If a server listens in a forest and there was no one there to start it, does it really exist?
A systems-oriented biennial conference, complementary in its mission to the mainstream Information Access and Retrieval conferences, emphasizing the innovative technological aspects of search and retrieval systems.

It gathers researchers and practitioners from both academia and industry to discuss the latest innovative and visionary ideas in the field.
Computing without Servers, V8, Rocket Ships, and Other Batsh*t Crazy Ideas in Data Systems

Jimmy Lin
David R. Cheriton School of Computer Science
University of Waterloo

Wednesday, August 29, 2018
We’ve done it!
Serverless BM25 Search and BERT Reranking
Servers

The most fundamental building blocks of IR systems
(both software and hardware)
In the beginning...
“Google” Circa 1997 (google.stanford.edu)

Slide from Jeff Dean, WSDM 2009 keynote
“Corkboards” (1999)

Slide from Jeff Dean, WSDM 2009 keynote
Google Data Center (2000)

Slide from Jeff Dean, WSDM 2009 keynote
Google Data Center (3 days later)
Servers

... in the cloud
Early 2001: In-Memory Index

Slide from Jeff Dean, WSDM 2009 keynote
Challenges Remain
(Especially if you’re not Google)

Always on!
Scaling up… scaling down…
Scaling to zero?
Serverless Preliminaries
Cloud computing allows us to explore different abstractions and organizations of computing
(trend towards disaggregation)
In the beginning...
Infrastructure as a Service (IaaS)
Multi-Tenancy

- Hypervisor
- Physical Machine

- VM
- OS
- App
Containers >> VMs
Typical Stack

- App
- OpenSearch
- Container
- Container Engine
- OS
- Physical Machine
Multi-Container Orchestration

- Apps
  - OpenSearch
  - Container

- Container Engine
- OS
- Physical Machine
Platform as a Service

App
OpenSearch as a Service
OpenSearch
Container
Container Engine
OS
Physical Machine

App
OpenSearch
Container
Container Engine
OS
Physical Machine

App
OpenSearch
Container
Container Engine
OS
Physical Machine

App
OpenSearch
Container
Container Engine
OS
Physical Machine
Platform as a Service

Service-level agreements (SLA) in terms of latency, capacity, scalability, etc.
What about the apps?

OpenSearch as a Service

- App
- Container
- Container Engine
- OS
- Physical Machine
Scaling out the apps...
Operational Semantics of Computing...

\[ \langle E, s \rangle \Rightarrow V \]
\[ \langle L := E, s \rangle \rightarrow (s \cup (L \mapsto V)) \]

if the expression \( E \) in state \( s \) reduces to value \( V \),
then the program \( L := E \) will update the state \( s \) with the assignment \( L = V \)

State  State as a Service
(persistent storage, databases, message queues, …)

Transitions  Function as a Service
(blocks of code with a well-defined entry and exit points)
Computing without Servers

Developer: Write a bunch of functions typically – read state, perform some computation, update state

State as a Service
(persistent storage, databases, …)

Cloud provider handles everything else!
allocation of resources for execution, scaling up and down, load balancing, cleaning up, etc.

Cost model: pay per function invocation
Serverless Examples

Example: Image Thumbnail Creation

Photograph is taken → S3

Lambda is triggered

Photo is uploaded to S3 Bucket

Lambda runs image resizing code to generate web, mobile, and tablet sizes

Example: Retail Data Warehouse ETL

Online order is placed

Order data is stored in operational database

Lambda is triggered

Lambda runs data transformation code and loads results into data warehouse

Analytics generated from data

Source: Amazon Web Services
Serverless computing isn’t actually computing without servers…

It’s just that servers become someone else’s problem!
If a server listens in a forest and there was no one there to start it, does it really exist?
What would a serverless search engine look like?
Serverless Search

Where do the postings go?

Lambda function as a service

DynamoDB key-value store (postings lists)

Client

API Gateway

Query Evaluation

Serverless Search

Lambda function as a service

DynamoDB key-value store (postings lists)

I got 99 problems but scaling ain't one!
How well does serverless search work?

**tl; dr – not very well…**

Take 2
(This work)
Technical Highlights

tl;dr – Serverless Lucene
Index structures stored on S3
Query evaluation in AWS Lambda

Minimal modifications to “vanilla” Lucene!

What about query latency?
“cold instance” startup – loads indexes in memory
“warm instance” execution – indexes already in memory

No different from “standard” in-memory search!
Serverless Lucene isn’t enough!
Retrieve + Rerank

Select some promising texts

Rerank selected texts

Lucene
Retrieve + Rerank

Select some promising texts +

Serverless!

Lucene

monoBERT
(cross-encoder reranker)

Serverless?
Yea, so what about serverless BERT?
(Good thing it’s embarrassingly parallel!)
Technical Highlights

tl;dr – Serverless BERT inference in AWS Lambda

“Early-Exit” optimizations* to reduce inference latency
Main technical challenge is model size – solved by Elastic Container Registry

It’s SMOP!

The bad: CPU inference
The good: Massive parallelism (100-way parallelism)
The ugly: Image packaging is a pain!

Search client

Web Browser

API Gateway

FetchLambda

SearchLambda

Lucene indexes

S3

Raw documents

DynamoDB

Serverless Search
## Results
(MS MARCO passage ranking)

Setup: reranking 1000 hits from Lucene with monoBERT

<table>
<thead>
<tr>
<th>Stage</th>
<th>Latency (s/Q)</th>
<th>Cost (/100Q)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Mean</td>
<td>P50</td>
</tr>
<tr>
<td>BM25</td>
<td>0.65</td>
<td>0.65</td>
</tr>
<tr>
<td>DynamoDB Fetch</td>
<td>0.95</td>
<td>0.96</td>
</tr>
<tr>
<td>BERT reranking</td>
<td>11.21</td>
<td>10.64</td>
</tr>
<tr>
<td>End to end</td>
<td>12.81</td>
<td>12.24</td>
</tr>
<tr>
<td>BERT reranking (V100)</td>
<td>26.21</td>
<td>25.52</td>
</tr>
</tbody>
</table>

(We confirmed that effectiveness is the same for serverless vs. server-based deployments)

Wait, how can CPU be faster than GPU?
7-8× more expensive = breakeven at 85%-90% idle
Objections

Source: https://www.flickr.com/photos/usdagov/22484527807/
It’s still too slow! (12s end-to-end)
It’s still too expensive! ($0.16 per query)

Agreed… but this is only the beginning!
Serverless infrastructure will become more efficient.
Lots of neural inference acceleration techniques to try.

The price of minimal management overhead and infinite scaling?
Serverless search is worth considering?!
Questions?