Linux Kernel in Docker Containers
Leon Romanovsky
August, 2019
## Short Bio

<table>
<thead>
<tr>
<th>Role</th>
<th>Leon Romanovsky</th>
<th>Jason Gunthorpe</th>
</tr>
</thead>
<tbody>
<tr>
<td>RDMA kernel maintainer</td>
<td>Internal (Mellanox)</td>
<td>External (upstream)</td>
</tr>
<tr>
<td>Unified RDMA user space library maintainer</td>
<td>External (upstream, co-founder)</td>
<td>External (upstream, founder)</td>
</tr>
<tr>
<td>Linux experience</td>
<td>&gt; 20 years</td>
<td>&gt; 20 years</td>
</tr>
</tbody>
</table>
## Short Bio

<table>
<thead>
<tr>
<th></th>
<th>Leon Romanovsky</th>
<th>Jason Gunthorpe</th>
</tr>
</thead>
<tbody>
<tr>
<td>RDMA kernel maintainer</td>
<td>Internal (Mellanox)</td>
<td>External (upstream)</td>
</tr>
<tr>
<td>Unified RDMA user space</td>
<td>External (upstream, co-</td>
<td>External (upstream, founder)</td>
</tr>
<tr>
<td>library maintainer</td>
<td>founder)</td>
<td></td>
</tr>
<tr>
<td>Linux experience</td>
<td>&gt; 20 years</td>
<td>&gt; 20 years</td>
</tr>
</tbody>
</table>

### Patch statistics in 2018

<table>
<thead>
<tr>
<th></th>
<th>Leon Romanovsky</th>
<th>Jason Gunthorpe</th>
</tr>
</thead>
<tbody>
<tr>
<td>Authored kernel patches</td>
<td>166</td>
<td>200</td>
</tr>
<tr>
<td>Authored rdma-core patches</td>
<td>60</td>
<td>220</td>
</tr>
<tr>
<td>Authored iproute2 patches</td>
<td>23</td>
<td>---</td>
</tr>
<tr>
<td>Handled (reviewed and successfully</td>
<td>713</td>
<td>1164</td>
</tr>
<tr>
<td>submitted) kernel patches</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Perfect Solution

- Hide operating system complexity from kernel and QEMU developers
- Give latest development and run environments
- Seamless integration with emulated and real hardware
- Run real operating system and real kernel
- Fast write-build-test loop
- Source and patch oriented flow
- Built-in continuous integration
- Work anywhere
- Provide out-of-the box experience
- Easy customization
- Ready for cloud orchestration software
Perfect Solution

- **Hide** operating system complexity from kernel and QEMU developers
- **Give latest** development and run environments
- **Seamless** integration with emulated and real hardware
- Run **real** operating system and **real** kernel
- **Fast** write-build-test loop
- **Source** and **patch** oriented flow
- Built-in **continuous integration**
- Work **anywhere**
- Provide **out-of-the box** experience
- **Easy** customization
- Ready for **cloud** orchestration software
Development Flow

Ideal

Write Build Test
Development Flow

Ideal

Write → Build → Test

Reality

Write

Test ↔ Build
Existing Solutions

- Plenty of docker builders
- **virtme**
  - Relies on running kernel environment
  - Based on busybox and not on real OS
  - [https://git.kernel.org/pub/scm/utils/kernel/virtme/virtme.git/](https://git.kernel.org/pub/scm/utils/kernel/virtme/virtme.git/)
- **docker-qemu**
  - Run full VM
  - [https://github.com/Ulexus/docker-qemu](https://github.com/Ulexus/docker-qemu)
- ....
  - Don’t have any option to use compiled kernel
github.com/mellanox/mkt

**Note**: It contains Mellanox specific code
Layers

Hypervisor
• source code
• build artefacts
• logs

Runners
• build
• ci
• run
• images

Containers
• qemu runner
• builders support
Initial Setup

Pre-requirements
- Modern distribution, tested on Fedora 26, Ubuntu 16.04 and RedHat 8
- Python 3.5 or above
Initial Setup

- Pre-requirements
  - Modern distribution, tested on Fedora 26, Ubuntu 16.04 and RedHat 8
  - Python 3.5 or above
- Download and setup MKT to be in the PATH
  - cd
  - git clone https://github.com/Mellanox/mkt.git
  - mkdir ~/bin
  - ln -s $HOME/mkt/mkt ~/bin/
  - export PATH=$HOME/bin:$PATH
Initial Setup

