Unikernels and Event-driven Serverless Platforms

Madhuri Yechuri

Agenda

- Bio
- Application Deployment Paradigms - Past, Present, Future
- Why Serverless?
- Advantages of Event-driven Serverless Model
- Event-driven application: shrink wrap needs
- Event-driven application: shrink wrap options (current)
- Unikernel definition, demo
- Event-driven application: shrink wrap options (future)
- Acknowledgements
- Q & A

Bio

- Bachelors in Computer Science (IIT Kharagpur)
- Masters in Computer Science (IU Bloomington)
- 11+ years at Oracle Database Server Technologies (RAC, TimesTen)
- 3 years at VMware (Distributed Resource Scheduler)
- 1.5 years at ClusterHQ (Flocker)
- 1 year at Elotl (stealth)

Application Deployment Paradigms - Past, Present, Future

- Past
  - (Heavyweight) Monolithic App
  - Platform: Private Cloud
  - Application Shrink Wrap: Virtual Machine
- Present
  - (Lightweight) Microservice App
  - Platform: Private Cloud, **Public Cloud**
  - Application Shrink Wrap: Containers
- Future
  - (Lightweight) Microservice App
  - Platform: Private Cloud, **Public Cloud, <u>IoT</u>**
  - Application Shrink Wrap: Containers, unikernels?

Why Serverless?

Always-on microservices lead to -

- **Always burning (cpu, memory, network) resources**
  - Resources == $$$
- **Orchestration framework overhead**
  - Start, health check, load balance a microservice that is only needed for “if this then that” event.
- **Provisioning and Auto-scaling resource foresight**
  - How many resources (cpu, memory) will each instance of my microservice need to be “happy” under peak workload?
  - How will my microservice scale with workload?
- **On-disk image** backing always-on microservice needs to be **in-situ on every IoT Edge device**

Advantages of Event-driven Serverless Model

- Reduce Operational costs == lower cloud bills
  - Use (cpu, memory, network) resources only when there is a need from application workload
- Reduce moving parts == reduce points of failure
  - Reduce orchestration framework bookkeeping when there is no client workload for the app
- Improve app performance == happier customer
  - Minimize application performance impact due to incorrect resource provisioning decisions made ahead of time
- Improve app mobility == expand into IoT markets
  - Minimize on-disk footprint of the app so that it can be easily stretched across Private/Public cloud and IoT Edge devices.

Event-driven App: Shrink Wrap Needs

- **Lightweight**
    - On-disk
        - Image size should be small to allow functions to run across traditional and IoT compute nodes
    - Runtime
        - Resource (cpu, memory) overhead should be low
- **Agile**
    - Recyclable
        - Application startup and shutdown times should be low
    - Reusable
- **Secure**
        - Application runtime security vulnerabilities should be minimal
- **Observable**
    - Application Performance Monitoring hooks

Shrink-wrap evaluation - sample app

Nodejs webserver:

```
// Load the http module to create an http server.
var http = require('http');

// Configure our HTTP server to respond with Hello World to all requests.
var server = http.createServer(function (request, response) {
  response.writeHead(200, {"Content-Type": "text/plain"});
  response.end("Hello World\n");
});

// Listen on port 8002, IP defaults to 127.0.0.1
server.listen(8002);

// Put a friendly message on the terminal
console.log("Server running at http://127.0.0.1:8002/");
```

Event-driven App: Shrink Wrap Options (Current)

App: Nodejs webserver
Platform: Ubuntu 16.04 Server (Linux 4.4.0-51-generic)

	On-disk image size (MB)	Agility - Start time (seconds)	Agility - Runtime Memory Overhead (MB)	Security vulnerabilities	APM
Container (Alpine 3.5 base)	53.48	1.13	274.4	Inherit Linux vulnerabilities (ex: VENOM attack)	Vanilla (Amazon CloudWatch), Custom (IOPipes)

Event-driven App: Shrink Wrap Options (Current)

AWS Lambda	**Google Functions**	**Microsoft Azure Functions**	**IBM OpenWhisk**
Container	Container	Container	Container

