Run Node.js in a unikernel reliably

Andreia Ocanoia | andreia@genezio.com
Our Vision

- Optimize resource allocation and cost
- Fast response times for calls
- Easy-to-use, secure, auto-scaling cloud platform
Genezio Runtime FaaS Architecture
Challenges

1. Reduce cold starts
2. Upgrade unikernel image without redeploying user code
3. Security and process isolation
Reduce cold starts (1) - Create snapshots

1. Boot the unikernel
2. Start NodeJs
3. Create Snapshot
Reduce cold starts (2) - Spawn a new VM

1. Start Firecracker
2. Load Snapshot
3. Attach new disk with user code
4. Mount disk + Node import
Reduce cold starts (3) - Warmed VMs Pool
Upgrade unikernel image

- User code is mounted as a separate filesystem
- Enable OSv/unikernel upgrades without rebuilding each user image
Security and Isolation

- Use Firecracker Jailer to run each process sandboxed
  - Separate network namespace
  - Separate filesystem (chroot)
  - Separate PID namespace
- Limit resources for VMs (CPU time, I/O throughput)
Contributions - Fixed bugs

- NodeJS/V8 compiler uses `popf` instruction and it incorrectly disables the interrupts when running in privileged mode.
- Fix `rofs` cache bug that was preventing multiple `rofs` in the same OSv instance.
- `pthread_rwlock_tryrdlock` and `pthread_rwlock_trywrlock` not POSIX-compliant resulting in a deadlock.
Round-Trip Time Benchmark
Benchmark Setup

- AWS Lambda
- (Genezio) OSv
- (Genezio) Linux
Hello World Example

```typescript
export class HelloWorldService {
  helloWorld(): string {
    console.log("Hello world - request received!")
    return "Hello world!";
  }
}
```
Hello world

Cold call
Hello world

Warm call
Next Steps

- Deploy using more unikernels - currently we tested only with OSv
- Add support for more programming languages
Let’s stay in touch

GitHub Org: https://github.com/genez-io

Team Contact:
andreia@genezio.com
vali@genezio.com
bogdan@genezio.com
Resources
