Quo vadis Linux?!?

The rise of new cloud O/S

Dr. Udo Seidel

Agenda

- Introduction
- The players
- And?
- Summary

Me :-)

- Teacher of mathematics and physics
- PhD in experimental physics
- Started with Linux in 1996
- With Amadeus since 2006
- Before:
  - Linux/UNIX trainer
  - Solution Engineer in HPC and CAx environment
- Now: Architecture & Technical Governance

Introduction

Cloud and Cloud O/S

- Multiple definitions
- Cloud → here:
  - IaaS type
  - Scale out not up
  - API driven
- Cloud O/S → here:
  - Date Centre, not Desktop
  - Layer between 'hardware' and 'service'

Cloud and Opensource

- ATM: Highly tight together
- Foundation and framework around
- New Business models
- Just the start … not the end

Opensource Cloud O/S

- Meeting previous requirements
  - Cloud
  - Cloud O/S
  - Source code available
- Cloud 2.0

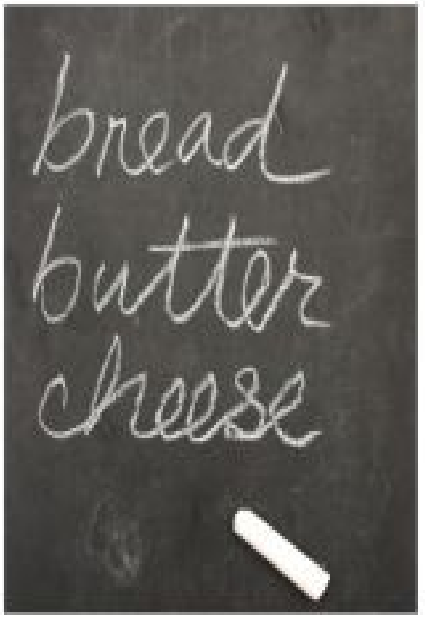
The players

Selection

- Wide range of options/technologies/...
- Here:
  - Enterprise Linux' metric
  - Traditional approach to Linux
- Three candidates

Approach

- History & facts
- Kernel (space)
- User-space
- Discussion
  - First thoughts
  - Second thoughts :-)
  - Maybe call to action



CoreOS

GUUG FFG FEB 2016

History & Facts

- Initial release: October 2013
- License: Apache 2.0
- Kernel: Monolithic
- Language: C/C++
- Platforms:
  - x86_64: physical, virtual
  - ARM64: to come (unofficially images available)

Kernel

- Vanilla based
- Lighter than Enterprise
- Quite recent

User space

- No package management
  - Shadow /usr file system
  - Boot entries
- Small process footprint
- File system
  - Read-only /usr
  - Symlinks in /etc to /usr
- Containerize everything

More on user space

- *etcd & fleet* … and more
- Configuration
  - cloud-config concept
  - Part of boot process
- User:
  - Less is more
  - Authentication via ssh key only
  - Configuration read-only

About *etcd*

- Written in Go
- Apache license 2.0
- Distributed key value store
- Cluster frame-work
- API: HTTP and JSON .. *etcdctl*
- Service/application discovery

About *fleet*

- Written in Go
- Apache license 2.0
- Distributed init system
  - Foundation: *systemd*
  - Uses *etcd*
- API: *fleetctl* …. *ssh*
- Topology: CoreOS instance meta data

Service/System/Application

- Business as usual
  - Language
  - Environment
  - Process
- See later ;-)

Who is using CoreOS

- AeroFS
- Auth0
- Engine Yard
- Shoppify
- Several Cloud/hosting providers

OSv

GUUG FFG FEB 2016

History & Facts

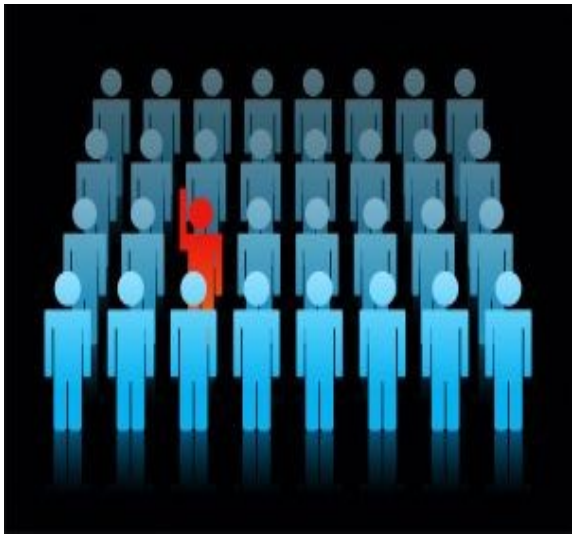
- Initial release: September 2013
- License: BSD-3
- Kernel: Monolithic
- Language: C++
- Platforms:
  - x86_64: virtual only
  - ARM64: (still) to come

