Technische Universität Braunschweig

Institute of Operating Systems and Computer Networks

NV-Hypervisor: Hypervisor-based Persistence for Virtual Machines

Vasily A. Sartakov, Rüdiger Kapitza

The 44th Annual IEEE/IFIP International Conference on Dependable Systems and Networks (DSN 2014)

Agenda

- Intro
- Models of persistence
- Hypervisor-based persistence
- Implementation
- Evaluation and early results
- Future work

Clouds

Infrastructure

- Virtualization
- Availability 24/7
  - Fault tolerance
  - Fast recovery

Power fault protection

- UPS
- Power generator
- Additional costs and does not provide 100% protection for data.

Faults happen

- Loss of Data
- Loss of performance
- Unavailability of services
- Time-consuming recovery

Non-volatile random-access memory

NVRAM

- Retain data during power outage
- DRAM-like performance
- Prospective technologies provide more than DRAM density
- Byte-addressable access

Main memory can be persistent and storage-sized

- Requires new design and architecture of system and applied software

System architecture in Persistent area

Revision of hardware and software architectures

- New models of execution (start/stop/install/remove/update)
- Which part of the system should be persistent (Whole? Library? Process?)
- What about legacy, proprietary and code reusing?
- What about hardware, like storage, devices, etc?

We propose

- Hypervisor-based persistence: model of persistence
- NV-Hypervisor: lightweight extension of QEMU for working in non-volatile environment

Agenda

- Intro
- Models of persistence
- Hypervisor-based persistence
- Implementation
- Evaluation and early results
- Future work

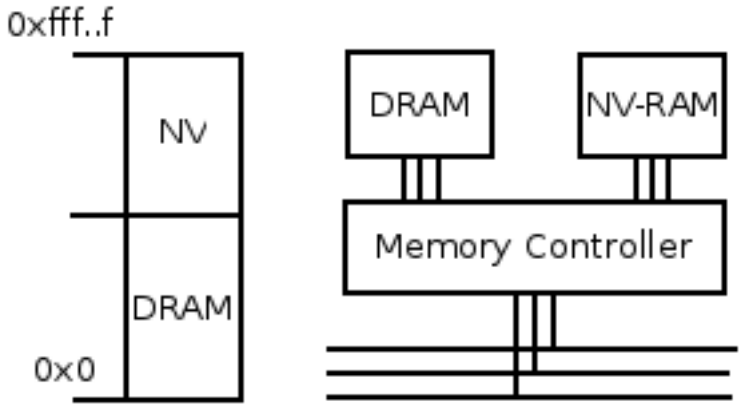
Models of persistent :: Hardware

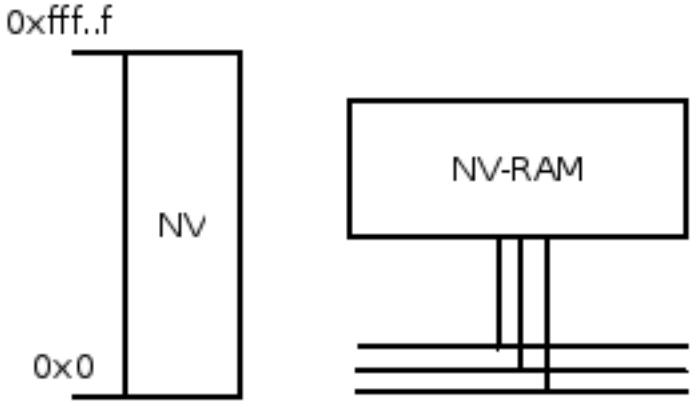
How NVRAM is attached?

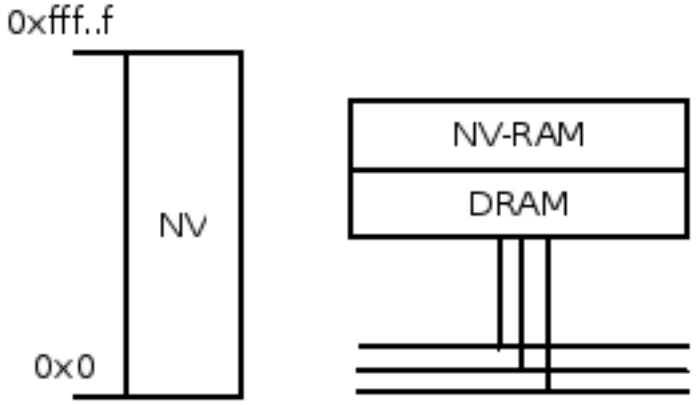
- Hybrid memory:

  NVRAM + DRAM, serial and parallel connection

- Only NVRAM







Models of persistent :: Software

Type of persistence	Main Idea	NV Abstraction	Modification
Language/library-based	Development tools provide primitives for utilise NVRAM in programs.	Variables and objects	Kernel, libc, programs
Process-based	Kernel uses NVRAM for allocating memory for new process.	Whole programs	Kernel
System-wide	All parts of a system execute directly in NVRAM	All programs and kernel	Kernel

Models of persistent :: Software

Type of persistence	Main Idea	NV Abstraction	Modification
Language/library-based	Development tools provide primitives for utilise NVRAM in programs.	Variables and objects	Kernel, libc, programs
Process-based	Kernel uses NVRAM for allocating memory for new process.	Whole programs	Kernel
System-wide	All parts of a system execute directly in NVRAM	All programs and kernel	Kernel

No solution allows direct usage of NV-RAM for legacy and proprietary software

Agenda

- Intro
- Models of persistence
- Hypervisor-based persistence
- Implementation
- Evaluation and early results
- Future work

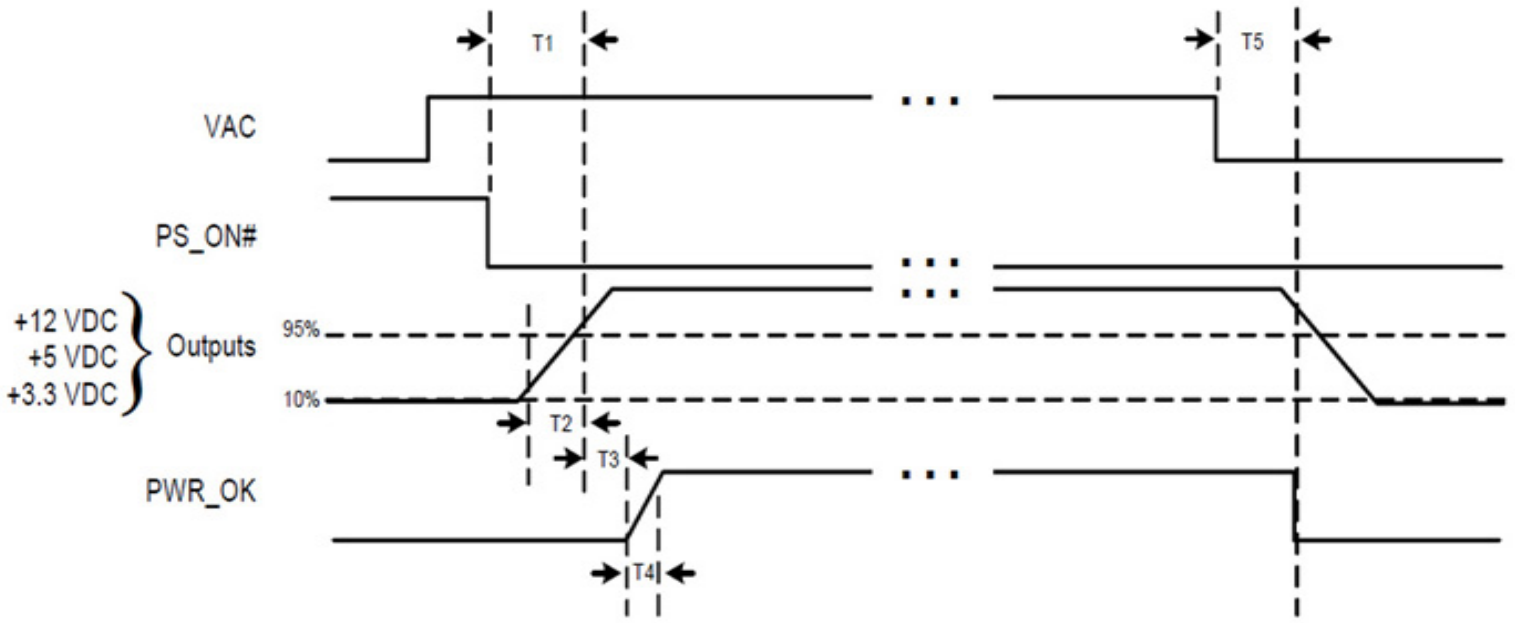
Hypervisor-based persistence

- Transparent provided by NV-Hypervisor
- Hybrid memory use
- Protect virtual machines from power outage
- Legacy and proprietary SW support

