cesnet

POUŽITÍ CONTAINERŮ V METACENTRU

aneb Singularity

Miroslav Ruda

CESNET

Seminář MetaCentra, Praha

11.5. 2018

Obsah

- Virtualizace a containery obecně
- Containery pro HPC - Singularity
- Příklady
- Použití v MetaCentru
- Docker

Containery v HPC I

Motivace - ne vždy vyhovuje prostředí MetaCentra

- software vyžadující specifický systém, knihovny
- předpřipravené prostředí, vlastní nebo projektové
- důraz na reprodukovatelnost

Dlouhodobě používaná je virtualizace celého počítače

- VMWare, VirtualBox, Xen, KVM
- nyní používanější, proto se s ním na začátku trochu srovnáme
- srovnání je o efektivitě, ztrátě výkonu, bezpečnosti, izolovatelnosti, přenositelnosti

Containery v HPC II

Virtualizace hardware - iluze celého fyzického počítače

- na něm běží kompletní operační systém
- umožňuje jeden fyzický server rozdělit na několik virtuálních
- základní technologie pro cloudy, softwarově definovaná datacentra

Containery - zabalení/iluze vlastního systémového prostředí potřebného pro konkrétní aplikaci

- jeden společný operační systém, ten spravuje containery, HW, přístup k datům
- plus nástroje pro práci s containery
- container jde sdílet - neobsahuje žádná data!

Containery v HPC III

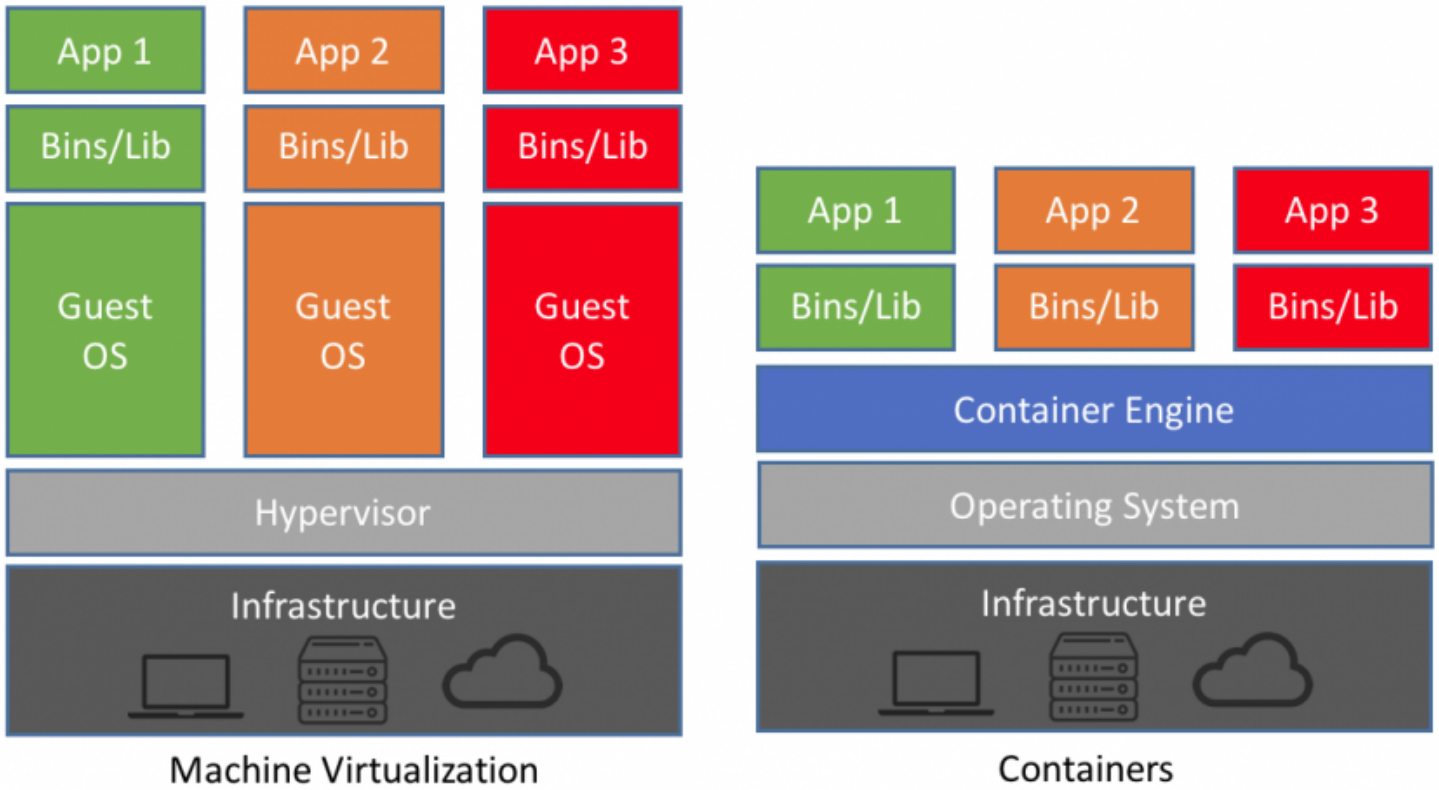
Historická vsuvka - hardwarová virtualizace

