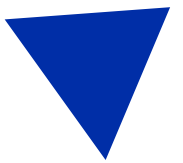
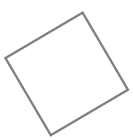







Seamless Kernel Update

Author: 何文良 hewenliang4@huawei.com
何静娴 hejingxian@huawei.com

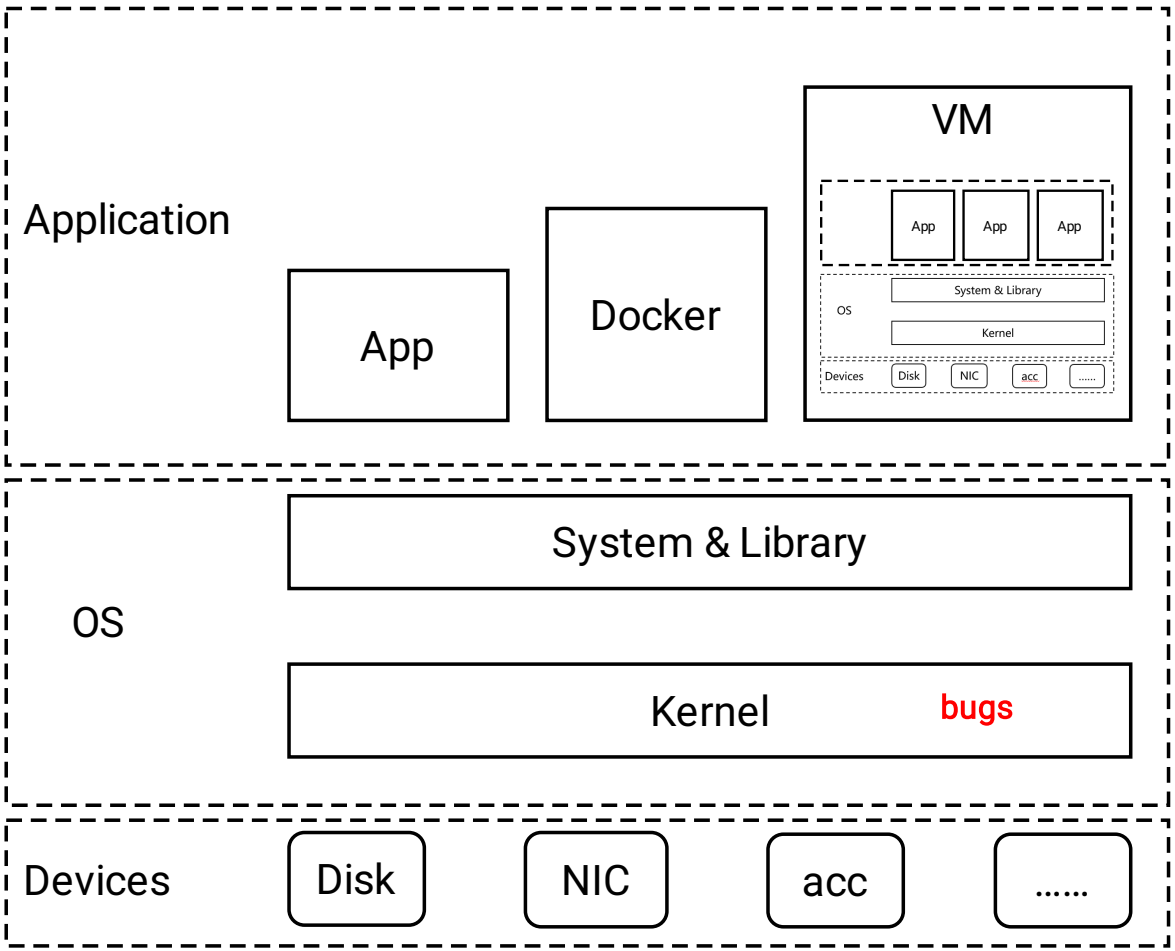
openEuler

目录

openEuler

difficult

01 Background



Kernel Bugs

- Kernel live patch to fix the bug
  - No live patch for some bug
  - Manager difficult
- Live migration for APP/VM & Reboot
  - No method for the pass-thought device
  - Difficult to transmit large memory

For Example

Machine: Bare-Metal Server Kunpeng 920
Memory： 380G
Application： Mysql (DB)

