Advanced Operating Systems
EbbRT: A Framework for Building Per-Application Library Operating Systems

Presented by

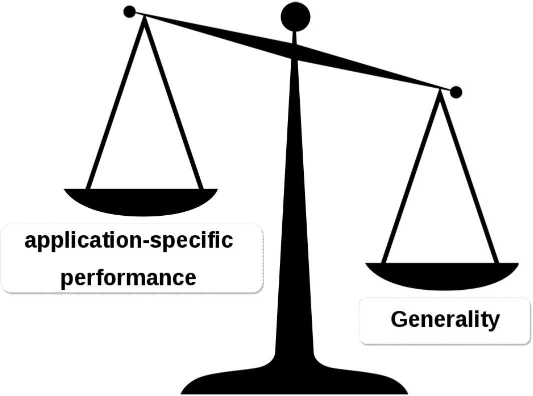
BARALLON Lucas and SID-LAKHDAR Riyane

(M2 MoSIG: ENSIMAG / UGA)

January 3, 2017

DEFINE THE PROBLEM

- **General purpose OS** ⇒ mechanisms for safe interaction between heterogeneous applications
- But this **”generality”** is a drawback for optimizing the performance of a specific application.



Existing solutions

- Different approaches to link the bridge between generality and custom performance
    1. Kernel bypass
    2. Hardware virtualization
    3. ...
- Limitation: all of them require a lot of engineering effort

THE EBBRT SOLUTION

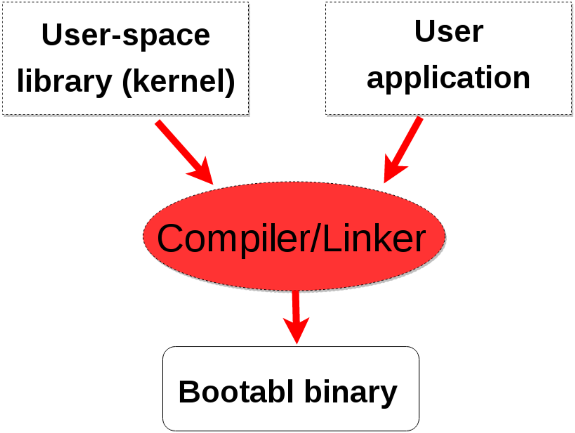
- The Elastic Building Block Runtime (EbbRT)
- Framework to build application-specific OS
- How does the built OS adapts to the specifications of a user application?
- How does it manages to be compatible with the main stream user application requirements?

Specification of the EbbRT framework and resulting OS

Custom OS to optimize the application-specific performance

A framework build for a large set of application

EbbRT: Building Library OS



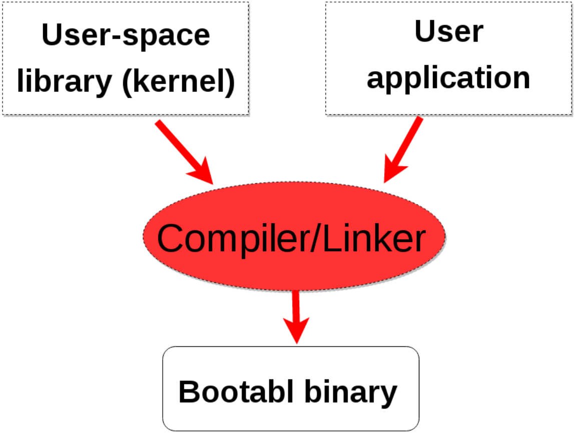
User-space library (kernel)