- Pre-requirements
  - Modern distribution, tested on Fedora 26, Ubuntu 16.04 and RedHat 8
  - Python 3.5 or above
- Download and setup MKT to be in the PATH
  - cd
  - git clone https://github.com/Mellanox/mkt.git
  - mkdir ~/bin
  - ln -s $HOME/mkt/mkt ~/bin/
  - export PATH=$HOME/bin:$PATH
- Install Docker CE, git and bring source code from gerrit
  - mkt setup
  - mkt setup-master – for multi-machine setups
  - mkt setup-slave MASTER_IP – for multi-machine setups
Support Container

- Has all dependencies to build
  - Kernel
  - QEMU
  - rdma-core
  - iproute2
- In use for
  - Source code build
  - Local CI
  - Local KVM images
- Files generated with user ownership
- Allows installation of any software
  - git tree url
  - git commit SHA
  - spec file
  - extra patches

```
#!/bin/bash
# ---
# git url: git://repo.or.cz/smatch.git
# git_commit: f0092dafe69db06b174122d301d8e3d7cdf3825
# other_files:
#   - 0001-Explicitly-use-python2-to-solve-rpmbuild-error.patch
patch -p1 < /opt/00*.patch
cat <<EOF > smatch.spec
Name: smatch
Version: 1
Release: 1%%dist
Summary: A semantic parser of source files
Group: Development/Tools
License: MIT
URL: http://smatch.sourceforge.net/
%description
Smatch is a semantic parser of source files.
%build
make %{?_smp_mflags}
%install
make INSTALL_PREFIX="/opt/smatch" DESTDIR="\{buildroot\}" PREFIX="/opt/smatch" install
mkdir -p \{buildroot\}/opt/smatch/share/smatch/smatch_data/
cp -r /opt/src/smatch_data/db \{buildroot\}/opt/smatch/share/smatch/smatch_data/
%clean
make clean
%files
/opt/smatch/share/man/man1/*
/opt/smatch/bin/*
/opt/smatch/include/*
/opt/smatch/share/smatch/*
/opt/smatch/share/smatch/smatch_data/db/*
/opt/smatch/lib/*
/opt/smatch/lib/pkgconfig/*
EOF
rpmbuild --build-in-place -bb smatch.spec
```
Build Code

- Silent and smart project build discovery
  - `mkt build <project_to_build>`
- Preconfigured CCACHE to speed up recompilation
- Proper compilation flags
- Correct understanding of number of available CPUs for build
- Build from recipe file for custom builds
- Able to build user space applications against new kernel headers, useful for user space development
- **Minimal** kernel .config
  - virtio-* drivers
  - Pre-configured to boot from 9pfs filesystem
  - Only Mellanox drivers are enabled

```bash
➜ kernel git:(rdma-next) pwd
 IMAGES/leonro/src/kernel
➜ kernel git:(rdma-next) time mkt build
Start kernel compilation in silent mode
mkt build 0.16s user 0.06s system 0% cpu 46.612 total
```
CI Testing

- Focused on code static analyzers
  - smatch from the git
  - sparse from the git
  - Latest gcc with extra warnings
  - checkpatch
  - clang-9
  - Various compilation tests

- Reuse support container and build runner
  - Common CCACHE
  - Deep patch inspection to compile only minimal part
  - Non-blocking asynchronous compilation
  - Executed with `mkt ci`

```bash
$ kernel git:(rdma-next) time mkt ci
2467425b0b34 (HEAD -> build) IB/mlx5: Add CREATE_PSV/DESTROY_PSV for devx interface
In file included from ./include/rdma/ib_verbs.h:64, from drivers/infiniband/hw/mlx5/mlx5_ib.h:38, from drivers/infiniband/hw/mlx5/gsi.c:33:
          ^~~~~~~~~~~~
./include/linux/dim.h:326:1: warning: `tx_profile’ defined but not used [-Wunused-const-variable=] 326 | tx_profile[DIM_CQ_PERIOD_NUM_MODES][NET_DIM_PARAMS_NUM_PROFILE] = {  |
          ^~~~~~~~~~~
./include/linux/dim.h:320:1: warning: `rx_profile’ defined but not used [-Wunused-const-variable=] 320 | rx_profile[DIM_CQ_PERIOD_NUM_MODES][NET_DIM_PARAMS_NUM_PROFILE] = {  |
          ^~~~~~~~~
mkt ci  0.17s user 0.05s system 0% cpu 1:13.80 total
```
Run Flow