* Difficult to transmit large memory data

1.1 Instant OS Updates via Userspace Checkpoint-and-Restart

USENIX

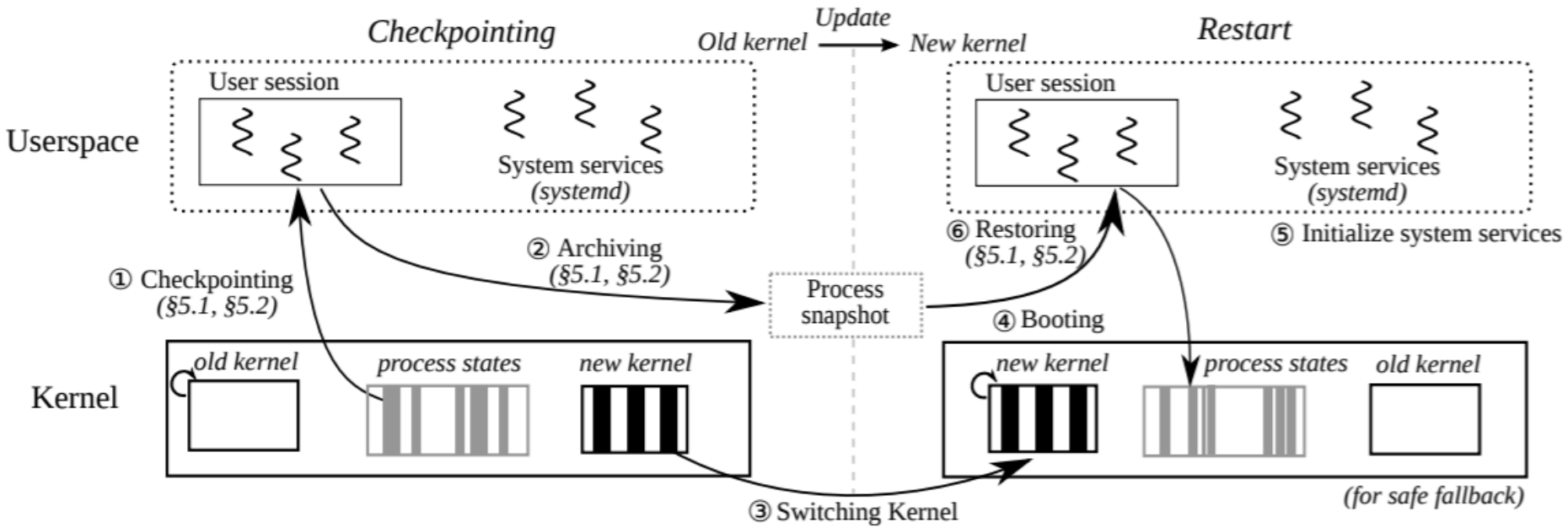
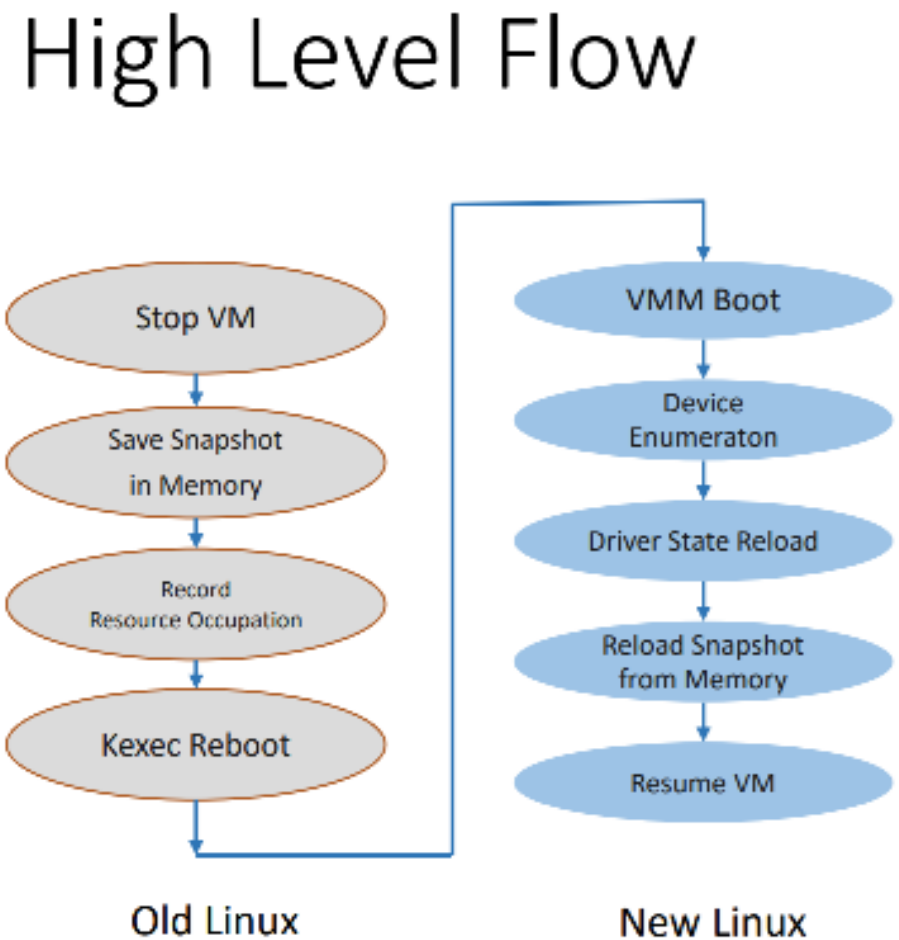


Figure 2: Overview of KUP's updating procedures. KUP first checkpoints user's processes ①, and archives their snapshots ②. After checkpointing selected processes in a user's current session, KUP replaces the old kernel to the new kernel image ③, and finally switches to the new kernel ④. After the new kernel boots, KUP first initializes its system daemons ⑤, and finally restores snapshots of user applications ⑥.