Kernel

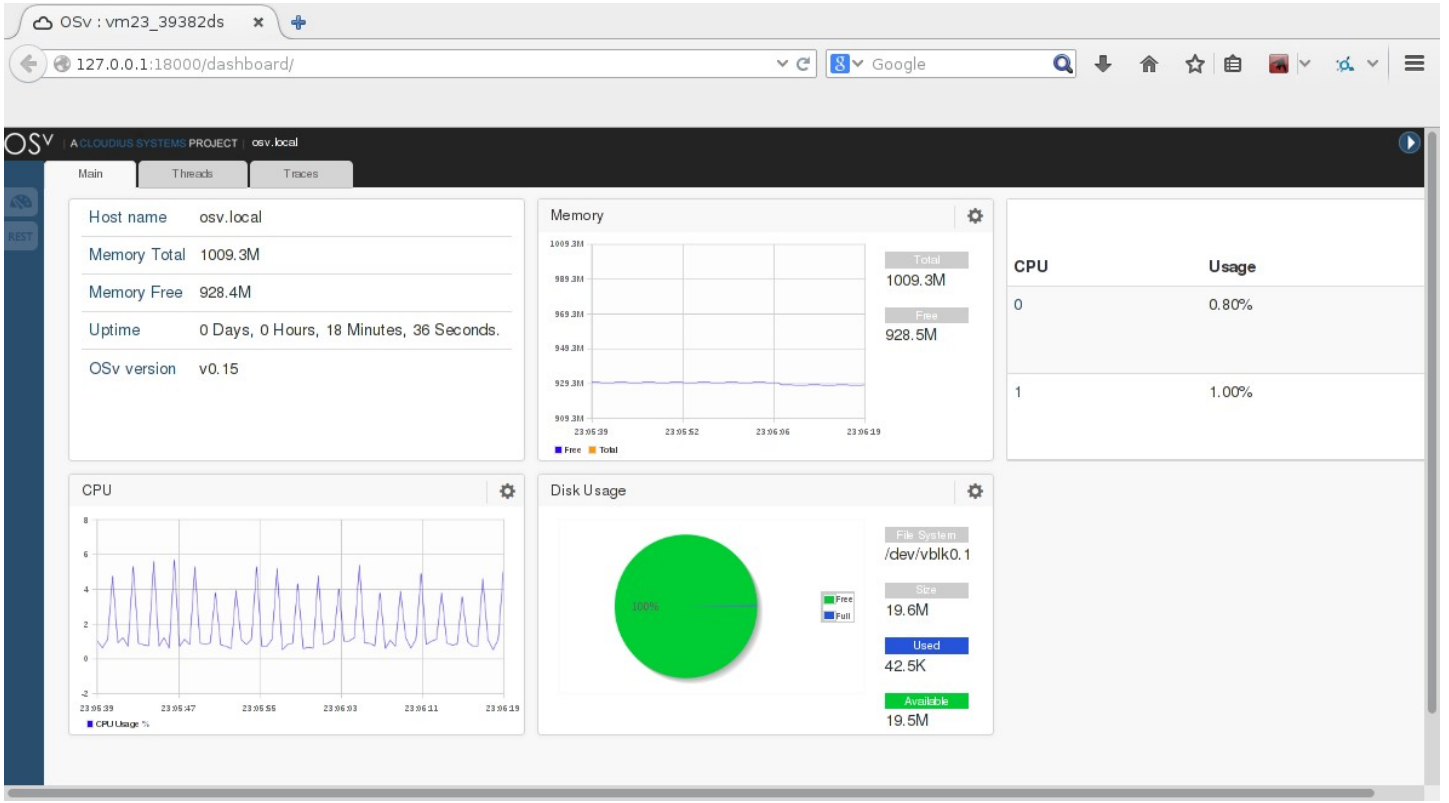
- Newly developed
- No physical hardware support
- Smallest possible layer
  - Run on hypervisor
  - Host application
  - E. g. no spinlocks
- Linux-ABIs in place

User space

- Not existing!
- No user concept
- Single process only
  - Threads
  - Relocatable object code application
  - Missing: *fork(), vfork(), clone()*



REST API/Dashboard



Service/System/Application

- Source code
- Build environment
- OSv integration
  - Locally
    - *capstan → Capstanfile*
    - Download, build, run
  - Cloud-init

'Hello world'

udo@tron:~/capstan-ffg

```
$ ls
Capstanfile  hello.cc  Makefile
$
$ cat hello.cc
/*
 * Copyright (C) 2014 Cloudius Systems, Ltd.
 *
 * This work is open source software, licensed under the terms of the
 * BSD license as described in the LICENSE file in the top-level directory.
 */

#include <iostream>

int main()
{
    std::cout << "Hallo FFG 2016!" << std::endl;
}
$
$ make
  CXX hello.o
  LINK hello.so
$
$ ~/bin/capstan run
Building capstan-ffg...
Uploading files...
1 / 1 [================================================================================] 100.00 %
Created instance: capstan-ffg
OSv v0.16
eth0: 192.168.122.15
Hallo FFG 2016!

$
```

Images Market Place

- Java (openJDK)
- Tomcat
- Memcached
- Reddis
- Cassandra
- Kafka
- Netperf
- … DIY

Who is using OSv?

