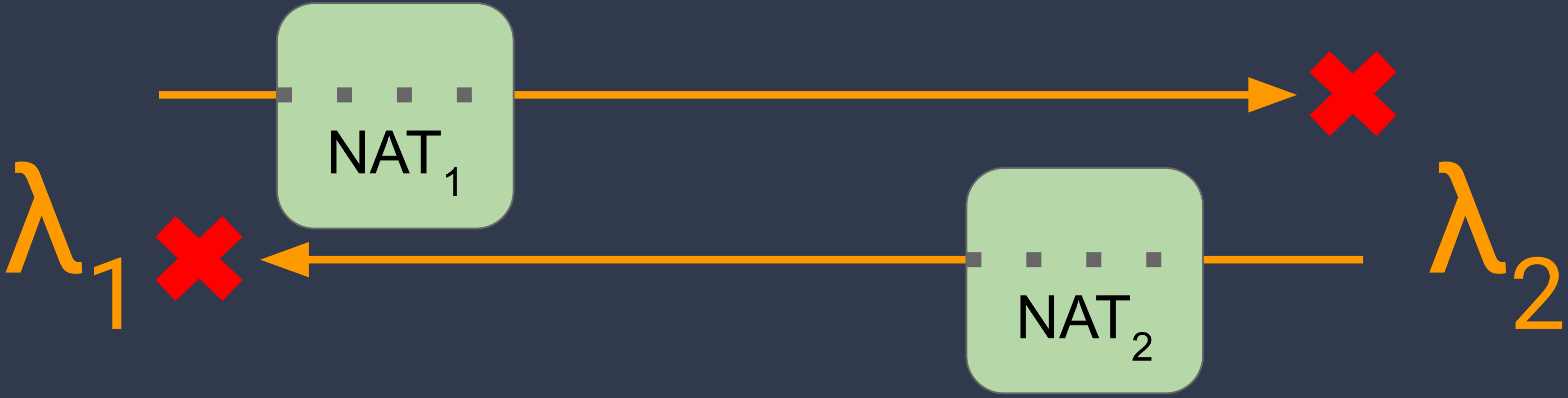
If you're curious about this ^^^^^^^

Tim Wagner in A Cloud Guru
Sep 30 · 7 min read

Serverless Mullet Architectures

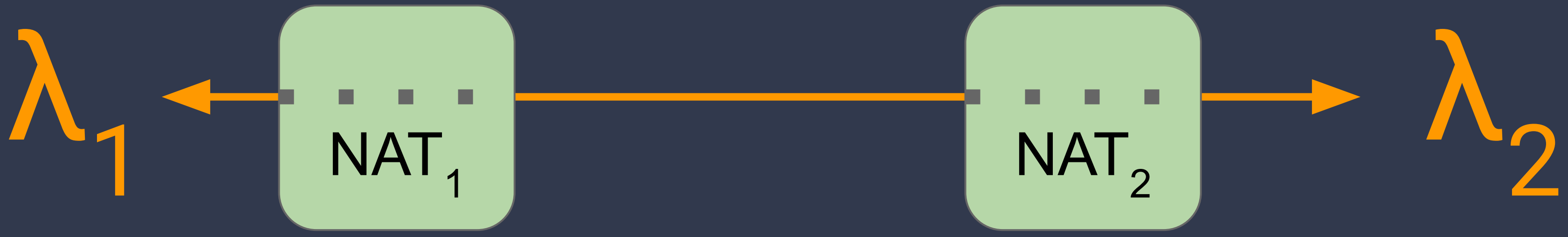
Business in the front, party in the back. Bring on the mullets!





Step 1: The two sides use an intermediary to learn their "true" source IPs.

Step 2: Each side sends a message to the other...these messages bounce, but open the NAT pathways.



Step 1: The two sides use an intermediary to learn their "true" source IPs.
Step 2: Each side sends a message to the other...these messages bounce, but open the NAT pathways.
Step 3: Normal communication starts.

Turns out there are multiple scenarios...