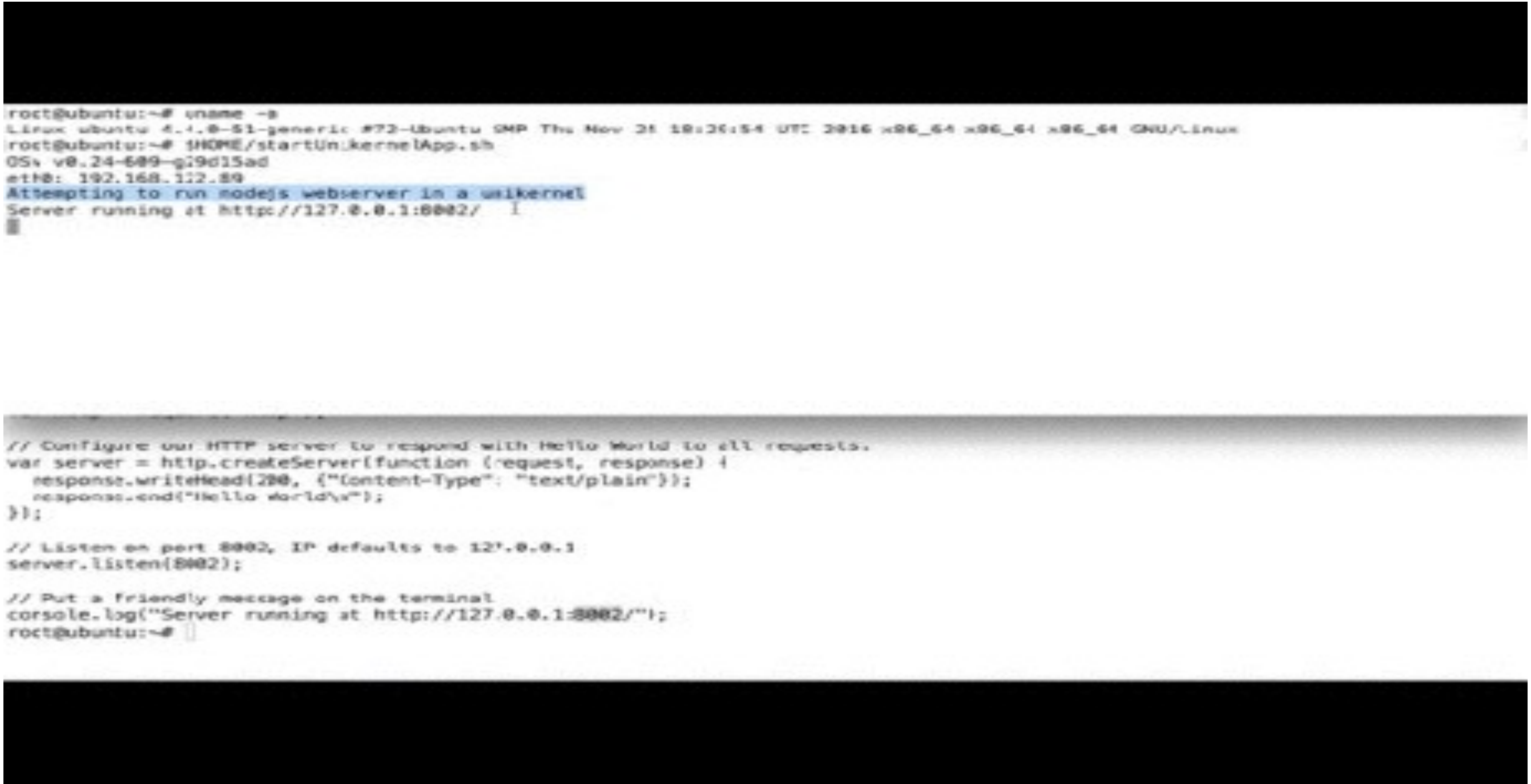
Event-driven App: Shrink Wrap Options (Future)

Are there any other shrink wrap options that meet Event-driven Application's needs?

Unikernel - Definition

- Unikernel (working definition)
  - Single purpose (single-process) virtual appliance (multi-threading available)
  - Statically linked image of your Application and a hypervisor (no general OS or extra library code)
  - No extraneous services, no full-fledged shell, no fork() facility to start a second process

Unikernel - Demo



```
root@ubuntu:~# uname -a
Linux ubuntu 4.4.0-51-generic #72-Ubuntu SMP Thu Nov 24 18:29:54 UTC 2016 x86_64 x86_64 x86_64 GNU/Linux
root@ubuntu:~# $HOME/startUnikernelApp.sh
OSv v0.24-609-g29d15ad
eth0: 192.168.122.89
Attempting to run nodejs webserver in a unikernel
Server running at http://127.0.0.1:8002/
```

```
// Configure our HTTP server to respond with Hello World to all requests.
var server = http.createServer(function (request, response) {
  response.writeHead(200, {"Content-Type": "text/plain"});
  response.end("Hello World\n");
});

// Listen on port 8002, IP defaults to 127.0.0.1
server.listen(8002);

// Put a friendly message on the terminal
console.log("Server running at http://127.0.0.1:8002/");
root@ubuntu:~#
```

Event-driven App - Shrink Wrap options (future)

	On-disk image size (MB) - lower is better	**Agility - Start time** (seconds) - lower is better	**Agility - Runtime Memory Overhead** (MB) - lower is better	**Security vulnerabilities** - Fewer is better	**APM**
Container	53.48	1.13	274.4 **(126% smaller)**	Inherit Linux vulnerabilities (ex: VENOM attack)	Amazon CloudWatch, IOPipes, etc
Unikernel	27.8 **(93% smaller)**	0.483 **(134% faster)**	619	Minimal attack surface	TBD

Takeaways

- Serverless is a good fit for cost effectively running microservice applications on existing platforms (private/public cloud)
- Containers are a good fit to back serverless platforms on private/public cloud
- Unikernels exhibit promising characteristics to be a good fit for running microservice applications on existing (private/public cloud) and emerging (IoT edge) platforms.

Acknowledgements

- Emit organizers - Nick Gottlieb, Casey Shultz
- Serverless.com
- OSv
- Rean Griffith
- Audience - Thank you!

Questions?

madhuri@elotl.co