1.2 Seamless Cloud System Upgrade with VMM fast Restart

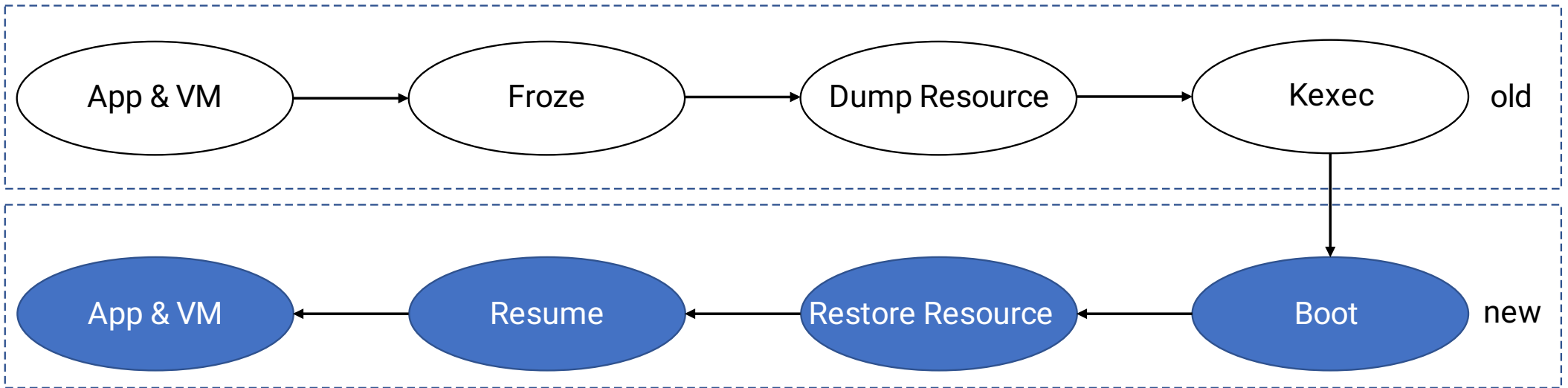


KVM Forum

- Save Snapshot in Memory (in old Linux) & VM Quit
  - DAX filesystem in DRAM-as-PMEM
  - Don't free HW resources (IRTE, etc.)
- Record Resource Occupation
  - Device list, memory, etc.
- KEXEC Reboot
  - No hardware clobber in driver shutdown
- VMM Boot (new Linux)
  - Reserve resources
- Device Enumeration
  - No hardware clobber in PCI enumeration
  - No native driver attaching
- Driver State Reload
  - IOMMU driver reload state
- Reload Snapshot from Memory (in new Linux)
  - Re-enable DAX filesystem in DRAM-as-PMEM
- Resume VM
  - Reload DMA mapping
  - Re-enable MSI/MSIX