- Rich CLI and configuration file
  - `mkt run`
- Fast boot into VM
- No need to generate VM image for QEMU
- No need to copy/install kernel and modules
- Includes working network and SSH connection
- Ctrl-A X closes QEMU and kills container
Rich Configuration Syntax

```bash
[defaults]
src = /images/leonro/src/
kernel = /images/leonro/src/kernel/
rdma-core = /images/leonro/src/rdma-core/
iproute2 = /images/leonro/src/iproute2/
simx = /images/leonro/src/simx/
logs = /images/leonro/logs/
ccache = /images/leonro/ccache/
image = simx
dir = /images/leonro/src/rdma-core /images/leonro/src/iproute2

[cx5-ib]
pci = 0000:05:00.0 0000:05:00.1
boot_script = /labhome/leonro/scripts/opensm

[cx4-ib]
pci = 0000:0b:00.0 0000:0b:00.1
boot_script = /labhome/leonro/scripts/opensm

[cx5-roce]
pci = 0000:88:00.0 0000:88:00.1

[cx4-roce]
pci = 0000:84:00.0 0000:84:00.1

[cx3]
pci = 0000:81:00.0
num_of_vfs = 3
boot_script = /labhome/leonro/scripts/opensm

[cxib]
pci = 0000:08:00.0
boot_script = /labhome/leonro/scripts/opensm

[simx]
pci = cx4-ib

[simx-sriov1]
pci = cx4-eth cx6-eth
num_of_vfs = 6
custom_qemu = true
```
QEMU Image

- Don’t create VM images – use container layout
- Built on the fly as docker entrypoint
- Everything is virtio-9p-pci
- Mount with systemd
QEMU Image

- Don’t create VM images – use container layout
- Built on the fly as docker entrypoint
- Everything is virtio-9p-pci
- Mount with systemd

Mount root fs inside docker
QEMU Image

- Don’t create VM images – use container layout
- Built on the fly as docker entrypoint
- Everything is virtio-9p-pci
- Mount with systemd

Mount root fs inside docker

Create systemd units to mount hypervisor dirs
QEMU Image

- Don’t create VM images – use container layout
- Built on the fly as docker entrypoint
- Everything is virtio-9p-pci
- Mount with systemd

1. Mount root fs inside docker
2. Create systemd units to mount hypervisor dirs
3. Configure qemu to passthrough fs
QEMU Image

- Don’t create VM images – use container layout
- Built on the fly as docker entrypoint
- Everything is virtio-9p-pci
- Mount with systemd

Mount root fs inside docker

Create systemd units to mount hypervisor dirs

Execute QEMU by replacing entry point

Configure qemu to passthrough fs
QEMU Image

- Don’t create VM images – use container layout
- Built on the fly as docker entrypoint
- Everything is virtio-9p-pci
- Mount with systemd

Mount root fs inside docker
Create systemd units to mount hypervisor dirs
Execute QEMU by replacing entry point
Configure qemu to passthrough fs
Boot QEMU OS

Boot QEMU OS
### QEMU Image

- Don’t create VM images – use container layout
- Built on the fly as docker entrypoint
- Everything is virtio-9p-pci
- Mount with systemd

- mount –bind / /mnt/self
- qemu ... -fsdev local,id=host_fs,security_model=passthrough,path=/mnt/self -device virtio-9p-pci,fsdev=host_fs,mount_tag=/dev/root ...
- os.execvp(....)
QEMU Network

- Managed interface
  - Containers run in privileged mode
    - `--net=host --privileged`
  - Senses br0 interface
    - + full external in/out network
    - - NAT access for localhost port 4444 connected to SSH
QEMU Network

- Managed interface
  - Containers run in privileged mode
    - --net=host --privileged
  - Senses br0 interface
    - + full external in/out network
    - - NAT access for localhost port 4444 connected to SSH