- 1960's IBM S/360 Mainframes
- 1999 VMWare "VMWare Workstation"
- aplikační vituální stroj - 1996 Sun Java
- opensource - Xen 2003, KVM od 2007

Historická vsuvka - containery

- 1979 UNIX chroot (added to BSD in 1982)
- 2001 Linux VServer (VPS Solution)
- 2005 OpenVZ
- 2008 Control Groups (cgroups v jádře Linuxu)
- 2008 LXC (LinuX Containers)
- 2013 Docker

Containery


App 1
App 2
App 3
Bins/Lib
Bins/Lib
Bins/Lib
Guest OS
Guest OS
Guest OS
Hypervisor
Infrastructure
Machine Virtualization
App 1
App 2
App 3
Bins/Lib
Bins/Lib
Bins/Lib
Container Engine
Operating System
Infrastructure
Containers

Containery v Linux

- Linux kernel Cgroups
  - skupinu procesů
  - jde přiřadit (omezené) zdroje - CPU, paměť, disk
- Linux kernel Namespaces
  - omezení viditelnosti systému
  - Mounts, UTS, uname, IPC, PID, Networks, User
- obraz disku containeru
  - iluze disku s filesystémem, kde je instalovaný systém, knihovny i aplikace
  - často jen tar soubor
  - dovoluje hierarchie, vrstvy, nejvyšší je zapisovatelná
- Docker, Singularity, ale i LXC, uDocker, gVisor …
- rozdíly - efektivita, výkon, bezpečnost

Docker

- nejpoužívanější linuxové řešení containerů
- poskytuje nástroje pro vytvoření, správu a distribuci containerových obrazů
- primárně určeno pro virtualizaci síťových služeb
- je výhodný pro vývojáře, usnadňuje přenositelnost, reprodukovatelnost instalace
- není až tak vhodný pro HPC výpočty, protože se snaží nabízet až moc vlastností virtuálních strojů
  - přístup k rootovským oprávněním, proto vyžaduje rootovské oprávnění v systému
  - proto isolace sítě pomocí virtualizace (výkon!)
  - implementace dělá velké problémy paralelním prostředím ala MPI, GPU kartami
  - nesnadná spolupráce s autory na vylepšeních pro HPC

Singularity I

- containery určené pro náročné výpočty
- pracuje s obrazem containeru v souboru
  - portabilita - ne přenositelnost zdrojového kódu, ale distribuování odladěného prostředí ve formě virtuálního stroje/disku z počítače
- určeno pro prostředí, kde uživatelé nemají rootovské oprávnění, proto ho ani neemuluje a tím snižuje bezpečnostní problémy
- funguje s MPI úlohami
- velký důraz na reprodukovatelnost
- snaha využití nástrojů pro Docker, minimální závislost na vlastnostech v linuxovém jádře