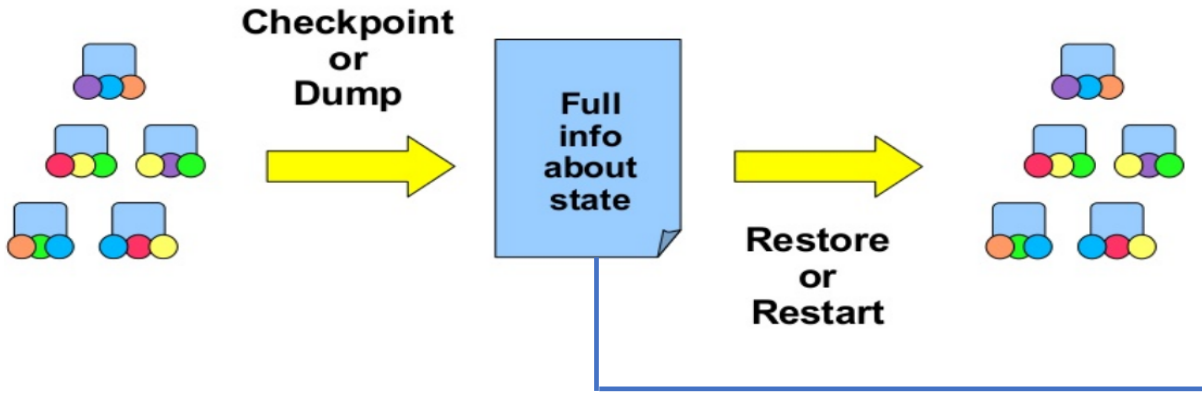
openEuler

02 Froze/Resume the Application



QEMU： qemu save/restore
CRIU： criu dump/restore
DMTCP： dmtcp save/restore

03 Keep Memory



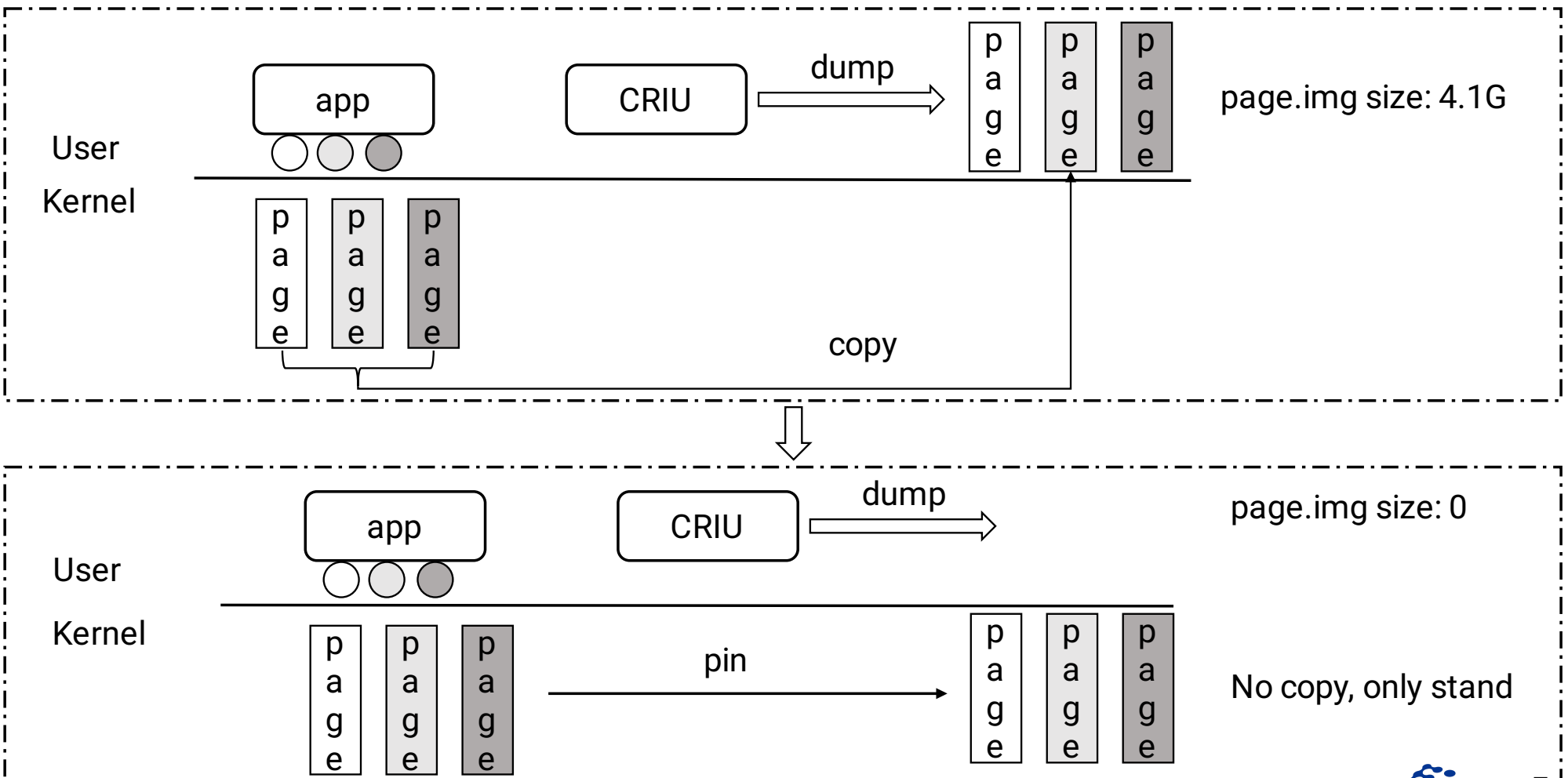

```
root@PEK1000107703:~# ls -alh output/
total 4.1G
drwxrwxr-x 2 luan luan 4.0K Nov 15 10:36 .
drwxrwxr-x 3 luan luan 4.0K Nov 15 10:39 ..
-rw-r--r-- 1 root root 1.8K Nov 15 10:19 core-3567.img
-rw-r--r-- 1 root root 1.8K Nov 15 10:36 core-5681.img
-rw-r--r-- 1 root root   44 Nov 15 10:33 fdinfo-2.img
-rw-r--r-- 1 root root  657 Nov 15 10:36 files.img
-rw-r--r-- 1 root root   18 Nov 15 10:19 fs-3567.img
-rw-r--r-- 1 root root   18 Nov 15 10:36 fs-5681.img
-rw-r--r-- 1 root root   34 Nov 15 10:19 ids-3567.img
-rw-r--r-- 1 root root   34 Nov 15 10:36 ids-5681.img
-rw-r--r-- 1 root root   42 Nov 15 10:36 inventory.img
-rw-r--r-- 1 root root 1.4K Nov 15 10:19 mm-3567.img
-rw-r--r-- 1 root root 1.4K Nov 15 10:36 mm-5681.img
-rw-r--r-- 1 root root  307 Nov 15 10:19 pagemap-3567.img
-rw-r--r-- 1 root root  16K Nov 15 10:36 pagemap-5681.img
-rw-r--r-- 1 root root 4.1G Nov 15 10:36 pages-1.img
-rw-r--r-- 1 root root   26 Nov 15 10:36 pstree.img
-rw-r--r-- 1 root root   12 Nov 15 10:36 seccomp.img
-rw-r--r-- 1 root root   57 Nov 15 10:36 stats-dump
-rw-r--r-- 1 root root   26 Nov 15 10:19 stats-restore
-rw-r--r-- 1 root root  179 Nov 15 10:36 tty-info.img
```

CRIU writes the copy of application memory into disk file for restoring the application. When the data is large, the copy operation will cost too much time.