- Tested interface (External routing)
  - Disable reverse proxy and ARP filtering
  - Configure routing table
  - Increase priority of output port
  - Clean routing cache
External Routing (2 NICs)

```
echo 0 > /proc/sys/net/ipv4/conf/all/rp_filter
echo 1 > /proc/sys/net/ipv4/conf/all/accept_local
echo 1 > /proc/sys/net/ipv4/conf/all/arp_filter
echo 1 > /proc/sys/net/ipv4/conf/all/arp_ignore
echo 2 > /proc/sys/net/ipv4/conf/all/arp_announce
```

Disable filtering
External Routing (2 NICs)

echo 0 > /proc/sys/net/ipv4/conf/all/rp_filter
echo 1 > /proc/sys/net/ipv4/conf/all/accept_local
echo 1 > /proc/sys/net/ipv4/conf/all/arp_filter
echo 1 > /proc/sys/net/ipv4/conf/all/arp_ignore
echo 2 > /proc/sys/net/ipv4/conf/all/arp_announce

ip rule del pref 0
ip rule add from all lookup local pref 100

ip rule add iif eth1 lookup local pref 0
ip rule add from 192.168.122.76 table 10 pref 10
ip route add 192.168.122.0/24 dev eth1 src 192.168.122.76 table 10
ip route add local 192.168.122.76 dev eth1 src 192.168.122.76 table 10

ip rule add iif eth2 lookup local pref 0
ip rule add from 192.168.122.77 table 11 pref 10
ip route add 192.168.122.0/24 dev eth2 src 192.168.122.77 table 11
ip route add local 192.168.122.77 dev eth1 src 192.168.122.77 table 11
External Routing (2 NICs)

echo 0 > /proc/sys/net/ipv4/conf/all/rp_filter
echo 1 > /proc/sys/net/ipv4/conf/all/accept_local
echo 1 > /proc/sys/net/ipv4/conf/all/arp_filter
echo 1 > /proc/sys/net/ipv4/conf/all/arp_ignore
echo 2 > /proc/sys/net/ipv4/conf/all/arp_announce

ip rule del pref 0
ip rule add from all lookup local pref 100

ip rule add iif eth1 lookup local pref 0
ip rule add from 192.168.122.76 table 10 pref 10
ip route add 192.168.122.0/24 dev eth1 src 192.168.122.76 table 10
ip route add local 192.168.122.76 dev eth1 src 192.168.122.76 table 10

ip rule add iif eth2 lookup local pref 0
ip rule add from 192.168.122.77 table 11 pref 10
ip route add 192.168.122.0/24 dev eth2 src 192.168.122.77 table 11
ip route add local 192.168.122.77 dev eth1 src 192.168.122.77 table 11

ip rule add to 192.168.122.77 table 10 pref 10
ip rule add to 192.168.122.76 table 11 pref 10
External Routing (2 NICs)

echo 0 > /proc/sys/net/ipv4/conf/all/rp_filter
echo 1 > /proc/sys/net/ipv4/conf/all/accept_local
echo 1 > /proc/sys/net/ipv4/conf/all/arp_filter
echo 1 > /proc/sys/net/ipv4/conf/all/arp_ignore
echo 2 > /proc/sys/net/ipv4/conf/all/arp_announce

ip rule del pref 0
ip rule add from all lookup local pref 100

ip rule add iif eth1 lookup local pref 0
ip rule add from 192.168.122.76 table 10 pref 10
ip route add 192.168.122.0/24 dev eth1 src 192.168.122.76 table 10
ip route add local 192.168.122.76 dev eth1 src 192.168.122.76 table 10

ip rule add iif eth2 lookup local pref 0
ip rule add from 192.168.122.77 table 11 pref 10
ip route add 192.168.122.0/24 dev eth2 src 192.168.122.77 table 11
ip route add local 192.168.122.77 dev eth1 src 192.168.122.77 table 11

ip rule add to 192.168.122.77 table 10 pref 10
ip rule add to 192.168.122.76 table 11 pref 10

ip route flush cache
QEMU Hardware Support

- Based on VFIO PCI

qemu ... -device vfio-pci,host=PCI_BOF ...
QEMU Hardware Support

- Based on VFIO PCI
- `qemu ... -device vfio-pci,host=PCI_BOF ...`

Unbind from real driver → Bind to vfio-pci → Pass PCI passthrough
github.com/mellanox/mkt
Join us and make MKT generic
Thank You