bhyvearm64: CPU and Memory Virtualization on Armv8.0-A

Alexandru Elisei
alexandru.elisei@gmail.com

BSDCan 2019
Ottawa, Canada
May 2019

About Me

- Bachelor's Degree in Computer Science from University POLITEHNICA of Bucharest (Summer of 2018)
- bhyvearm64 was the subject of my Bachelor's thesis

FreeBSD

Outline

FreeBSD

Outline

FreeBSD

bhyvearm64

- What is bhyvearm64?
  - Type 2 hypervisor for FreeBSD
  - Virtualization on Armv8.0-A
  - Based on bhyve for x86 and Armv7
  - Not yet integrated with the FreeBSD kernel

[1] https://arm.com/fvp

bhyvearm64

- What is bhyvearm64?
  - Type 2 hypervisor for FreeBSD
  - Virtualization on Armv8.0-A
  - Based on bhyve for x86 and Armv7
  - Not yet integrated with the FreeBSD kernel
- What can it do?
  - Run a FreeBSD virtual machine on the Foundation Platform[1]
  - It can use virtio-mmio for network and block devices

[1] https://arm.com/fvp

bhyvearm64

- What is bhyvearm64?
  - Type 2 hypervisor for FreeBSD
  - Virtualization on Armv8.0-A
  - Based on bhyve for x86 and Armv7
  - Not yet integrated with the FreeBSD kernel
- What can it do?
  - Run a FreeBSD virtual machine on the Foundation Platform[1]
  - It can use virtio-mmio for network and block devices
- What it cannot do?
  - Create multiple virtual CPUs for a virtual machine
  - Has not been run on real hardware

FreeBSD

[1] https://arm.com/fvp

Arm is Coming

Arm wants to enter the server market

FreeBSD

[2] S. McIntosh-Smith, J. Price, T. Deakin, and A. Poenaru, "Comparative Benchmarking of the First Generation of HPC-optimised ARM Processors on Isambard."

Arm is Coming

Arm wants to enter the server market

- Amazon AWS Graviton CPUs (based on Cortex-A72)

FreeBSD

[2] S. McIntosh-Smith, J. Price, T. Deakin, and A. Poenaru, "Comparative Benchmarking of the First Generation of HPC-optimised ARM Processors on Isambard."

Arm is Coming

Arm wants to enter the server market

- Amazon AWS Graviton CPUs (based on Cortex-A72)
- Arm Neoverse N1 CPUs (server-specific microarchitecture)

[2] S. McIntosh-Smith, J. Price, T. Deakin, and A. Poenaru, "Comparative Benchmarking of the First Generation of HPC-optimised ARM Processors on Isambard."

Arm is Coming

Arm wants to enter the server market

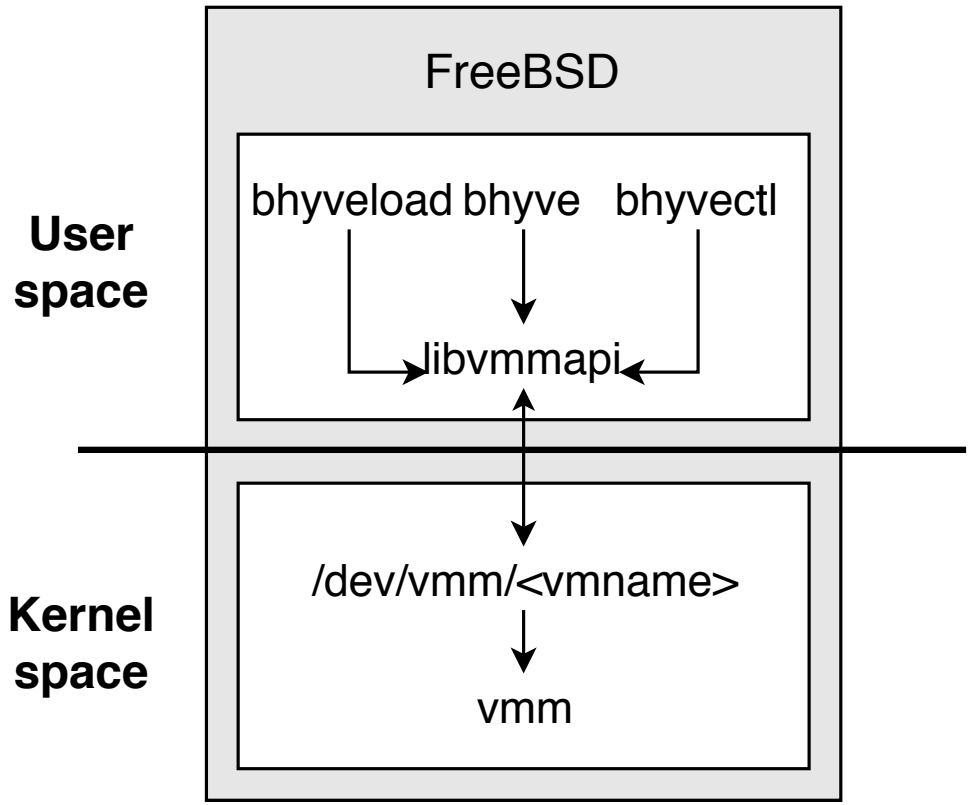
- Amazon AWS Graviton CPUs (based on Cortex-A72)
- Arm Neoverse N1 CPUs (server-specific microarchitecture)
- Cavium ThunderX2:

  > "The results presented in this paper demonstrate that Arm-based processors are now capable of providing levels of performance competitive with state-of-the-art offerings from the incumbent vendors, while significantly improving performance per Dollar."[2]

FreeBSD

[2] S. McIntosh-Smith, J. Price, T. Deakin, and A. Poenaru, "Comparative Benchmarking of the First Generation of HPC-optimised ARM Processors on Isambard."

bhyvearm64 Architecture



bhyveload

```
bhyveload     [-b <memory-base-address>]    \
              [-m <mem-size>]               \
              [-k <kernel-image>]           \
              [-l <load-address>]           \
              [-e <name=value>]             \
              <vmname>
```

Guest virtual hardware: devicetree

FreeBSD

bhyve

```
bhyve    -s  '0x200@0x7000#24:virtio-blk,virtio.img'\
         -s  '0x200@0x6000#23:virtio-net,tap0'   \
         -b                                      \
         <vmname>
```

FreeBSD

bhyvectl

```
bhyvectl --vm=<vmname> --destroy
```

FreeBSD

Outline

FreeBSD

Challenges

Challenges:

- The host must control the guest

FreeBSD

Challenges

Challenges:

- The host must control the guest

Motivation: Popek and Goldberg's safety requirement for virtualization

FreeBSD

Challenges

Challenges:

- The host must control the guest

Motivation: Popek and Goldberg's safety requirement for virtualization

Solution: use Exception Level 2 (EL2)

FreeBSD

EL2

CPU execution mode for virtualization

EL2

CPU execution mode for virtualization

- Shared general purpose registers

FreeBSD

EL2

CPU execution mode for virtualization

- Shared general purpose registers
- Different virtual address space

FreeBSD

EL2

CPU execution mode for virtualization

- Shared general purpose registers
- Different virtual address space
- New system registers that control EL2

FreeBSD

EL2

CPU execution mode for virtualization

- Shared general purpose registers
- Different virtual address space
- New system registers that control EL2
- New system registers that control EL1&EL0

FreeBSD

EL2 Limitations in Armv8.0

- Different system register names

EL2 Limitations in Armv8.0

- Different system register names
- Virtual address space that cannot be shared with userspace

FreeBSD

EL2 Limitations in Armv8.0

- Different system register names
- Virtual address space that cannot be shared with userspace
- Synchronous exceptions are normally routed to EL1

FreeBSD

EL2 Limitations in Armv8.0

- Different system register names
- Virtual address space that cannot be shared with userspace
- Synchronous exceptions are normally routed to EL1
- **It is optional**

FreeBSD

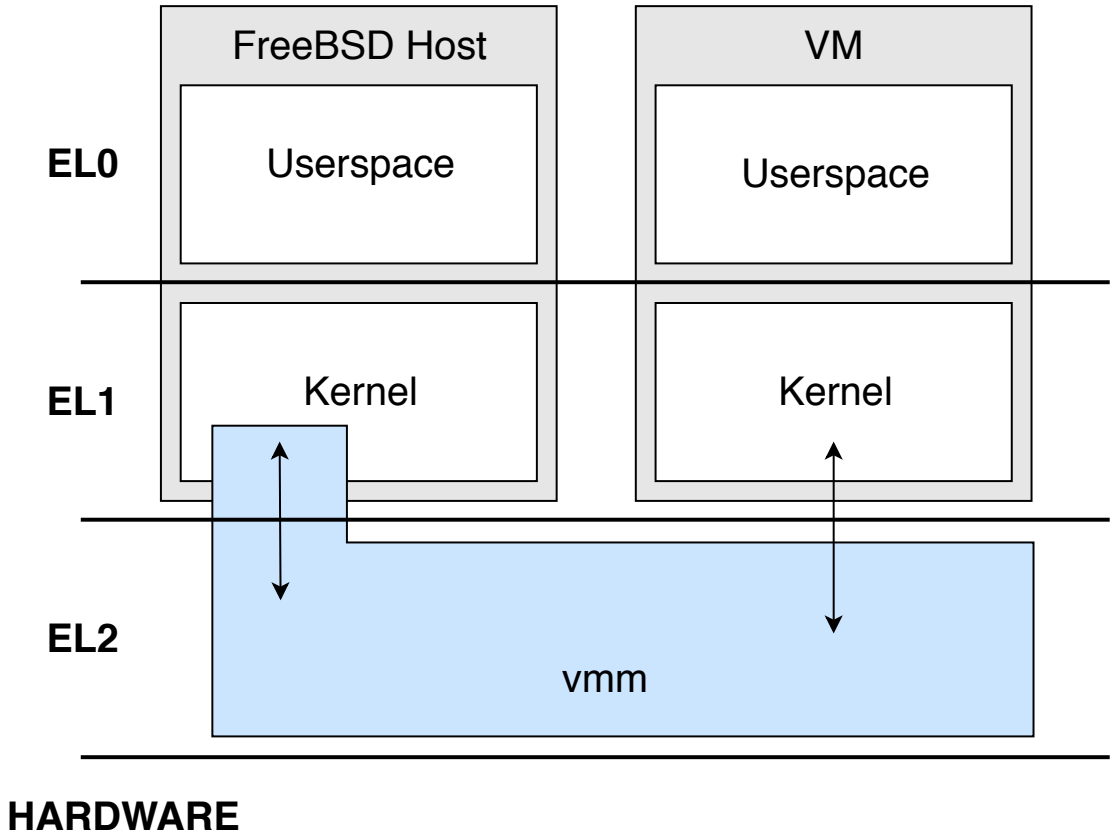
EL2 Limitations in Armv8.0

- Different system register names
- Virtual address space that cannot be shared with userspace
- Synchronous exceptions are normally routed to EL1
- **It is optional**