User application

Compiler/Linker

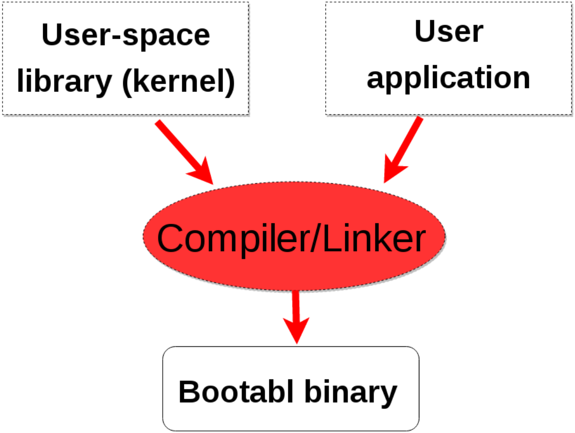
Bootabl binary

EbbRT: building library OS



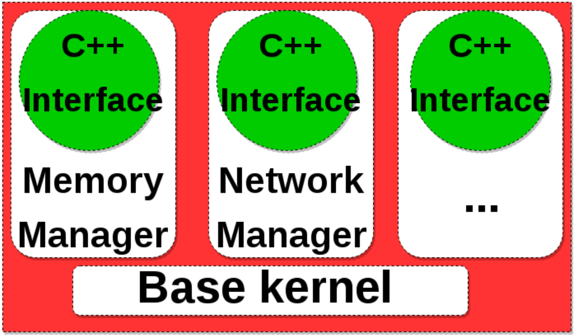
- Lightweight software stack
- Reduce engineering effort (reuse host os functionality).

EbbRT: building library OS



- Lightweight software stack
- Reduce engineering effort (reuse host os functionality).
- Unikernel OS (avoid generality mechanism performance footprint)

EbbRT: Software architecture



- Choose the implementation that most fits the application
- Allow adding custom implementations

EBBRT: THE ELASTIC BUILDING BLOCK

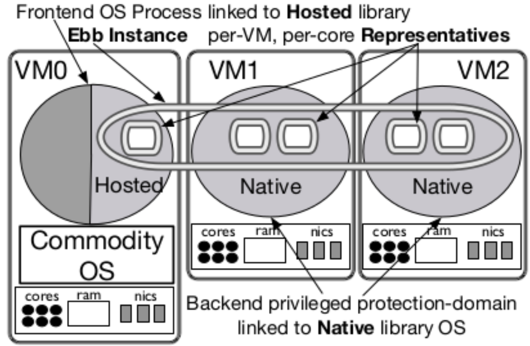

Frontend OS Process linked to **Hosted** library
Ebb Instance
per-VM, per-core **Representatives**
VM0
VM1
VM2
Hosted
Native
Native
Commodity OS
cores
ram
nics
Backend privileged protection-domain linked to **Native** library OS

Figure: High Level EbbRT architecture

- EbbRT applications are composed of Ebbs
- It provide a C++ class interface
- Encapsulate data and functionnalities
- Customizable by the user of the framework

EbbRT: Heterogenous structure

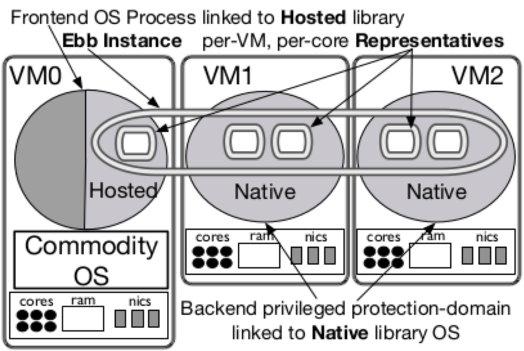

Frontend OS Process linked to **Hosted** library
Ebb Instance
per-VM, per-core **Representatives**
VM0
VM1
VM2
Hosted
Native
Native
Commodity OS
cores
ram
nics
cores
ram
nics
cores
ram
nics
Backend privileged protection-domain linked to **Native** library OS

Figure: High Level EbbRT architecture

- Native Runtime : part of the OS
- Hosted Runtime : user-level library

EBBRT: EVENT ORIENTED PROGRAMMING

- non preemptive event model
- Cooperative threading

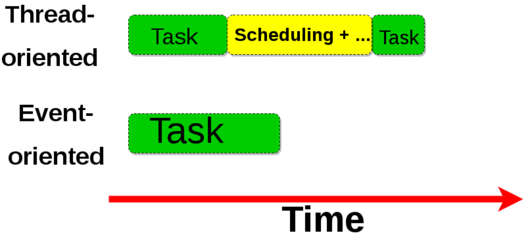
Specification of the EbbRT framework and resulting OS