NVH::Assumptions and HW support

- NVRAM and DRAM parallel use
- Volatile CPU and devices
- Power outage detector (POD)
- Residual energy
  - Heiser, Gernot, et al. "RapiLog: Reducing system complexity through verification." Proceedings of the 8th ACM European Conference on Computer Systems. ACM, 2013.
  - Narayanan, Dushyanth, and Orion Hodson. "Whole-system persistence." ACM SIGARCH Computer Architecture News. Vol. 40. No. 1. ACM, 2012.

Residual Energy



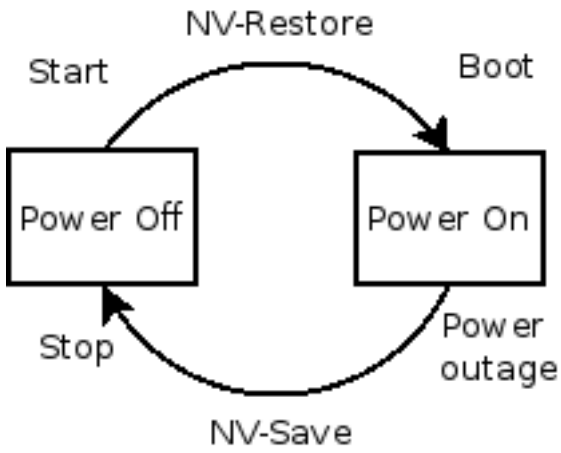
PWR_OK Sense level = 95% of nominal

T1: Power-on time
T2: Rise time
T3: PWR_OK delay
T4: PWR_OK rise time
T5: AC loss to PWR_OK hold-up time

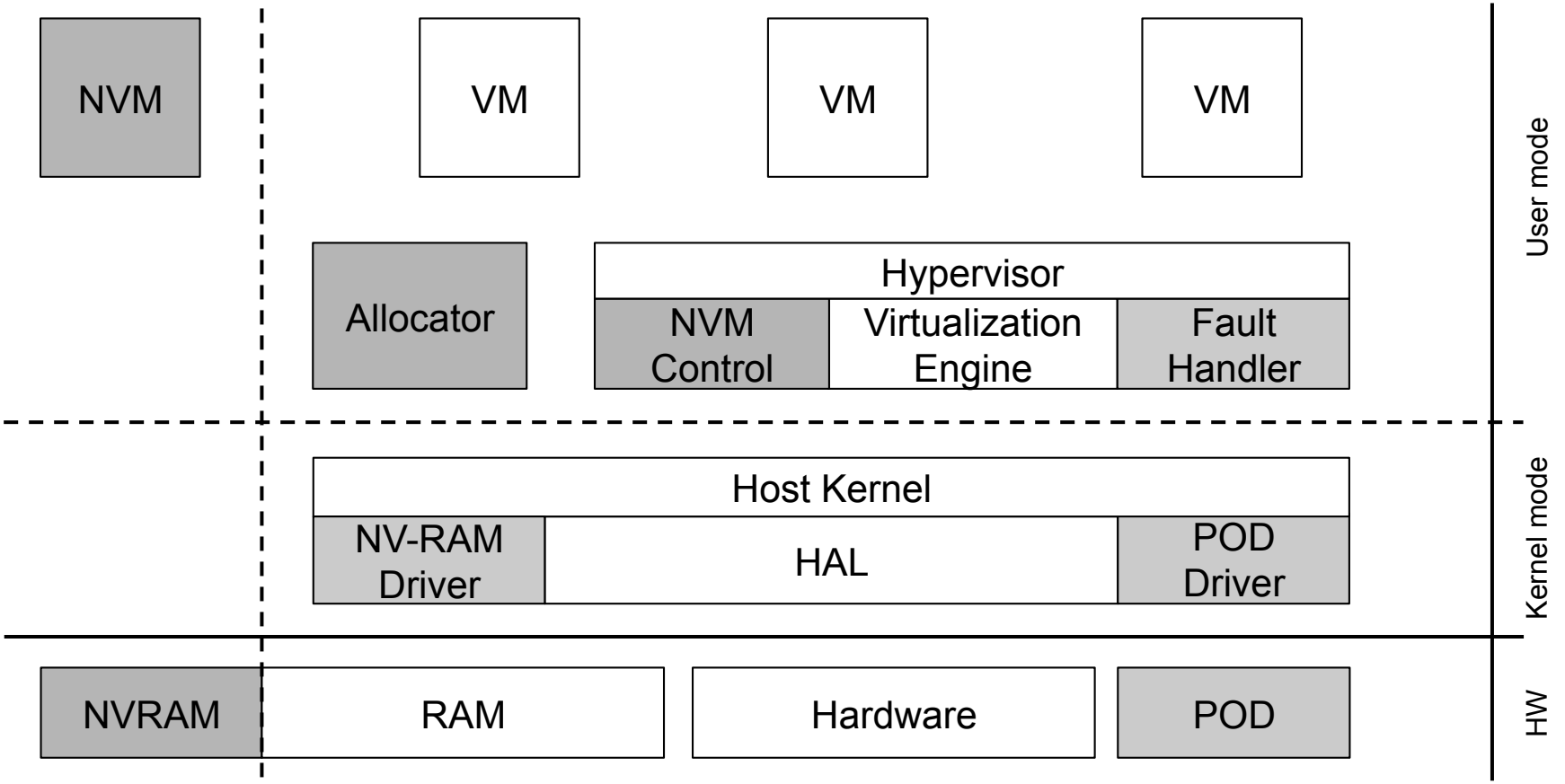
Source: http://www.techpowerup.com/articles/overclocking/psu/160/8