Conclusion: the host must run at EL1

FreeBSD

hypervisor Architecture



Challenges (continued)

Challenges:

- The host must control the guest
- Alternate execution between the host and the guest

FreeBSD

Challenges (continued)

Challenges:

- The host must control the guest
- Alternate execution between the host and the guest

Solution: save CPU state

FreeBSD

Challenges (continued)

Challenges:

- The host must control the guest
- Alternate execution between the host and the guest

Solution: save CPU state

CPU state:

- General purpose registers
- EL1 system registers
- EL2 system registers that control execution at EL1&EL0

FreeBSD

World Switch

World switches are executed:

- From guest to host
- From host to guest

FreeBSD

World Switch

World switches are executed:

- From guest to host
- From host to guest

Save CPU state:

- EL2 stack for host CPU state
- One `struct hypctx` for each virtual CPU

FreeBSD

Challenges (continued)

Challenges:

- The host must control the guest
- Alternate execution between the host and the guest
- The host must execute privileged instructions on behalf of the guest

FreeBSD

Challenges (continued)

Challenges:

- The host must control the guest
- Alternate execution between the host and the guest
- The host must execute privileged instructions on behalf of the guest

Motivation: Popek and Goldberg's fidelity requirement for virtualization

FreeBSD

Challenges (continued)

Challenges:

- The host must control the guest
- Alternate execution between the host and the guest
- The host must execute privileged instructions on behalf of the guest

Motivation: Popek and Goldberg's fidelity requirement for virtualization

Solution: host access to guest state

FreeBSD

Running in EL2

Goals:

- Execute code
- Share memory with host running in EL1

FreeBSD

Running in EL2

Goals:

- Execute code
- Share memory with host running in EL1

Running in EL2:

- Create function wrapper for the `HVC` assembly instruction

FreeBSD

Running in EL2

Goals:

- Execute code
- Share memory with host running in EL1

Running in EL2:

- Create function wrapper for the `HVC` assembly instruction
- Replace EL2 exception vector table

FreeBSD

Running in EL2

Goals:

- Execute code
- Share memory with host running in EL1

Running in EL2:

- Create function wrapper for the `HVC` assembly instruction
- Replace EL2 exception vector table
- Map the code in the EL2 address space

FreeBSD

Running in EL2

Goals:

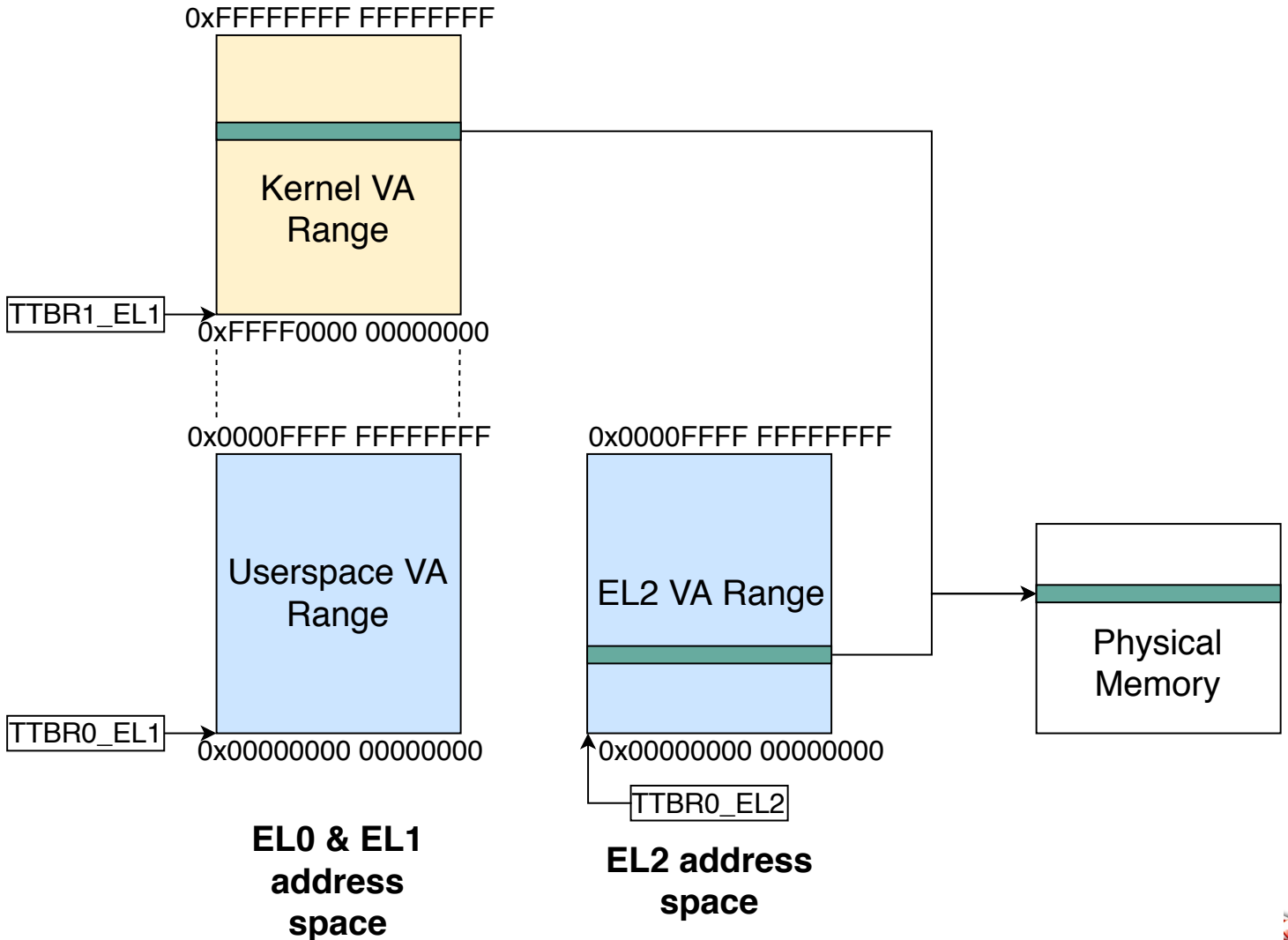
- Execute code
- Share memory with host running in EL1