- Cloudius Systems
  - CloudRouter
  - Opendaylight
- Research projects
- ???

MirageOS

History & Facts

- Initial release: December 2013
- License: ISC
- Kernel: N/A (Unikernel)
- Language: OCamel/OPAM
- Platforms:
  - x86_64: virtual (Xen)
  - ARMv7+: virtual (Xen)

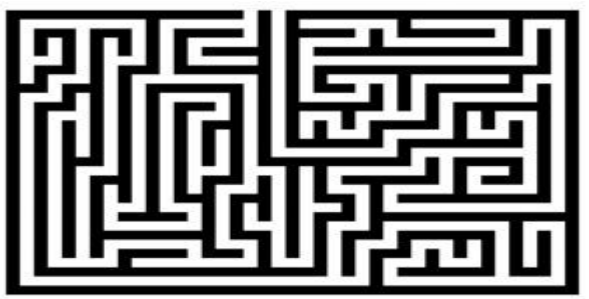
Kernel

- N/A!
  - Library Operating System
  - Unikernel/targets
  - DIY for almost everything
- Written in OCamel
- Functions via libraries

User space

- Not existing
- Actually …

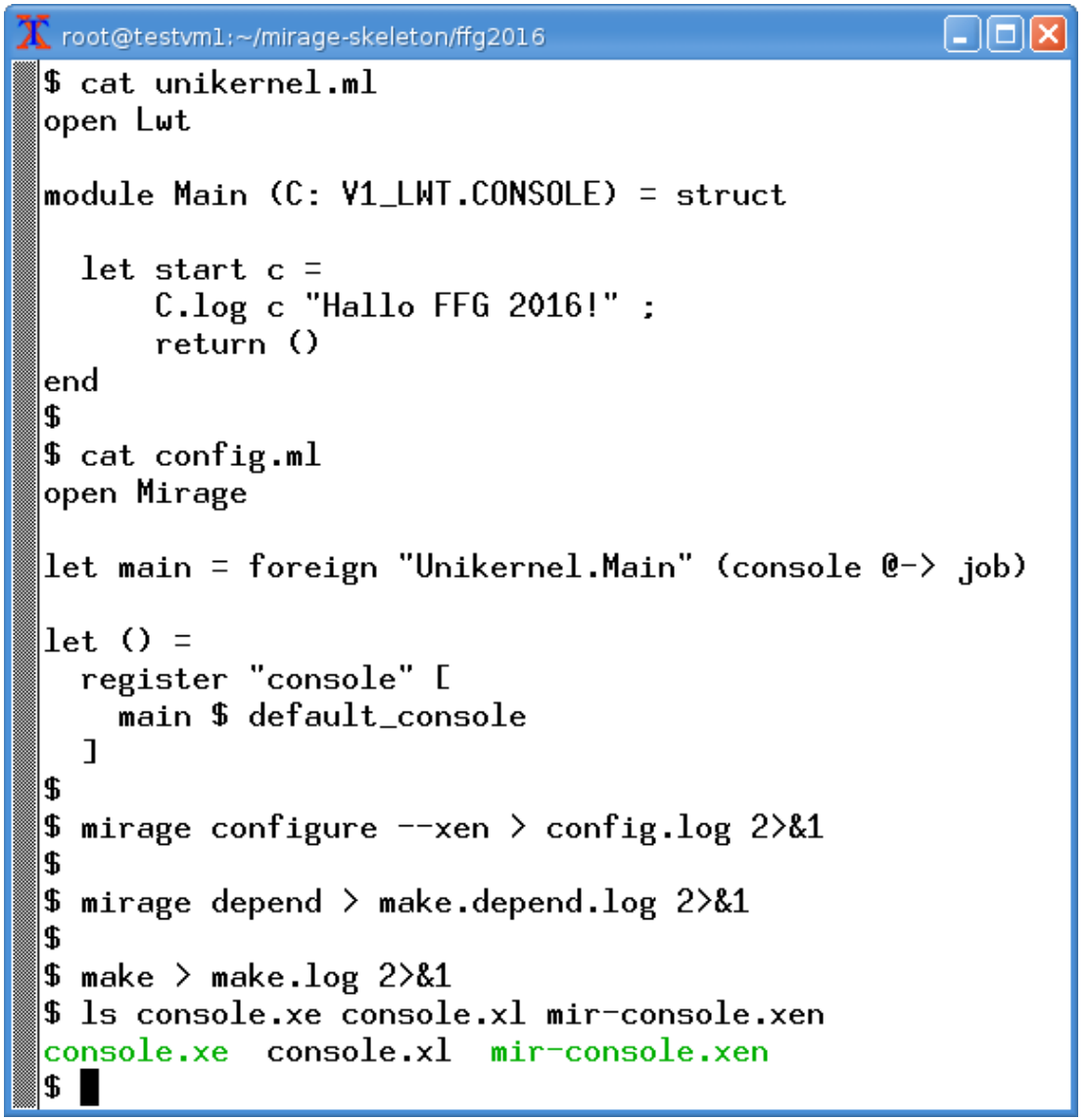
  … just the O/S Kernel



Service/System/Application

- Lot of (pre-)thinking
  - Application
  - Kernel
- OCamel and OPAM
  - *unikernel.ml*
- Mirage Integration
  - *config.ml*
  - *mirage ….*

'Hello world'


root@testvm1:~/mirage-skeleton/ffg2016
$ cat unikernel.ml
open Lwt
module Main (C: V1_LWT.CONSOLE) = struct
let start c =
C.log c "Hallo FFG 2016!" ;
return ()
end
$
$ cat config.ml
open Mirage
let main = foreign "Unikernel.Main" (console @-> job)
let () =
register "console" [
main $ default_console
]
$
$ mirage configure --xen > config.log 2>&1
$
$ mirage depend > make.depend.log 2>&1
$
$ make > make.log 2>&1
$ ls console.xe console.xl mir-console.xen
console.xe console.xl mir-console.xen
$

'Kernel Libraries'

- Core
- Storage, e.g.
  - Block device
  - File system
- Network, e.g.
  - TCP/IP
  - HTTP
- Formats, e.g
  - JSON

Known/tested Use Case

- Webserver
- DNS
- Openflow Controller

Who is using MirageOS?

- Research projects
- ???

Other

