MeSHwA: The case for a Memory-Safe Software and Hardware Architecture for Serverless Computing

Anjo Vahldiek-Oberwagner, Mona Vij
Intel Labs
Monolithic Webservices

- Rating
- Rate Archive
- Frontend
- GeoLocation
- Reservation
- Hotels DB

Operating System

VM

CPU

NIC

CSP run
Monolithic Serverless Computing

- Rating
- Rate Archive
- GeoLocation
- Reservation
- Hotels DB

Frontend

Operating System

VM

CPU

NIC
Monolithic to Microservices

Decompose service into communicating functional units individually deployed in containers

Rating
GeoLocation
Reservation
Rate Archive
Hotels DB

CSP run

| Rating   | GeoLocation | Reservation | Rate Archive | Hotels DB |
|----------|-------------|-------------|--------------|-----------|
| OS VM    | OS VM       | OS VM       | OS VM        | OS VM     |
| CPU      | NIC         |             |              |           |
Microservices to Function-as-a-Service (FaaS)

Decompose even further into functions triggered when needed

CSP run
Emergence of Infrastructure Tax

- 25% of CPU cycles spent on marshalling, memory copy, synchronization, ...

![Diagram showing the emergence of infrastructure tax with interconnections between Rating, GeoLocation, Reservation, Rate Archive, and Hotels DB with relevant services like OS and VM on CPU and NIC.]
Service Meshes and Proxies

- Proxy encapsulates service functionality (e.g., load balancing)
- eBPF proxies reduced overhead, still observable
Shared memory as an alternative?

- Involves marshalling and copying of objects
- Needs synchronization or polling
- Needs library capable of regular and shared-mem network
Infrastructure Processing Units (IPUs) as an alternative?

- Offloads cost to cheaper HW, freeing main CPU
- Focused on remote communication
Memory-Safe Software/Hardware Architecture (MeSHwA)

- Execute in single address space
- Isolate using memory-safety guarantees of languages and RT

Communication is a function call

---

Memory Safety-based Isolation

- GeoLocation
- Reservation
- OS
- Scheduler
- FS
- Net
- Devices
- VM
- CPU
- NIC
Co-Designing the Software/Hardware Tradeoff

Software-only

Software defines computation; Hardware defines execution abstraction.
Rise of Memory-Safe Languages and Runtimes

- Provide safeguards for memory accesses and control flow

- Microsoft/Google report 70% of security vulnerabilities caused by memory safety violations
  - Microsoft, Google, Amazon, FB, NSA urge use of memory safe languages
Examples: Webassembly/Rust

- Compilation target for common languages (e.g., C/C++, Rust, ...) and interpreters (e.g., Python)
- Light-weight isolation to Sandbox memory
- Performance 1.5-2x of native

- Compiler enforced memory safety
- Incremental adoption
  - Interfaces with legacy software (C/C++)
  - No VM/runtime
- Predictable Performance comparable to C/C++
MeSHwA Isolating Services

- Unifying abstraction across different languages and runtimes

- Restricting memory view
  - Object-granular languages vs. VM-based runtimes
  - Sharing across multiple memory-safe services

- Restricting execution targets
  - Limited targets within the service
  - Single exit acting as router across services
MeSHwA Software Runtime

- Specialized common services
- Discovery of common services
MeSHwA Hardware optimizations: Sharing

- Software-only drastically improves sharing, but in some cases still requires copying

- Language support for foreign types
  - Rust provides foreign function interface, Wasm provides interface types
  - Memory ownership unclear across services

- Capability-based Hardware
  - CHERI or Cryptographic Computing
  - Hardware pointer represents memory access privilege

Extend research in ownership across applications

Apply capability-based hardware to single address space
Conclusion: MeSHwA

- Serverless Microservice/FaaS development and deployment model demands SW/HW architecture improvement

- Recent advances in memory-safe languages and runtimes suggest stronger reliance on software

- MeSHwA argues for a single address space, memory-safe environment with optimized hardware
MeSHwA: The case for a Memory-Safe Software and Hardware Architecture for Serverless Computing

Anjo Vahldiek-Oberagner, Mona Vij (Intel Labs)
anjovahldiek@gmail.com, mona.vij@intel.com

Software-only
Hardware provides
compute only

MeSHwA

Software controls
computation only
Hardware controls