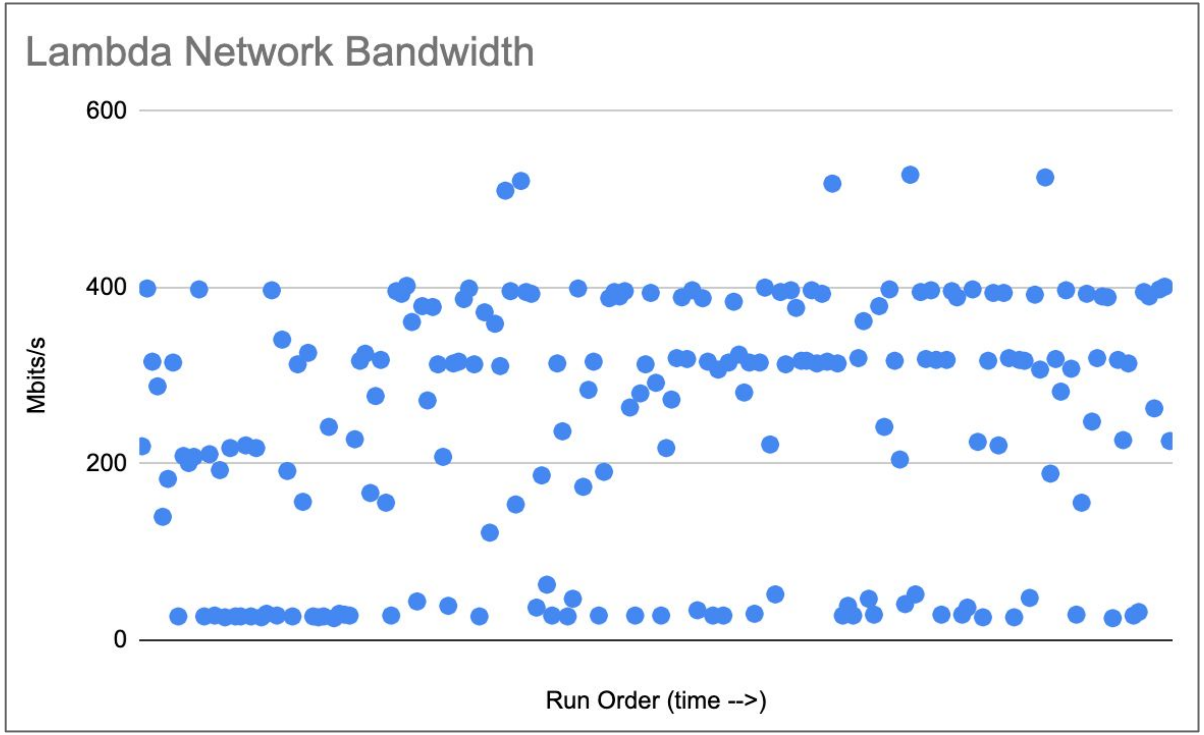
1. **Two Lambdas, neither in a VPC**
   - Cone NAT - stable address, can choose your port
   - Can use managed exchange service via API Gateway websocket
2. **Two Lambdas in the *same* VPC, with no IGW or managed NAT in their subnet routes**
   - Cone NAT - stable address, can choose your port
   - Requires a *privately hosted* exchange service
3. **One regular Lambda and one in a VPC, with IGW and managed NAT turned on**
   - Cone NAT on one side, symmetric NAT on the other
   - Works if the public side does an ephemeral port blast
   - Can use managed exchange service via API Gateway websocket

...and the sad cases:

4. **Two Lambdas in different VPCs with IGW and managed NAT in their subnets**
    - Two symmetric NATs - address and port are randomly assigned and depend on the full <source addr, source port, dest addr, dest port> quad
    - Only works if you send *and* listen on all ephemeral ports
    - Production use of this scenario would likely require NAT configuration support from AWS
5. **Two Lambdas in the *same* VPC, with IGW and managed NAT in their subnet routes**
    - The presence of the managed NAT routing turns this from scenario 2 into scenario 4 (no hairpins)

Beta supports scenario 1. Scenarios 2 & 3 have been shown to work.

Performance: Early Observations



Serverless Supercomputing:

Cloud Vendor Wishlist

- **Scalable bandwidth** - should scale with memory size
- **Consistent bandwidth** - stable performance regardless of VPC settings
- **Nicer NATs** - simplify hybrid and enable cross-VPC networking
- **Larger memory options**
- **Future: SDG-aware preprovisioning**

Want to help?