Custom OS to optimize the application-specific performance

A framework build for a large set of application

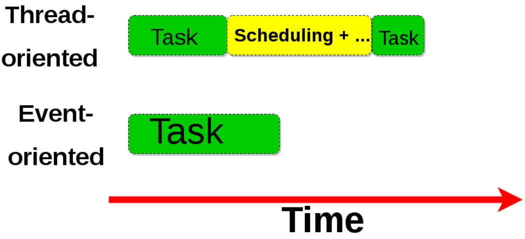
Reduce the OS footprint

- **Non-preemptive environment** (event-driven)



REDUCE THE OS FOOTPRINT

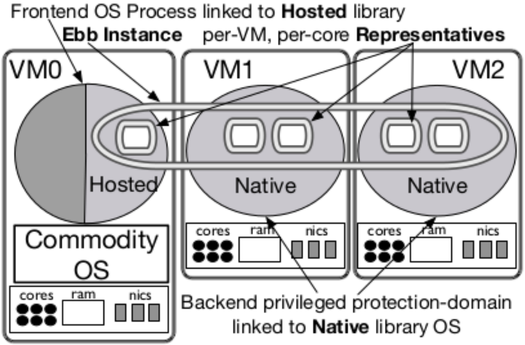
- **Non-preemptive environment** (event-driven)



- **Unikernel architecture:** Remove address translation footprint (time/space)

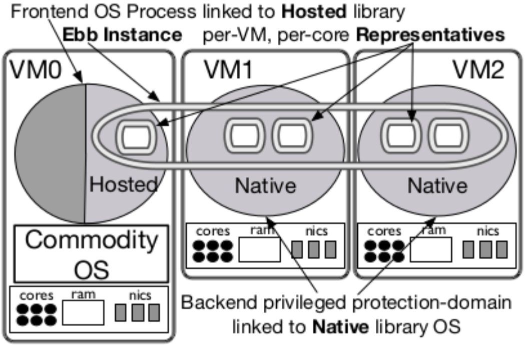
Distributed user app with lightweight communication

- **Distributed OS architecture**


Frontend OS Process linked to **Hosted** library
Ebb Instance
per-VM, per-core **Representatives**
VM0
VM1
VM2
Hosted
Native
Native
Commodity OS
cores
ram
nics
cores
ram
nics
cores
ram
nics
Backend privileged protection-domain linked to **Native** library OS

DISTRIBUTED USER APP WITH LIGHTWEIGHT COMMUNICATION

- **Distributed OS architecture**


Frontend OS Process linked to **Hosted** library
Ebb Instance
per-VM, per-core **Representatives**
VM0
VM1
VM2
Hosted
Native
Native
Commodity OS
cores
ram
nics
cores
ram
nics
cores
ram
nics
Backend privileged protection-domain linked to **Native** library OS

- Single application distributed on different hardware platforms
- Minimal distribution overhead (shared address space)

Specification of the EbbRT framework and resulting OS

Custom OS to optimize the application-specific performance

A framework build for a large set of application

HETEROGENEOUS STRUCTURE : DETAILS

- Native runtime :
  - Dont bother about interface and protection mechanism.
  - Single address space and provide functionalities like timers and memory allocation.
- Hosted runtime :
  - Decrease the interface compatibility requirement
  - Part of the work offload to the library
- Elastic Building Block :
  - Component of EbbRT application
  - Provide an interface from a standard C++ class

Event-driven execution Model : Details

- Non-preemptive environment :
  - Low overhead abstraction
  - Map directly to device interrupt
  - Avoid the cost of scheduling
- Cooperative Threading
  - Event explicitly save and restore control state
  - Facilitate the implementation of library using blocking system calls

Performance

- Different evaluation about memory allocation or network performance
- Benchmark with Memcached
- Node.js :
  - Javascript benchmark
  - webserver workload
- EbbRT provide better result in all the test

Conclusion

- Performance specialization
- Broad applicability
- Ease of development

References

Wikipedia: Unikernel, os library.
https://en.wikipedia.org/wiki/Unikernel#Library_operating_systems.

D. Schatzberg, J. Appavoo, J. Cadden, and O. Krieger.
Multilibos: An os architecture for cloud computing.
Technical report, Computer Science Department, Boston University, 2012.

D. Schatzberg, J. Cadden, H. Dong, O. Krieger, and J. Appavoo.
Ebbrt: A framework for building per-application library operating systems.
2016.