NV-Hypervisor::execution flow

- Initial Boot
  - Create and start execution of VMes
- Power outage
  - Saving sequence
- System Restore
- VM recovery



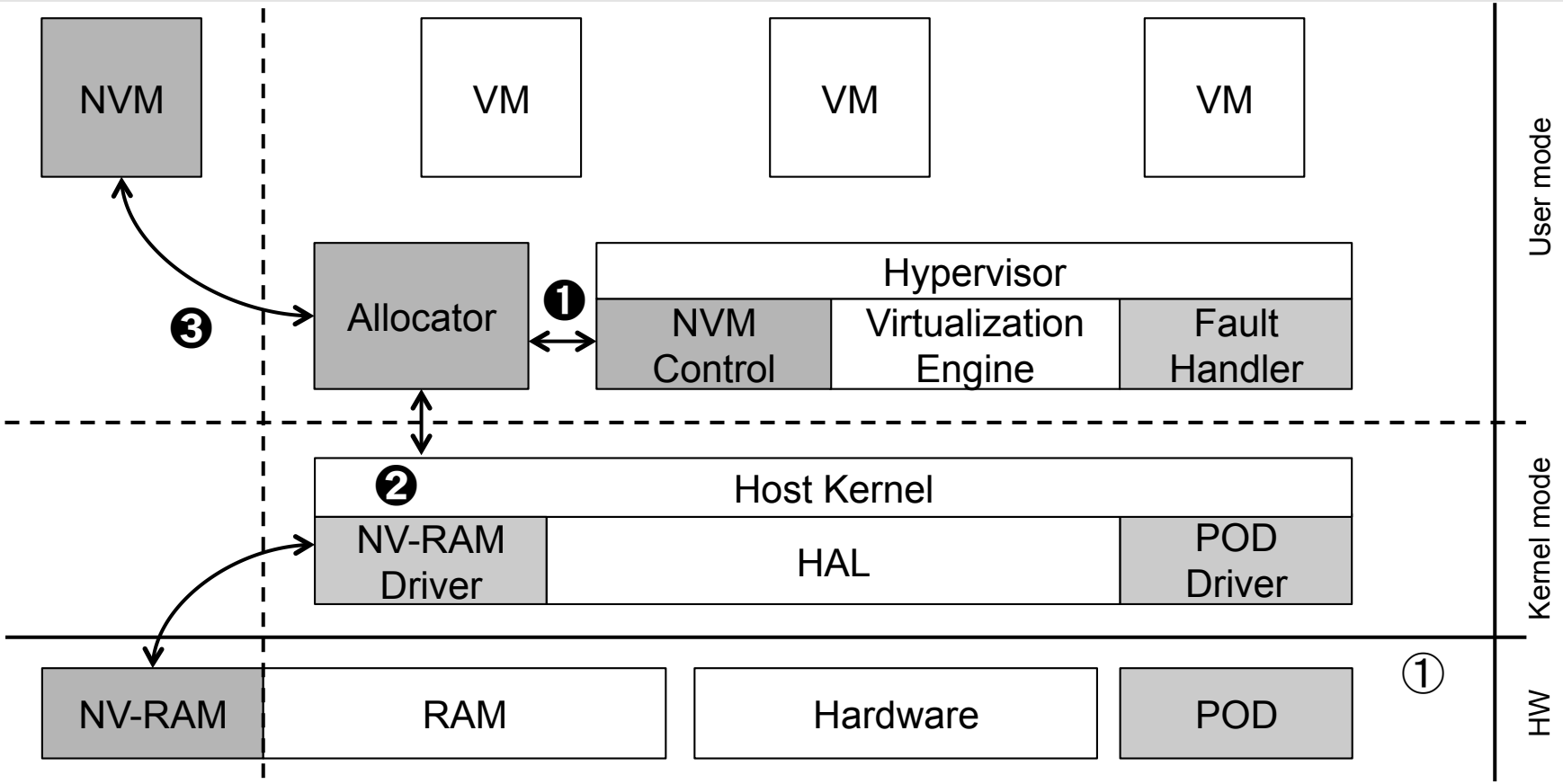
NV-Hypervisor::System architecture