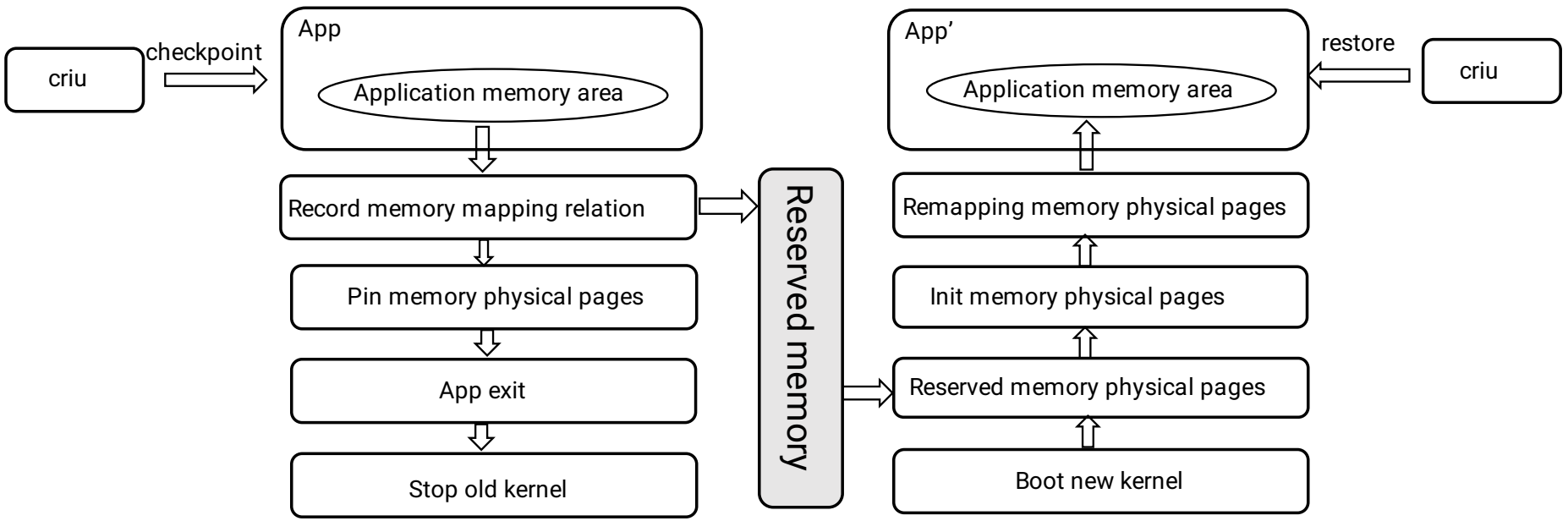
In order to avoid copy and read operation, we can keep the memory(pin memory) and remap the memory to the restoring task.

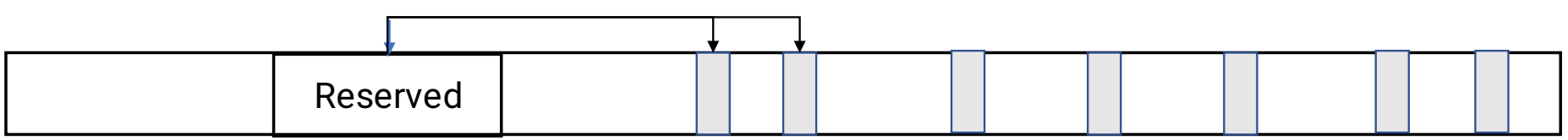
openEuler

3.1 Pin application memory

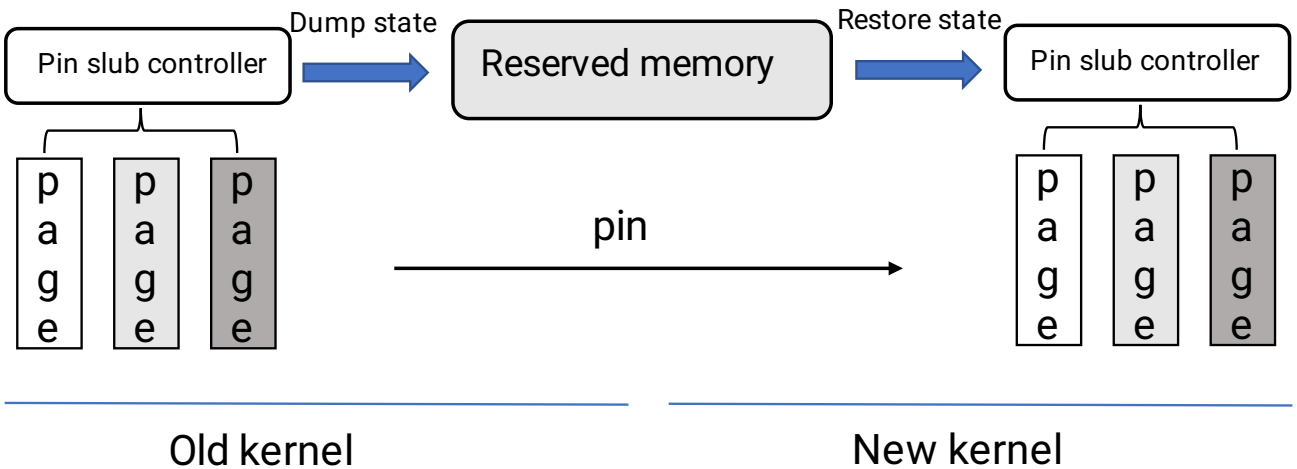


3.2 Keep user memory unchanged in new kernel





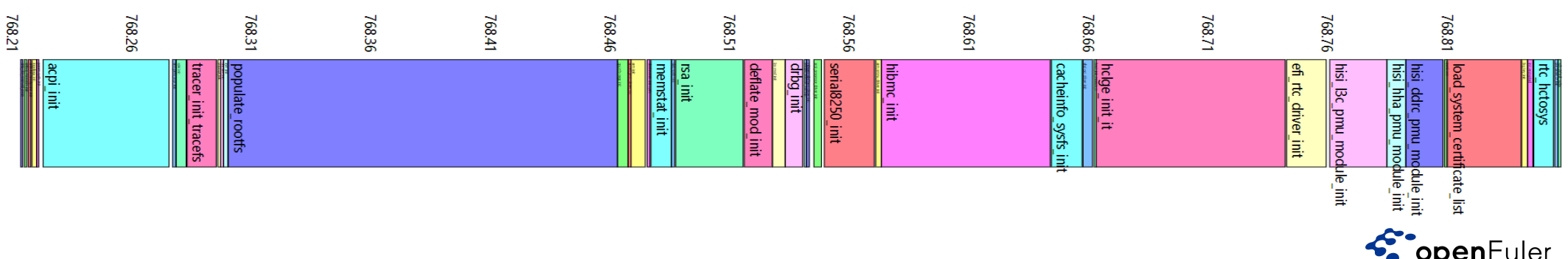
3.3 Keep kernel memory unchanged in new kernel



*Create pin slub controller to manage the old kernel pages which need to keep constant While booting the new kernel.