Singularity II

Využití - distribuce správně nainstalovaného software

- BioContainers (https://biocontainers.pro/)
- CMS, velká superpočítačová centra v USA

Bezpečnost

- dynamický a živelný vývoj, velmi časté chyby
  - EGI-SVG-2018-14213 (2018-03-29/2018-04-11)
  - Local privilege escalation via overlayfs
  - EGI-SVG-2018-14311 (2018-04-30)
  - Local privilege escalation (without overlay support)
  - nové verze v pátek odpoledne, z logů je zřejmý bezpečnostní problém
- nové verze zároveň mění chování příkazů

Jak v MetaCentru?

Příprava obrazu vyžaduje rootovské oprávnění

- Singularity na vlastním stroji nebo v cloudu
- konverze z Docker obrazu
- sandbox a docker://metacentrum/ubuntu-fakeroot

Připravený obraz uložený v /home nebo /projekt

- interaktivně nebo přes PBSPro
- funguje MPI, kerberos, /software

Očekávané využití uživateli MetaCentra

- vlastní software se specifickým prostředím
- přebírání předpřipravených obrazů containerů
- nástroj pro zlepšení reprodukovatelnosti


cesnet
Prostor na dotazy, následují příklady

Singularity workflow

Příprava containeru s rootovským oprávněním

- "když je potřeba root v containeru, musím být root i mimo container"
- vytvoření nového containeru
- bootstrap/install container
- modifikace containeru

Využití containeru uživatelem

- singularity shell
- singularity exec
- singularity run

Singularity příklady

```
$ singularity shell docker://ubuntu:latest
Docker image path: index.docker.io/library/ubuntu:latest
....
Singularity ubuntu:latest:~> cat /etc/lsb-release
DISTRIB_ID=Ubuntu
DISTRIB_RELEASE=18.04
DISTRIB_CODENAME=bionic
DISTRIB_DESCRIPTION="Ubuntu 18.04 LTS"
Singularity ubuntu:latest:~>
```

Singularity příklady

```
odin$ cat examples/debian/Singularity
BootStrap: debootstrap
OSVersion: stable
MirrorURL: http://ftp.us.debian.org/debian/

%runscript
    echo "This is what happens when you run the contai

%post
    echo "Hello from inside the container"
    apt-get update
    apt-get -y install vim
    apt-get clean
```

Singularity příklady

```
odin$ sudo singularity create debian.img
Creating empty 768MiB image file: debian.img
Formatting image with ext3 file system
Image is done: debian.img
odin$ sudo singularity bootstrap debian.img examples/debian/Sin
Building into existing container: debian.img
...
Singularity container built: debian.img
Cleaning up...
odin$ singularity shell debian.img
Singularity: Invoking an interactive shell within container...

Singularity debian.img:~/> ls
debian.img  singularity-2.5.1  singularity-2.5.1.tar.gz
Singularity debian.img:~/> exit
```

Singularity příklady

Využití namespaces

```
odin$ singularity exec -p debian.img ps axu
USER       PID %CPU %MEM    VSZ   RSS TTY      STAT START   TIME COMMAND
ruda         1  0.0  0.0  28196  2556 pts/4    R+   15:03   0:00 ps axu
odin$ singularity exec debian.img ps axu
USER       PID %CPU %MEM    VSZ   RSS TTY      STAT START   TIME COMMAND
root         1  0.0  0.0 185308  5740 ?        Ss   Apr27   0:56 /usr/lib/syst
root         2  0.0  0.0      0     0 ?        S    Apr27   0:00 [kthreadd]
root         3  0.0  0.0      0     0 ?        S    Apr27   0:18 [ksoftirqd/0]
root         5  0.0  0.0      0     0 ?        S<   Apr27   0:00 [kworker/0:0H
root         7  0.0  0.0      0     0 ?        D    Apr27   7:46 [rcu_sched]
root         8  0.0  0.0      0     0 ?        S    Apr27   0:00 [rcu_bh]
root         9  0.0  0.0      0     0 ?        S    Apr27   0:00 [migration/0]
root        10  0.0  0.0      0     0 ?        S    Apr27   0:03 [watchdog/0]
```