- POD device, driver and fault handler
- NVRAM and DRAM
- Hypervisor

- Volatile and non-volatile VMs
- NV-RAM allocator

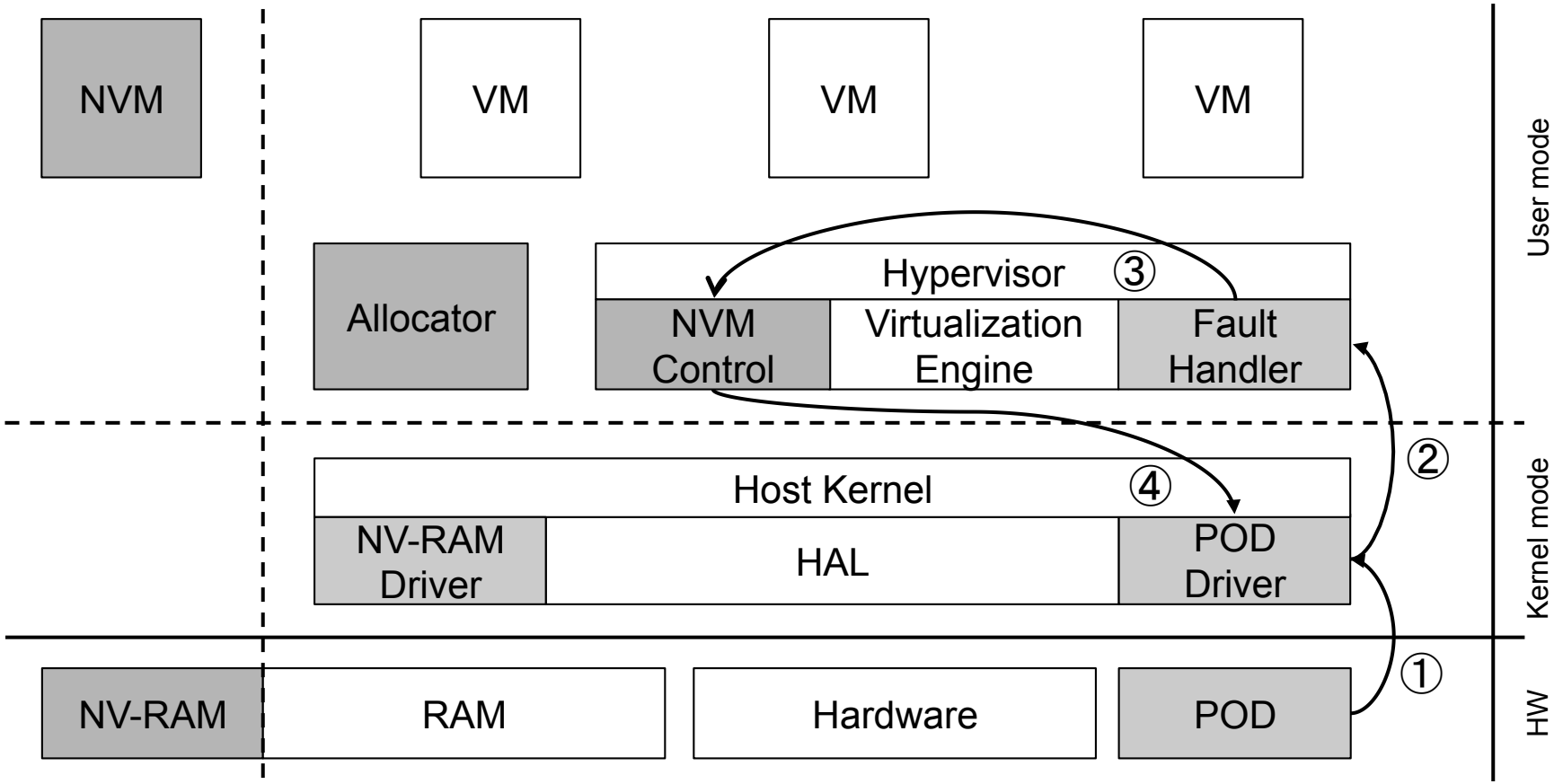
NV-Hypervisor::Creation of persistent VM