Open source contributors and innovative developers looking for a new opportunity

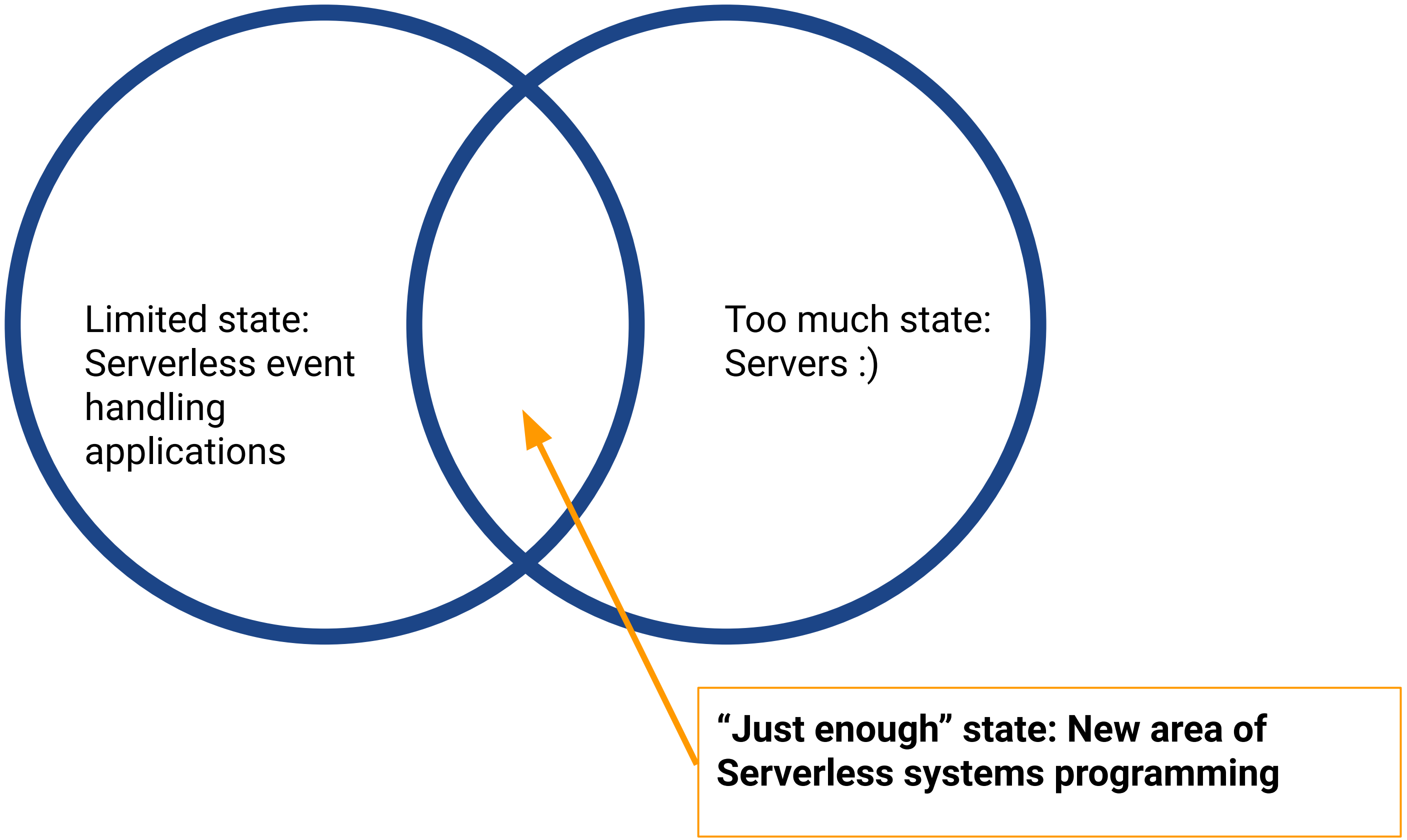
(Links are on the last slide)

Serverless Networking

- Other language bindings (Java, NodeJS, Ruby, etc.)
- Higher-level abstractions (asyncio)
- Benchmarking and testing frameworks
- VPC-to-VPC and VPC-to-Public implementations of NAT punching on AWS
- Porting to other cloud vendors
- Lots more...

Moving up the stack

- High-speed scheduling and job management
- Straggler detection & recovery
- Low-latency storage
- Immutable object management
- Serverless-optimized algorithms (including a lot of open research questions ideal for grad students!)


Limited state:
Serverless event handling applications
Too much state:
Servers :)
“Just enough” state: New area of Serverless systems programming

Build scalable *and stateful* services on a serverless foundation!

Serverless State

...the "what comes next" of serverless!

Dr. Tim Wagner

ServerlessConf NY 2019

Links

Public Lambda Layer (us-east-1 *only*):	`arn:aws:lambda:us-east-1:293602984666:layer:ServerlessNetworking-Python3:9`
Samples on AWS Serverless App Repo:	https://serverlessrepo.aws.amazon.com/applications/arn:aws:serverlessrepo:us-east-1:293602984666:applications~ServerlessNetworkingPython3
Source code:	https://github.com/serverlessunicorn/ServerlessNetworkingClients
Docs:	https://networkingclients.serverlesstech.net/
Website:	https://serverlesstech.net
Tim on twitter & DM:	timallwagner@
Tim on Medium:	@timawagner
This talk:	https://docs.google.com/presentation/d/1g06UmzJXAh_l7-uohwJyQXG0nS5J93uOpBEKYXANB7o