04 Kernel Fast Boot

- CPU Park & Unpark
- Pre decompress kernel & initrd
- Memory Defer initialization
- Deferring device driver probe
- Deferring device initialization
- Avoid unless device scanning
- NIC PHY issue



openEuler

4.1 CPU Park & Unpark

- In orignal CPU booting method, each ARM cpu core booting costs 0.036s.

```
[    2.796042] Detected VIPT I-cache
[    2.796078] GICv3: CPU81: found re
[    2.796097] GICv3: CPU81: using al
[    2.796155] CPU81: Booted secondar
[    2.836491] Detected VIPT I-cache
[    2.836527] GICv3: CPU82: found re
[    2.836546] GICv3: CPU82: using al
[    2.836603] CPU82: Booted secondar
[    2.876940] Detected VIPT I-cache
[    2.876977] GICv3: CPU83: found re
[    2.876996] GICv3: CPU83: using al
[    2.877053] CPU83: Booted secondar
```

- By use the cpu park and unpark method, each ARM cpu core booting costs less than 0.0004s.

```
[    0.049353] Detected VIPT I-cac
[    0.049358] GICv3: CPU6: found
[    0.049364] GICv3: CPU6: using
[    0.049396] CPU6: Booted second
[    0.049502] park_text 0xffff000
[    0.049510] Write cpu 7 entry 0
[    0.049681] Detected VIPT I-cac
[    0.049687] GICv3: CPU7: found
[    0.049692] GICv3: CPU7: using
[    0.049723] CPU7: Booted second
[    0.049827] park_text 0xffff000
[    0.049835] Write cpu 8 entry 0
[    0.050011] Detected VIPT I-cac
[    0.050018] GICv3: CPU8: found
```

4.2 Pre decompress kernel & initrd

- Decompress bzImage to Image(vmlinux) before kexec.
- Enhance kexec-tools to support for original kernel image.
- Decompress initramfs.gz before kexec.
- Unpack initramfs to ramfs in RAM before kexec.

4.3 Memory Defer initialization

mm: parallelize deferred_init_memmap()

```
mm: parallelize deferred_init_memmap()

Deferred struct page init is a significant bottleneck in kernel boot.
Optimizing it maximizes availability for large-memory systems and allows
spinning up short-lived VMs as needed without having to leave them
running.  It also benefits bare metal machines hosting VMs that are
sensitive to downtime.  In projects such as VMM Fast Restart[1], where
guest state is preserved across kexec reboot, it helps prevent application
and network timeouts in the guests.

Multithread to take full advantage of system memory bandwidth.
```

Intel(R) Xeon(R) Platinum 8167M CPU @ 2.00GHz (Skylake, bare metal)
2 nodes * 26 cores * 2 threads = 104 CPUs
384G/node = 768G memory

	kernel boot			deferred init		
node% (thr)	speedup	time_ms	(stdev)	speedup	time_ms	(stdev)
(0)	--	4089.7	(8.1)	--	1785.7	(7.6)
2% (1)	1.7%	4019.3	(1.5)	3.8%	1717.7	(11.8)
12% (6)	34.9%	2662.7	(2.9)	79.9%	359.3	(0.6)
25% (13)	39.9%	2459.0	(3.6)	91.2%	157.0	(0.0)
37% (19)	39.2%	2485.0	(29.7)	90.4%	172.0	(28.6)
50% (26)	39.3%	2482.7	(25.7)	90.3%	173.7	(30.0)
75% (39)	39.0%	2495.7	(5.5)	89.4%	190.0	(1.0)
100% (52)	40.2%	2443.7	(3.8)	92.3%	138.0	(1.0)

Intel(R) Xeon(R) CPU E5-2699 v3 @ 2.30GHz (Haswell, bare metal)
2 nodes * 18 cores * 2 threads = 72 CPUs
128G/node = 256G memory

	kernel boot			deferred init		
node% (thr)	speedup	time_ms	(stdev)	speedup	time_ms	(stdev)
(0)	--	1680.0	(4.6)	--	627.0	(4.0)
3% (1)	0.3%	1675.7	(4.5)	-0.2%	628.0	(3.6)
11% (4)	25.6%	1250.7	(2.1)	67.9%	201.0	(0.0)
25% (9)	30.7%	1164.0	(17.3)	81.8%	114.3	(17.7)
36% (13)	31.4%	1152.7	(10.8)	84.0%	100.3	(17.9)
50% (18)	31.5%	1150.7	(9.3)	83.9%	101.0	(14.1)
75% (27)	31.7%	1148.0	(5.6)	84.5%	97.3	(6.4)
100% (36)	32.0%	1142.3	(4.0)	85.6%	90.0	(1.0)