❶ Memory allocator
❷ NVRAM is mapped to memory map
❸ Allocate VM in NVRAM

- Similar to ordinary boot
- VM Configuration is provided by user (size, type, cpus, etc)

NV-Hypervisor::Power outage



① POD initiates interrupt
② Driver notifies NV-Hypervisor
③ Saving volatile state of VM
④ POD blocks memory operation

Recovery

- Ordinary boot of host system
- Reassigning non-volatile memory region during the initialization of VM
- State of virtual devices and CPU are restored
- Persistent VM continues execution

In sum

- Lightweight extension of hypervisor
- Use NVRAM and DRAM in parallel
- Allocate memory of NVM in the NVRAM
- Use power outage detector for tracking power fault and residual energy for finalizing state saving
- Save state of VM (virtual devices and vCPU) during the power outage
- Reuse persistent memory after power recovery

Agenda

- Intro
- Models of persistence
- Hypervisor-based persistence
- Implementation
- Evaluation and early results
- Future work

Implementation::base

- QEMU virtualization platform
- NVDIMMs – Battery backed DRAM provided by Viking Technology

Implementation::NVRAM support

Kernel module

- Adds NVRAM memory range into Memory map
- Provides read/write access to configuration registers

POD device

- Kernel module that catch non-mask interrupt and sends signal to hypervisor

Implementation::NVRAM support

libnvram

- *nv_alloc()*, *nv_free()*, *nv_init()* functions – allocating, freeing, initialization of persistent memory

QEMU extensions

- Signal catcher from POD, initiates saving sequence
- *dump-devices* and *nv-restore*: new QEMU monitor command, events for saving and restoring virtual devices
- "-nvm, nv-restore": new arguments for QEMU

Evaluation

- 2x Xeon server, 4GB main DRAM + 4GB NVRAM
- GNU/Linux (kernel version 3.4.12) as host system
- GNU/Linux as guest system
- QEMU 1.4.2
- “Power Good” signal as POD
- QEMU in a binary translation mode

Testing environment

Sysbench oltp test

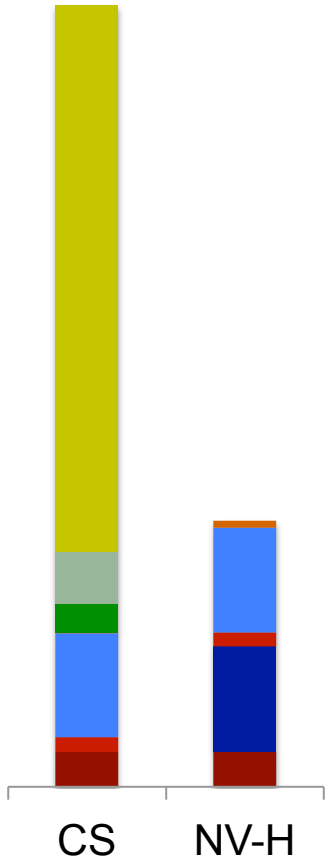
- Random access to 1000000 lines in table

Measure:

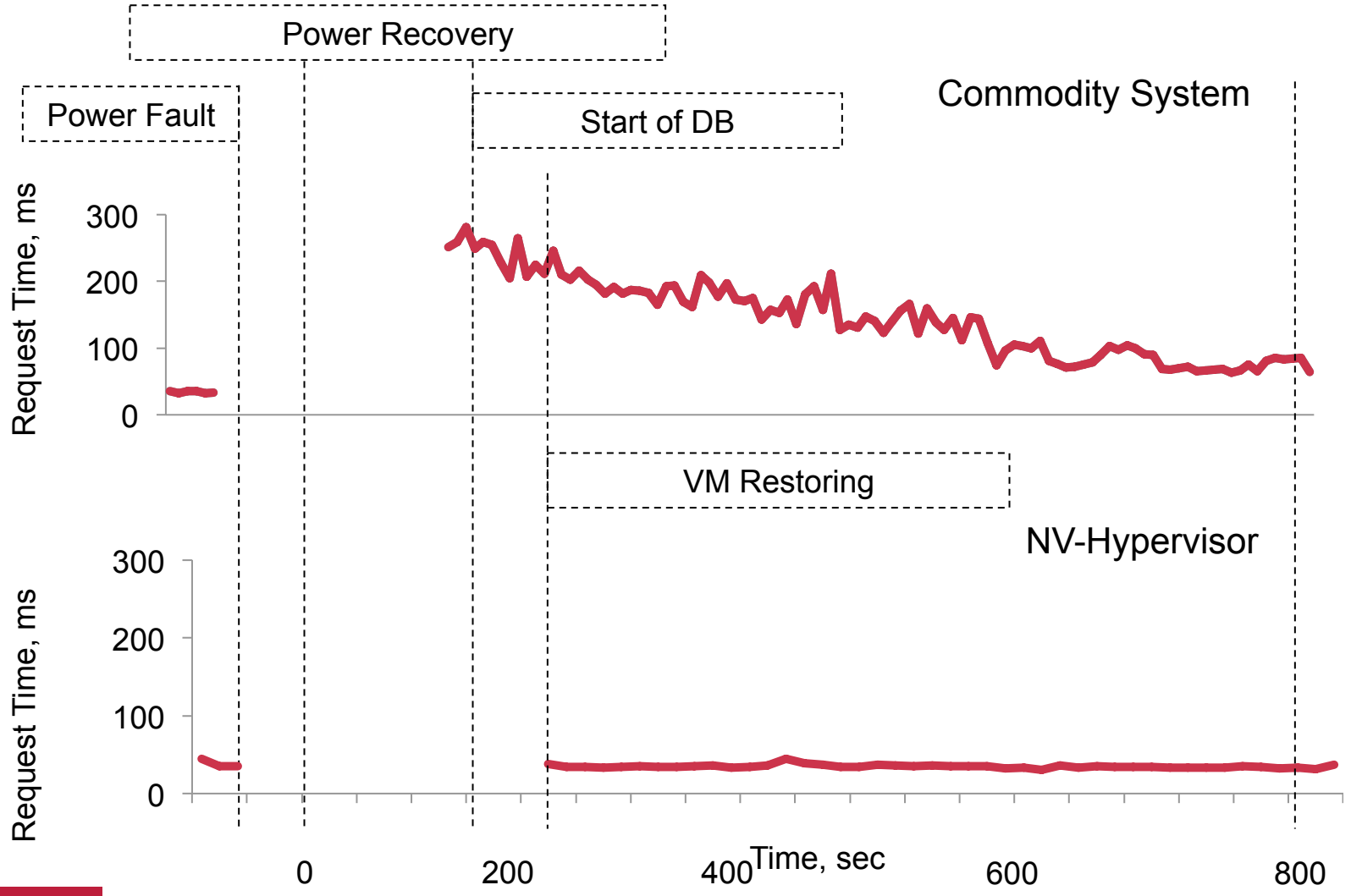
- Overall Recovery time
- Performance recovery time

Overall recovery time

Booting Step	Commodity System (sec)	NV-Hypervisor (sec)
DB warm up	566	n/a
DB recovery	54	n/a
Guest OS boot	31	n/a
QEMU starts	0.2	8
Host boot	108	108
BIOS	15	15
NV-DIMMs recovery	n/a	109
Server boot	36	36



Performance



Timings

Time for saving volatile state into the Storage:

8ms

Time for pushing Event from POD to QEMU:

11ms

Expected time:

30-50ms

Time from “Power good signal”

Around 1ms

Conclusion

- WIP: hypervisor-based persistence and NV-Hypervisor
- Can be used already with market available NVRAM technologies
- Decrease recovery time significant

Future Work

- Virtual memory support
- Hardware virtualization (VT-x) support
- New POD