Running in EL2:

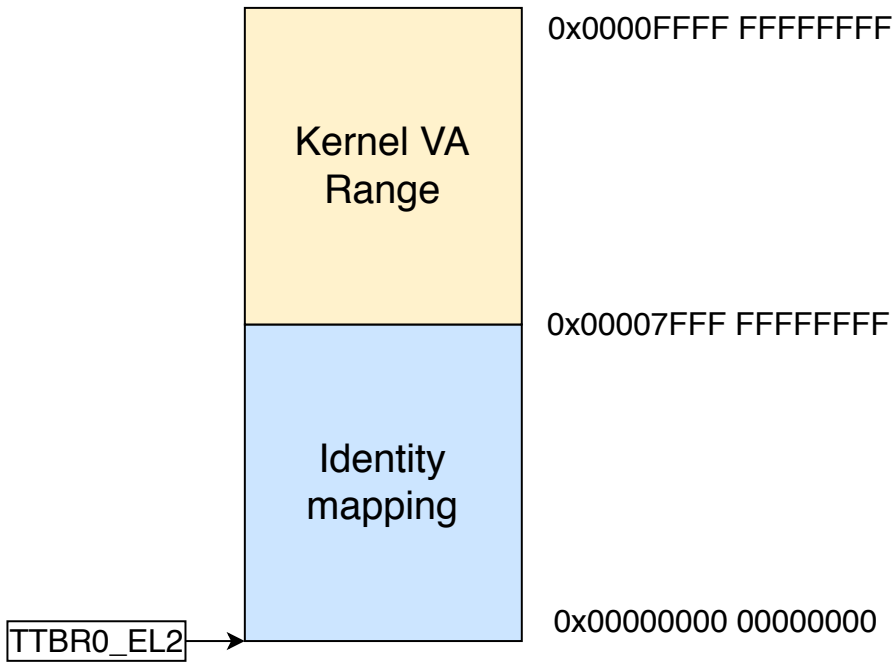
- Create function wrapper for the `HVC` assembly instruction
- Replace EL2 exception vector table
- Map the code in the EL2 address space
- Convention: first argument is the function to execute

FreeBSD

EL0, EL1 and EL2 address spaces



Sharing Memory



Outline

FreeBSD

Challenges

Challenges:

- The guest assumes it has access to the entire physical memory
- The host must control the entire physical memory

FreeBSD

Challenges

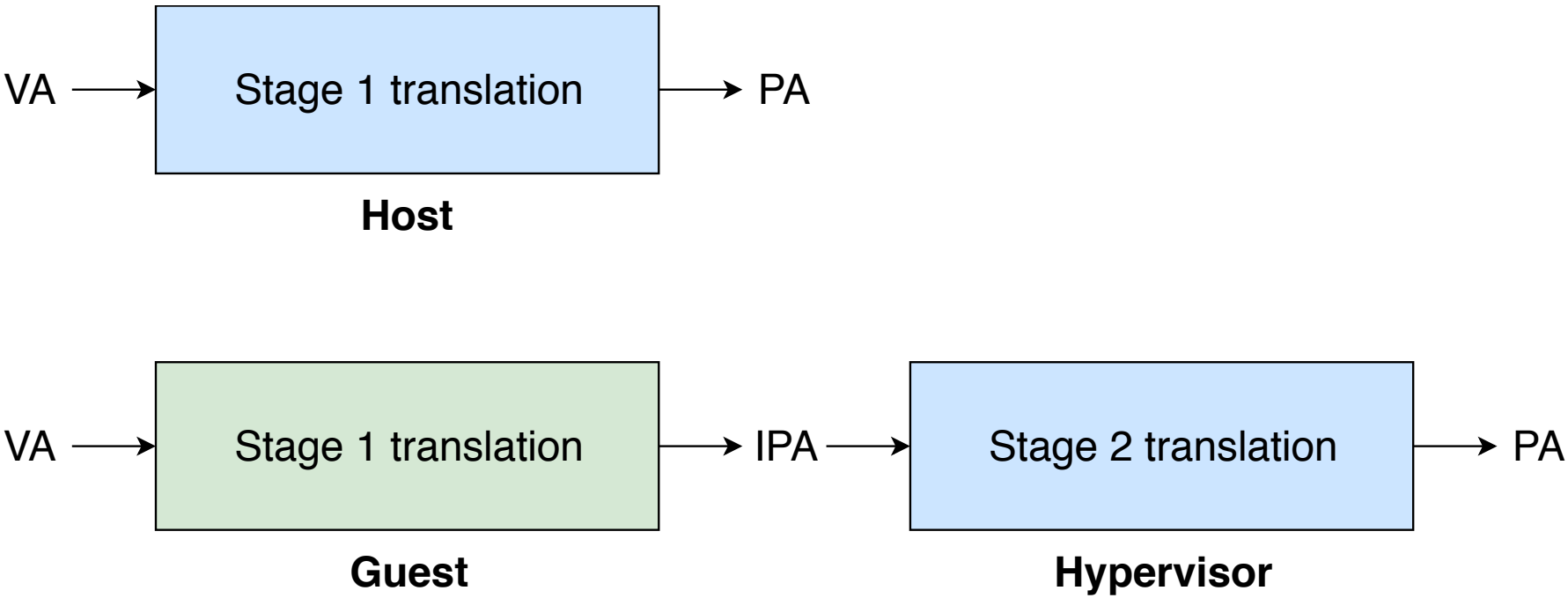
Challenges:

