Summary

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

Dan Schatzberg, James Cadden, Han Dong, Orran, Krieger, Jonathan Appavoo
Boston University

Keywords

- Application Specific Operating Systems
- Library OS
- Elastic Building Blocks
- Per-Application Performance
- Portability
- Security
- Non-Preemptive Scheduling
- Event Driven Model
- Futures and Lambdas
- Cooperative Threading
- Read-Copy-Update

Motivation

- General purpose Operationg systems are many times not useful when we need to carryout specific tasks only. Because they sacrifice performance in order to preserve generality which is not needed in this case.
- Present day Operating Systems are insufficient because they do not give us enough flexibility over hardware like using particular core, or particular scheduling strategy, directly controlling hardware devices etc.
- For this reason, application specific Operating Systems are created, which run trusted specific applications only and provide only the services needed by these applications. They provide much higher performance in scenarios they are used.
- But a substantial effort is required to construct these Operating Systems.
- Library Operating Systems are one type of application specific OS. They link to applications like a library and we can make functions calls instead of system calls and hence they have potential to give much higher performance.

Related Work

- Library Operating System structure was introduced by The Exokernel. It linked system functionality directly into application space and ran them in same protection domain.
- Library Operating Systems are shown to provide many useful properties like Portability, Security and Efficiency.
- OSv is a Library Operating System which is compatible with Linux and provides application portability whereas Mirage constructs minimal OS for security. Unlike these, EbbRT supports source level portability through language features.
- Many other works like CNK, Arraks and IX can be divided as targeting a narrow set of applications or a wide set of applications. But EbbRT provides a framework for construction of application specific operating systems.
- EbbRT differs from Choices and OSKit, other operating system frameworks in its performance objective and customisability of system software.

EbbRT Goals

EbbRT framework has three objectives:

- Allowing applications to customize the system at every level in order to achieve high performance.
- To be able to support many existing libraries and runtimes on which applications depend in order to make sure that framework has high utility.
- To reduce the effort required to construct and maintain Library Operating Systems.

EbbRT Overview

- Elastic Building Block Runtime (EbbRT) is a framework for building per application framework.
- EbbRT is comprised of Ebbs which can be reused, discarded, replaced or extended according to the need of application that is to be deployed. Also its execution environment allows applications to interact directly with hardware resources. Each Ebb has a unique EbbId
- Ebb and Processes differ in the respect that Ebb are sitting on top of even lower level of abstraction than the processes and have clearer view of hardware. Ebb is a library and spans multiple hardwares and this gets linked to processes.
- To reduce application porting effort, it supports both general purpose and specialized operating systems, in order to allow functionality to be offloaded.

- EbbRT provides heterogeneous Distributed Structure in the way that it is implemented as a bootable runtime (native) through which application can interact with bare hardware and also as hosted library through which legacy systems can be supported. This way same application can be deployed across several machines. Also the Ebb design make this framework modular.
- Execution in EbbRT is non-preemptive and event-driven. This model provides low overhead abstraction over the hardware as application can have registered handlers which map directly to device interrupts. In hosted library this behaviour is provided by event loops.
- Simple Cooperative threading model is also provided on top of events in order to support many applications for which event driven programming is not a good choice.
- Event driven programming model obfuscates control flow of a program. To mitigate this Futures and Lambdas are used. Futures is a data structure whose values are calculated asynchronously. Futures must be operated upon using THEN methods. This allows other functions to be invoked when a value has been calculated and hence it structures the code flow.
- To evaluate EbbRT, its memory allocation was compared against glibc and jemalloc and EbbRT performed better than glibc and comparably against jemalloc.
- Similarly Network Stack ported on EbbRT achieved better Goodput than Linux Network Stack for all packet sizes.
- A distributed memory caching tool - Memcached was reimplemented for EbbRT and its performance was compared against its implementation in OSv and Linux. For any latency, throughput for EbbRT implementation of memcached was better than others.

Check Your Understanding

- What are Fragmented objects and how are they different from replica?
- Describe Read-Copy-Update.
- How EbbRT allows applications to run on heterogeneous environment in Cloud?
- How does EbbRT design provides better Security?
- “Due to lack of pre-emption, most allocations can be serviced from a per-core cache without any synchronization”. How?