4.4 Deferring device driver probe

drivercore: add driver probe deferral mechanism

```
drivercore: Add driver probe deferral mechanism

Allow drivers to report at probe time that they cannot get all the resources
required by the device, and should be retried at a later time.
```

```
module.async_probe [KNL]
                Enable asynchronous probe on this module.
```

```
driver_async_probe=  [KNL]
                List of driver names to be probed asynchronously.
                Format: <driver_name1>,<driver_name2>...
```

```
deferred_probe_timeout=
                [KNL] Debugging option to set a timeout in seconds for
                deferred probe to give up waiting on dependencies to
                probe. Only specific dependencies (subsystems or
                drivers) that have opted in will be ignored. A timeout of 0
                will timeout at the end of initcalls. This option will also
                dump out devices still on the deferred probe list after
                retrying.
```

4.5 Part deferring device initialization

- Deferring device driver probe can only defer the entire device initialization , however, the kernel booting only uses part device function at sometime , we can defer the other part of the device initialization. For example:

```
0.860672] hns3 0000:7d:00.0: The firmware version is 1.8.12.3
0.940779] hns3 0000:7d:00.0: hclge driver initialization finished.
0.941897] hns3 0000:7d:00.1: The firmware version is 1.8.12.3
1.020563] hns3 0000:7d:00.1: hclge driver initialization finished.
1.021641] hns3 0000:7d:00.2: The firmware version is 1.8.12.3
1.100787] hns3 0000:7d:00.2: hclge driver initialization finished.
1.101882] hns3 0000:7d:00.3: The firmware version is 1.8.12.3
1.180567] hns3 0000:7d:00.3: hclge driver initialization finished.
```

- When we only use one hns3 NIC at kernel booting stage, the others can be deferred.

4.6 Avoid unless device scanning

Some device state has no change during kernel booting. Then we can store the device state information in reserved memory, and use the information to restore the device state instead of rescanning the devices:

- PCI state information
- SATA disk state information
- ……

4.7 NIC PHY issue

- When NIC reset PHY and reload firmware, the NIC transports no package to system until 3-5s later, which make the downtime of the application too long.

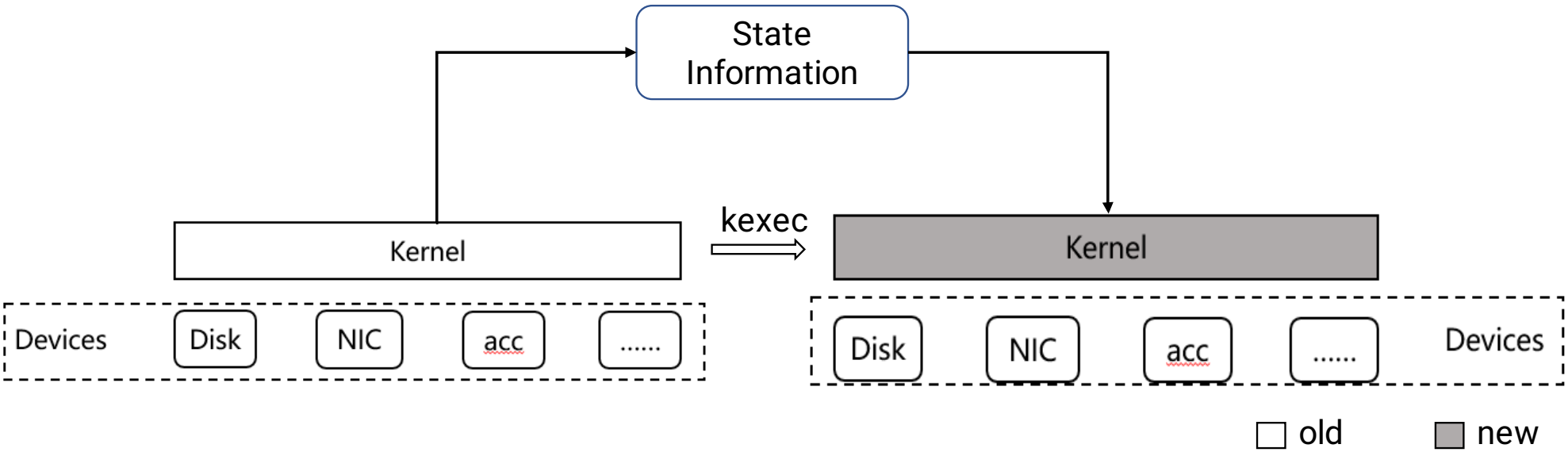
```
Link detected: yes 16:18:42
Link detected: no 16:18:43
Link detected: no 16:18:44
Link detected: no 16:18:45
Link detected: no 16:18:46
Link detected: no 16:18:47
Link detected: no 16:18:48
Link detected: no 16:18:49
Link detected: no 16:18:50
Link detected: no 16:18:51
Link detected: no 16:18:52
Link detected: no 16:18:53
Link detected: yes 16:18:54
```

- Modify the driver of the NIC， and avoid resetting PHY and reloading firmware during quick rebooting.

```
[ 4263.380716] icmp: icmp_echo: id=52763, seq=19903
[ 4263.484684] icmp: icmp_echo: id=52763, seq=19904
[ 4263.588715] icmp: icmp_echo: id=52763, seq=19905
[ 4263.692706] icmp: icmp_echo: id=52763, seq=19906
[ 4263.800706] icmp: icmp_echo: id=52763, seq=19907
[ 4263.904708] icmp: icmp_echo: id=52763, seq=19908
[ 4264.008702] icmp: icmp_echo: id=52763, seq=19909
[ 4264.116695] icmp: icmp_echo: id=52763, seq=19910
[ 4264.224690] icmp: icmp_echo: id=52763, seq=19911
[ 4264.328679] icmp: icmp_echo: id=52763, seq=19912
```

* Should make sure the NIC state correct without resetting PHY layer.