- The guest assumes it has access to the entire physical memory
- The host must control the entire physical memory

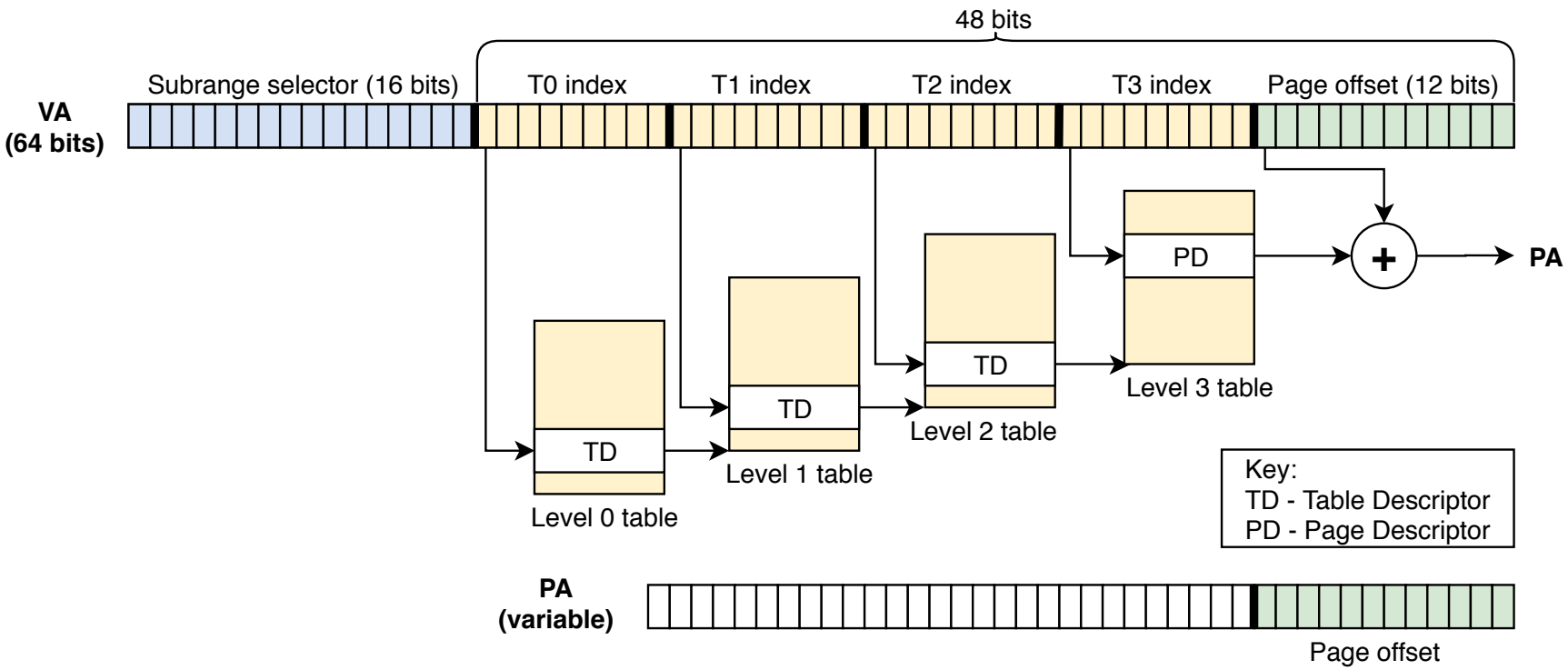
Solution: stage 2 translation

FreeBSD

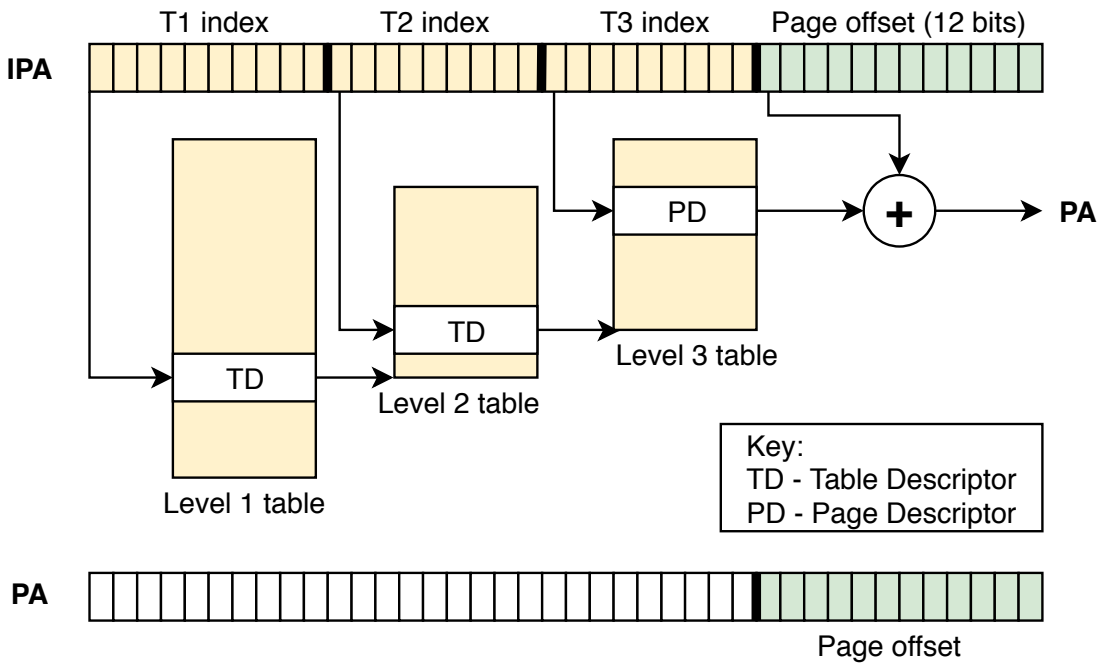
Translation Regimes



Stage 1 Translation with 4K Pages



Stage 2 Translation with 4K Pages



Stage 1 vs Stage 2

Differences between stage 1 an stage 2 translation:

- Stage 2 has 3 tables (vs 4)

Stage 1 vs Stage 2

Differences between stage 1 an stage 2 translation:

- Stage 2 has 3 tables (vs 4)
- Stage 2 starts at the level 1 table (vs level 0)

Stage 1 vs Stage 2

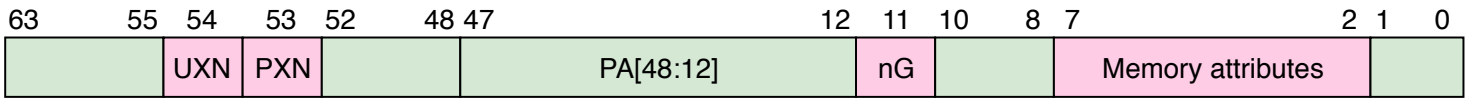
Differences between stage 1 an stage 2 translation:

- Stage 2 has 3 tables (vs 4)
- Stage 2 starts at the level 1 table (vs level 0)
- Stage 2 has a variable level 1 table size

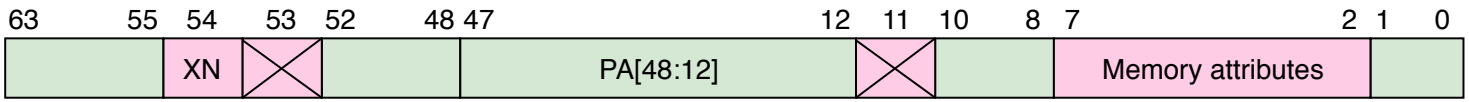
FreeBSD

4K Page Descriptors

Stage 1 page descriptor



Stage 2 page descriptor



Legend

Identical

Different

FreeBSD

Stage 1 vs Stage 2 (continued)

Differences between stage 1 and stage 2 translation:

- Stage 2 has 3 tables (vs 4)
- Stage 2 starts at the level 1 table (vs level 0)
- Stage 2 has a variable level 1 table size
- Different format for the page table entries

FreeBSD

Stage 1 vs Stage 2 (continued)

Differences between stage 1 and stage 2 translation:

- Stage 2 has 3 tables (vs 4)
- Stage 2 starts at the level 1 table (vs level 0)
- Stage 2 has a variable level 1 table size
- Different format for the page table entries

Solution: implement a new translation table format

Implementation

A new translation table type:

```
enum pmap_type {
        PT_STAGE1,
        PT_STAGE2,
        PT_INVALID,
};
struct pmap {
        ...
        enum pmap_type pm_type;
};
```

Existing functions take into account the table type:

```
int
pmap_enter(pmap_t pmap, ...)
{
        ...
        if (pmap->pm_type == PT_STAGE1) {
                /* Create stage 1 page */
        } else {
                /* Create stage 2 page */
        }
        ...
}
```