More and more and more

- CirrOS
- JeOS
- Atomic, Snapper, Photon, …
- DCOS

  ...
- SmartOS
- ZeroVM

And?

General Thoughts

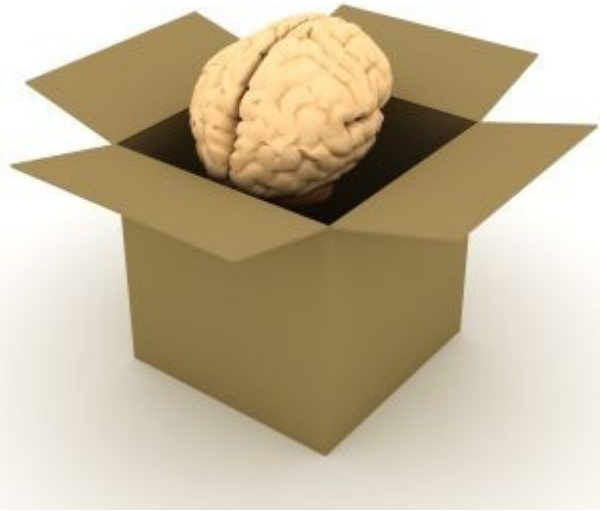
- Where and which?
- # of future paths
- Customer and business

First Thoughts on CoreOS

- Least number/size changes
- 'Traditional' approach possible
  - Technically
  - Non-technically
- Check
  - Prerequisites
  - APIs

Embrace CoreOS Features

- Review/rethink
  - User/Security
  - Application roll-out/-back
  - Data store
- Framework around
  - Systemd/Container
  - Service discovery



First Thoughts on OSv

- More adaptation needed
- ~~'Traditional' approach~~
- Potential for code re-use

Embrace OSv

- Single application!
- ELF shared object
- RUN in Kernel space
- Short/medium runtime
- Roll-out/-back on OSv instance level
- Data management

First Thoughts on MirageOS

- Biggest changes
- ~~'Traditional' approach~~
- ~~Code re-use~~
- Almost nothing known left

Embrace MirageOS

- Review Kernel needs
- Review coding language
- Test with UNIX target
- Application = Kernel
- Data Management

Summary

First 'Last' Thoughts

- Ufff!! ...Crystal ball?!?
- Potentially lot of work
  - Technically
  - Mindset/paradigm change
- Customer and business

Take Aways

- Focus shift
- At least reduction of O/S user-land
- Further paradigm shifts to come
- Today: multiple options
- Open your mind :-)

References

- http://coreos.com
- http://osv.io
- http://openmirage.org

Thank you!

Quo vadis Linux ?!?

The rise of new cloud O/S

Dr. Udo Seidel