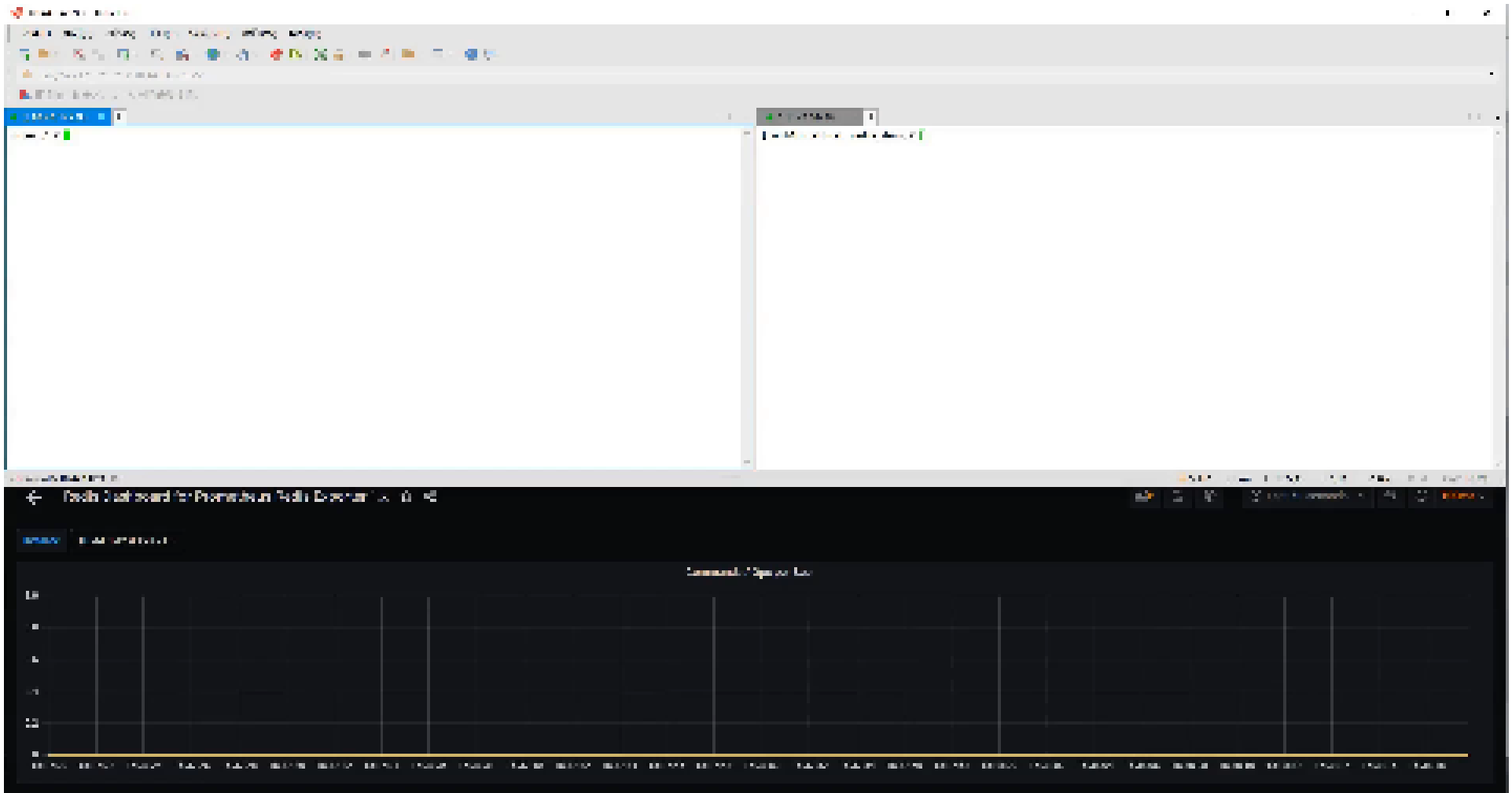
05 Keep Device State



State information：reserve memory to save device state information

* Caution: make the device state and kernel driver information concurrent.

06 Demo -- Benchmark



openEuler

07 Todo

- Open Source to openEuler
- Replace CRIU with kernel module
- Standard Device State Process
- Faster Kernel boot
- Multi Kernel parallel initialization

Reference

1 https://criu.org/Seamless_kernel_upgrade

2 https://www.usenix.org/system/files/conference/atc16/atc16_paper-kashyap.pdf

3 https://kvmforum2019.sched.com/event/TmvJ/seamless-cloud-system-upgrade-with-vmm-fast-restart-jason-zeng-intel

4 https://www.linuxplumbersconf.org/event/4/contributions/281/attachments/216/617/LPC_2019_kernel_fastboot_on_the_way.pdf

5 http://dmtcp.sourceforge.net/papers/dmtcp.pdf

Q&A

欢迎关注

新浪微博：openEuler社区　Twitter：openEuler　B站：openEuler　微信公众号：openEuler

社区网站

https://openeuler.org

代码托管平台

https://gitee.com/openeuler

微信交流群

添加小助手微信号“openeuler123”拉你进群

openEuler

openEuler