FreeBSD

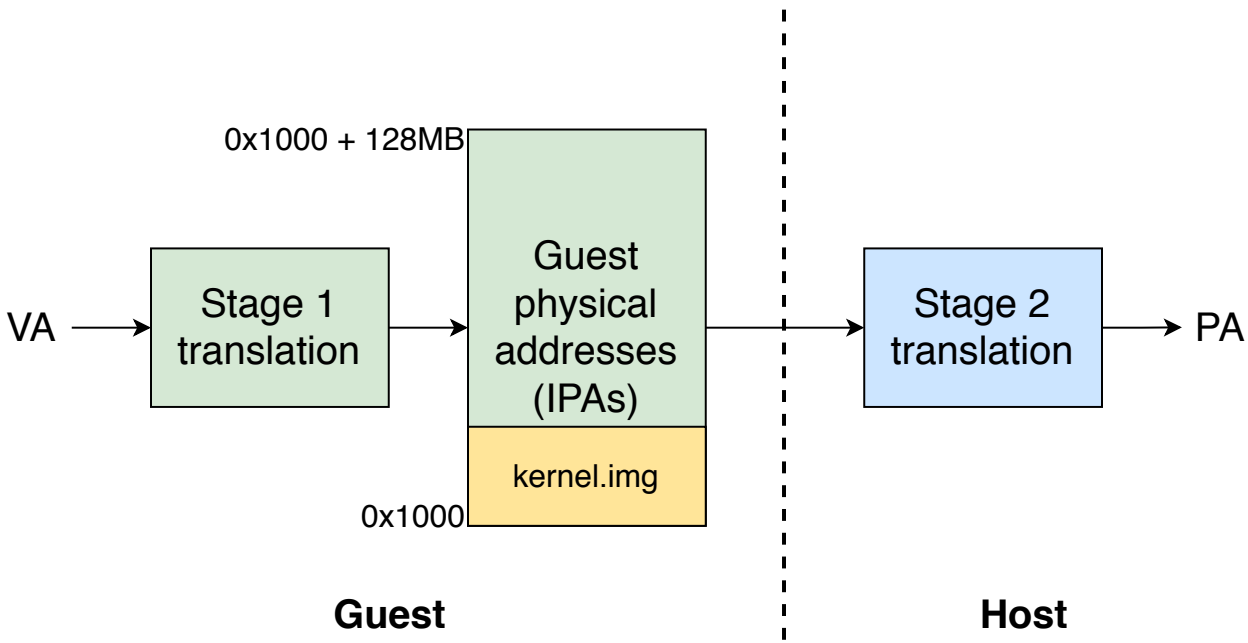
VM memory

```
bhyveload -k kernel.img -b 0x1000 -m 128MB \
          example_vm
```

FreeBSD

VM memory

```
bhyveload -k kernel.img -b 0x1000 -m 128MB \
          example_vm
```



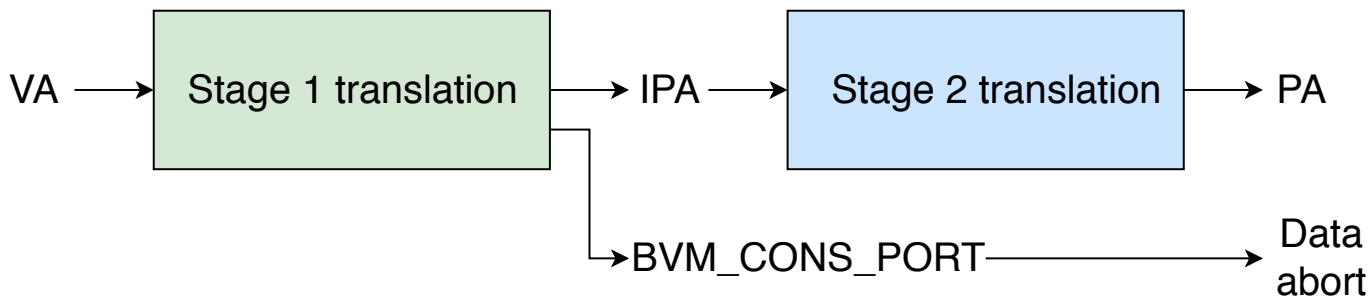
bvmconsole

```
bhyve -b example_vm
```

FreeBSD

bvmconsole

```
bhyve -b example_vm
```



Outline

FreeBSD

Future Work

Merge bhyvearm64

- Split bhyve for x86 into machine independent (MI) and dependent (MD) code
- User space improvements
- Hypervisor improvements
- Better validation

FreeBSD

MI/MD Split

Promote code reuse

FreeBSD

MI/MD Split

Promote code reuse

- Split libvmmapi (D17874)
- Split bhyve, bhyveload and bhyvectl
- Split the vmm kernel module

FreeBSD

User Space Improvements

- Use virtio-pci

User Space Improvements

- Use virtio-pci
- Emulate the NS16550A UART

User Space Improvements

- Use virtio-pci
- Emulate the NS16550A UART
- Emulate USB

FreeBSD

Hypervisor Improvements

- Implement SMP

FreeBSD

Hypervisor Improvements

- Implement SMP
- Implement Virtual Host Extensions (VHE) (Armv8.1)

FreeBSD

Hypervisor Improvements

- Implement SMP
- Implement Virtual Host Extensions (VHE) (Armv8.1)

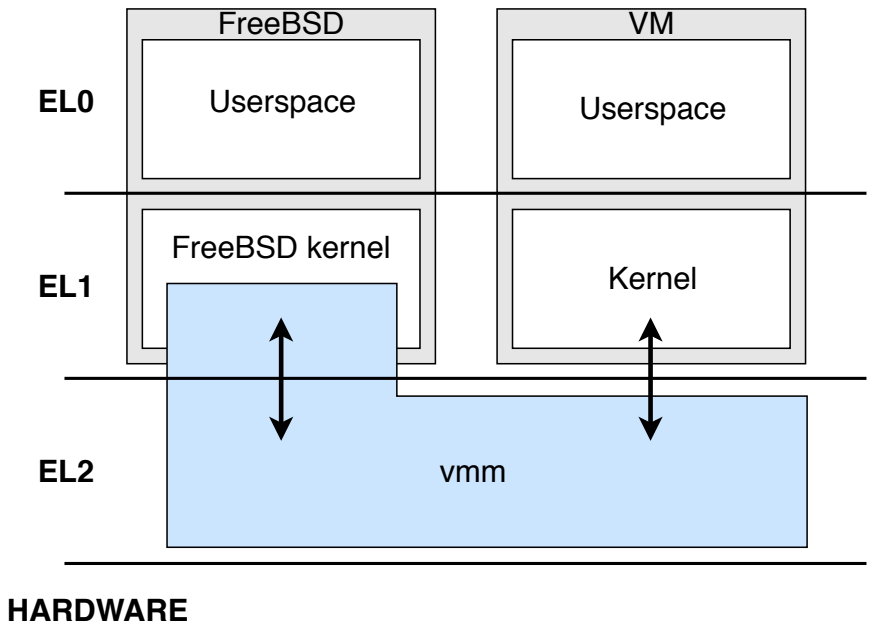


Figure: Without VHE

Hypervisor Improvements

- Implement SMP
- Implement Virtual Host Extensions (VHE) (Armv8.1)