Singularity příklady

Modifikace obrazu

```
odin$ singularity exec debian.img python --version
/.singularity.d/actions/exec: 9: exec: python: not found
odin$ sudo singularity exec --writable debian.img apt-get -y in
Reading package lists... Done
...
Setting up python (2.7.13-2) ...
odin$ singularity exec debian.img python --version
Python 2.7.13
```

Singularity příklady

```
odin$ cat hello.py
#!/usr/bin/python
import sys
print("Hello World: The Python version is %s.%s.%s" %
odin$ ./hello.py
Hello World: The Python version is 3.4.6
odin$ singularity exec debian.img ./hello.py
Hello World: The Python version is 2.7.13
odin$ cat hello.py |singularity exec debian.img pythor
Hello World: The Python version is 2.7.13
odin$
```

Singularity v MetaCentru

```
odin$ module add intelcdk-17.1
odin$ singularity shell -B /software debian.img
Singularity: Invoking an interactive shell within container...

Singularity debian.img:~> /software/intelcdk-17.1/bin/icc
icc: command line error: no files specified; for help type "icc
Singularity debian.img:~>
```

Singularity v MetaCentru

```
$ qsub -l select=1 -l walltime=24:00:00 -- \\
        /usr/bin/singularity exec debian.img \\
        bash -c "/path/to/script.sh"

$ cat mpi_job.sh
#!/bin/bash
#PBS -l select=2:ncpus=2:mem=1gb:scratch_shared=4gb
#PBS -l walltime=04:00:00
#PBS -l place=scatter

module add openmpi-2.0.1-gcc
cat $PBS_NODEFILE |uniq >nodes.txt
mpirun -n 2 --hostfile nodes.txt singularity exec \\
    debian.img /path/to/program
```

cesnet

Děkuji za pozornost

http://www.metacentrum.cz
https://wiki.metacentrum.cz/wiki/Singularity
http://singularity.lbl.gov/user-guide

Singularity v MetaCentru

```
odin$ singularity build --sandbox ubuntu \\
                        docker://metacentrum/ubuntu-fakeroot
odin$ singularity shell -w ubuntu
Singularity: Invoking an interactive shell within container...

Singularity ubuntu:~/soft-local/singularity> fakeroot
odin$ apt-get update -qqq
odin$ apt-get install python
...
odin$ exit
Singularity ubuntu:~/soft-local/singularity> exit
odin$ singularity build ubuntu.simg ubuntu
odin$ singularity exec ubuntu.simg python --version
Python 2.7.15rc1
odin$ cat hello.py |singularity exec ubuntu.simg python
Hello World: The Python version is 2.7.15
```

metacentrum cesnet

Docker v MetaCentru

- Docker v cloudovém prostředí (pro provoz služeb)
- klasický Ubuntu/Centos/Debian obraz
- doinstalovat Docker/Singularity
- ručně nebo automatizovaně pustit docker
- existují i minimalizované instalace, které Docker rovnou pustí (boot2docker)
- OpenStack má i možnost, ze Docker se používá jako virtualizační nástroj místo KVM
  - zatím s tím nemáme žádné velké plány
  - containery by šlo spravovat nástroji OpenStacku
- pro správu doporučujeme docker-host nebo Kubernetes
- pro zájemce máme i příklady z EGI /EOSC prostředí, kde používají i orchestraci přes více cloudových poskytovatelů

cesnet

Děkuji za pozornost

http://www.metacentrum.cz
https://wiki.metacentrum.cz/wiki/Singularity
http://singularity.lbl.gov/user-guide