Serverless
for data and AI

Avner Braverman
Binaris CEO
The future is **Serverless**
<table>
<thead>
<tr>
<th></th>
<th>Ride sharing</th>
<th>Serverless</th>
</tr>
</thead>
<tbody>
<tr>
<td>Infrastructure</td>
<td>Someone else's car</td>
<td>Never manage servers</td>
</tr>
<tr>
<td>Cost</td>
<td>Pay per ride</td>
<td>Pay per CPU time</td>
</tr>
</tbody>
</table>
Legacy applications

3% utilization

Cloud-Native applications

30% utilization
<table>
<thead>
<tr>
<th></th>
<th>Ride sharing</th>
<th>Serverless</th>
</tr>
</thead>
<tbody>
<tr>
<td>Infrastructure</td>
<td>Someone else's car</td>
<td>Never manage servers</td>
</tr>
<tr>
<td>Cost</td>
<td>Pay per ride</td>
<td>Pay per CPU time</td>
</tr>
<tr>
<td>Ease of use</td>
<td>Focus on your phone</td>
<td>Focus on business logic</td>
</tr>
<tr>
<td>Who can use?</td>
<td>No driver's license</td>
<td>No cloud expertise</td>
</tr>
</tbody>
</table>
myFunction()

exports.handler = () => {
  return 'hello, world!';
};
Demo
Key providers

- AWS Lambda
- Azure Functions
- Binaris
- Google Cloud Functions
- IBM Cloud Functions
① Serverless ETL
② Serverless MapReduce
③ Serverless streaming
④ Serverless training
⑤ Serverless inference
Serverless ETL

- S3
- Kinesis
- HTTP
- F()
- S3
- RedShift
### Upsides
- Easy to build
- Elastic
- Usually cheaper to run

### Challenges
- Micro-batching
- Run duration limits *(15 min on AWS Lambda)*
- Memory limits *(up to 3GB on AWS Lambda)*

---

**ETL accounts for 90% of Lambda CPU cycles**
① Serverless ETL
② Serverless MapReduce
③ Serverless streaming
④ Serverless training
⑤ Serverless inference
Serverless MapReduce

Data \rightarrow map() \rightarrow \ldots \rightarrow reduce() \rightarrow Result
Calculating $\pi$ using the Monte Carlo method:

1. Spread random points inside a square

2. Geometry tells us that
   - Circle area $\sim$ # red points $\sim \pi \cdot R^2$
   - Square area $\sim$ # points $\sim 4 \cdot R^2$

3. Compute:
   $\pi = 4 \times (# \text{ red points}) / (# \text{ points})$
function computePi(points) {
    let inside = 0;

    // repeat points times:
    for (let i = 0; i < points; i++) {

        // random point (use R = 1)
        const x = Math.random() * 2 - 1;
        const y = Math.random() * 2 - 1;

        // is it inside the circle?
        if (x * x + y * y < 1) {
            inside++;
        }
    }
    return inside;
    // use circle to square area ratios to compute π
    return 4 * inside / points;
}
exports.computePiMapper = ({ points }) => {
  let inside = 0;

  // repeat points times:
  for (let i = 0; i < points; i++) {
    const x = Math.random() * 2 - 1;
    const y = Math.random() * 2 - 1;

    // is it inside the circle?
    if (x * x + y * y < 1) {
      inside++;
    }
  }

  return inside;
}

exports.computePiReducer = ({ inputs, points }) => {
  return 4 * inputs.reduce((a, b) => a + b) / points;
}
Demo
# Serverless MapReduce

## Upsides
- Easy to build
- Elastic
- Scale at your fingertips

## Challenges
- Failures?
- Retries?
- Laggards?
Serverless MapReduce

```javascript
function mrAsyncController(...) {
  // add map commands to stream
}
```

```javascript
function mrMapper(...) {
  // invoke user mapper
  // save output
  // increase counter
  // if complete:
  //   invoke user reducer
}
```

```javascript
function mrAsyncController(...) {
  // add map commands to stream
}
```

```javascript
function mrMapper(...) {
  // ...
}
```

```javascript
function mrAsyncController(...) {
  // add map commands to stream
}
```

```javascript
function mrMapper(...) {
  // ...
}
```

```javascript
function mrAsyncController(...) {
  // add map commands to stream
}
```

```javascript
function mrMapper(...) {
  // ...
}
```

```javascript
function mrAsyncController(...) {
  // add map commands to stream
}
```
Serverless MapReduce

github.com/binaris/functions-examples
Serverless MapReduce

**Upside**
- Easy to build *(with framework)*
- Elastic
- Scale to your fingertips

**Challenges**
- Need cache (e.g. Redis) to hold state
- Need storage (e.g. S3) to handle data
① Serverless ETL
② Serverless MapReduce
③ Serverless streaming
④ Serverless training
⑤ Serverless inference
Serverless streaming

Stream $\rightarrow F() \rightarrow$ Stream $\rightarrow G()$
Serverless streaming
Serverless streaming

**Upsides**
- Easy to build
- Elastic
- Scale to your fingertips
- Real-time

**Challenges**
- Need a smarter framework

Shameless plug for Binaris
Today, serverless is not a good fit for training 😞

But, if

① Your data is small (few GB)
② You need hyperparameter optimization

Then you can leverage serverless scale!
① Serverless ETL
② Serverless MapReduce
③ Serverless streaming
④ Serverless training
⑤ Serverless inference
Serverless inference

(Yeah, we know this is not really inference, but same computational pattern)
Demo
<table>
<thead>
<tr>
<th>Upsides</th>
<th>Challenges</th>
</tr>
</thead>
<tbody>
<tr>
<td>Easy to build</td>
<td>No GPU</td>
</tr>
<tr>
<td>Use standard tools (py, sklearn, TF)</td>
<td>Might be limited CPU/memory for some TF models</td>
</tr>
<tr>
<td>Elastic</td>
<td></td>
</tr>
<tr>
<td>Scale to your fingertips</td>
<td></td>
</tr>
<tr>
<td>Real-time (no more plugs)</td>
<td></td>
</tr>
</tbody>
</table>
① Serverless ETL
② Serverless MapReduce
③ Serverless streaming
④ Serverless training
⑤ Serverless inference
Serverless

for data and AI

Serverless is easy to use
Cosmic ✨ scale without the rocket 🚀 science
Pay as you go

avner@binaris.com   @avnerbraverman
Rate today’s session

Cyberconflict: A new era of war, sabotage, and fear

Location: Ballroom
Secondary topics: Security and Privacy

We’re living in a new era of constant sabotage, misinformation, and fear, in which everyone is a target, and you’re often the collateral damage in a growing conflict among states. From crippling infrastructure to sowing discord and doubt, cyber is now the weapon of choice for democracies, dictators, and terrorists.

David Sanger explains how the rise of cyberweapons has transformed geopolitics like nothing since the invention of the atomic bomb. Moving from the White House Situation Room to the dens of Chinese, Russian, North Korean, and Iranian hackers to the boardrooms of Silicon Valley, David reveals a world coming face-to-face with the perils of technological revolution—a conflict that the United States helped start when it began using cyberweapons against Iranian nuclear plants and North Korean missile launches. But now we find ourselves in a conflict we’re uncertain how to control, as our adversaries exploit vulnerabilities in our hyperconnected nation and we struggle to figure out how to deter these complex, short-of-war attacks.

David Sanger
The New York Times

David E. Sanger is the national security correspondent for the New York Times as well as a national security and political contributor for CNN and a frequent guest on CBS This Morning, Face the Nation, and many PBS shows.

O’Reilly Events App

Session page on conference website