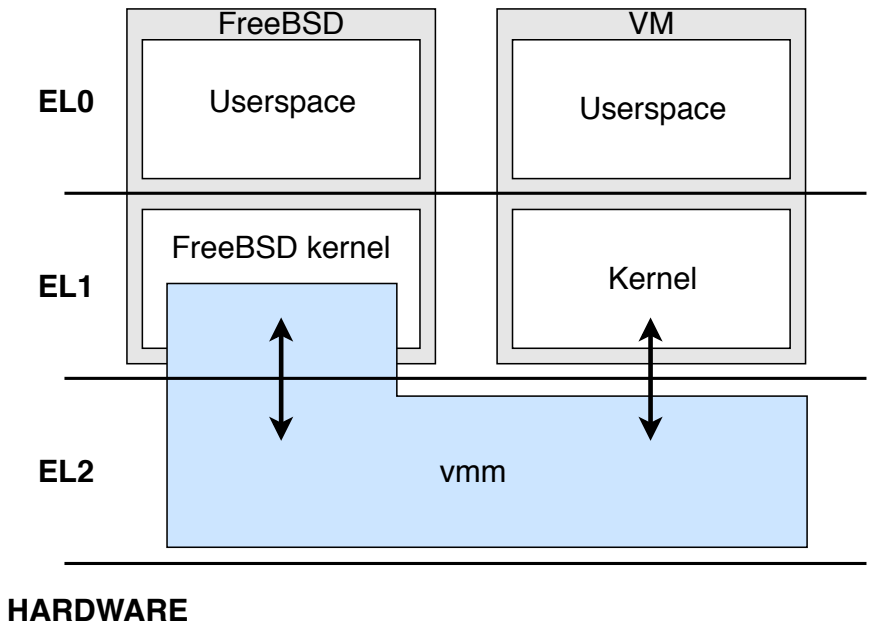


Figure: Without VHE

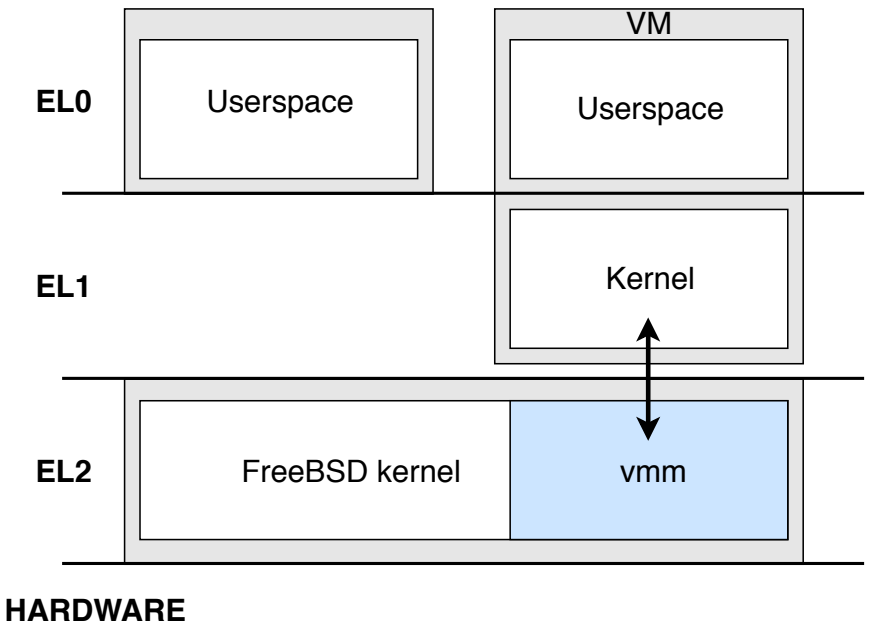


Figure: With VHE

Better Validation

- KVM-unit-tests[3]
  - Patches for using NS16550A UART and PSCI
  - Same boot protocol as the Linux kernel

FreeBSD

[3] https://www.linux-kvm.org/page/KVM-unit-tests

Better Validation

- KVM-unit-tests[3]
  - Patches for using NS16550A UART and PSCI
  - Same boot protocol as the Linux kernel
- Boot Linux as a guest

[3] https://www.linux-kvm.org/page/KVM-unit-tests

Outline

FreeBSD

Demo

Acknowledgements

- Mihai Carabaș (technical advisor)
- Mihai Darius (virtio-mmio)
- FreeBSD Foundation (financial support)

Questions?

Questions?

Thank you!

- Project repository:
  `https://github.com/FreeBSD-UPB/freebsd/tree/projects/bhyvearm64`
- Scripts and tutorial:
  `https://github.com/FreeBSD-UPB/bhyvearm64-utils`