HELLO "BAO" WORLD

VIRTUAL WORKSHOP

Co-promoted with:

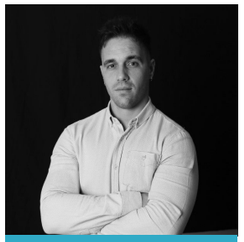
November 15th, 2023

Team

Today's Speakers

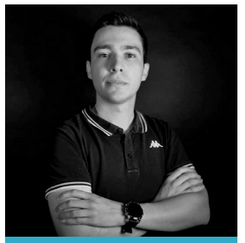
José Martins

David Cerdeira



Sandro Pinto

Daniel Oliveira



Diogo Costa

Bruno Sá

Afonso Santos

Miguel Silva

Cristiano Rodrigues

Workshop Goals

"I wish to do something Great and Wonderful,
but I must start by doing the little things like they were Great and Wonderful."
— Albert Einstein

1. **Understand Static Partitioning Hypervisors**
2. **Grasp Bao architecture and features**
3. **Learn how to:**
   - Setup a development environment for a Bao-based system
   - Create Bao VMs and configure memory, devices, interrupts
   - Add communication between VMs
   - Build Bao-based software stack from scratch
   - Bootstrap an emulated Bao-based system with Qemu
4. **Learn about:**
   - Advanced isolation mechanisms and features Bao implements that are beyond the scope of this workshop
   - Ongoing efforts and future directions

Agenda

Bao Hypervisor
Bao in a nutshell: architecture and features

System Setup
Setting up docker and development environment

I - Single-guest Bao setup
1x Baremetal with 2x vCPUs

II - Fine-tuned VM configuration
1x Baremetal with 4x vCPUs

III - Dual-guest Bao setup with FreeRTOS
1x Baremetal with 3x vCPUs, 1x FreeRTOS with 1x vCPU

IV - Dual-guest Bao setup with Linux
1x baremetal with 1x vCPUs; 1x Linux with 3 vCPUs

V - Inter-VM Communication
1x baremetal with 1x vCPUs; 1x Linux with 3 vCPUs, communicating through shared memory

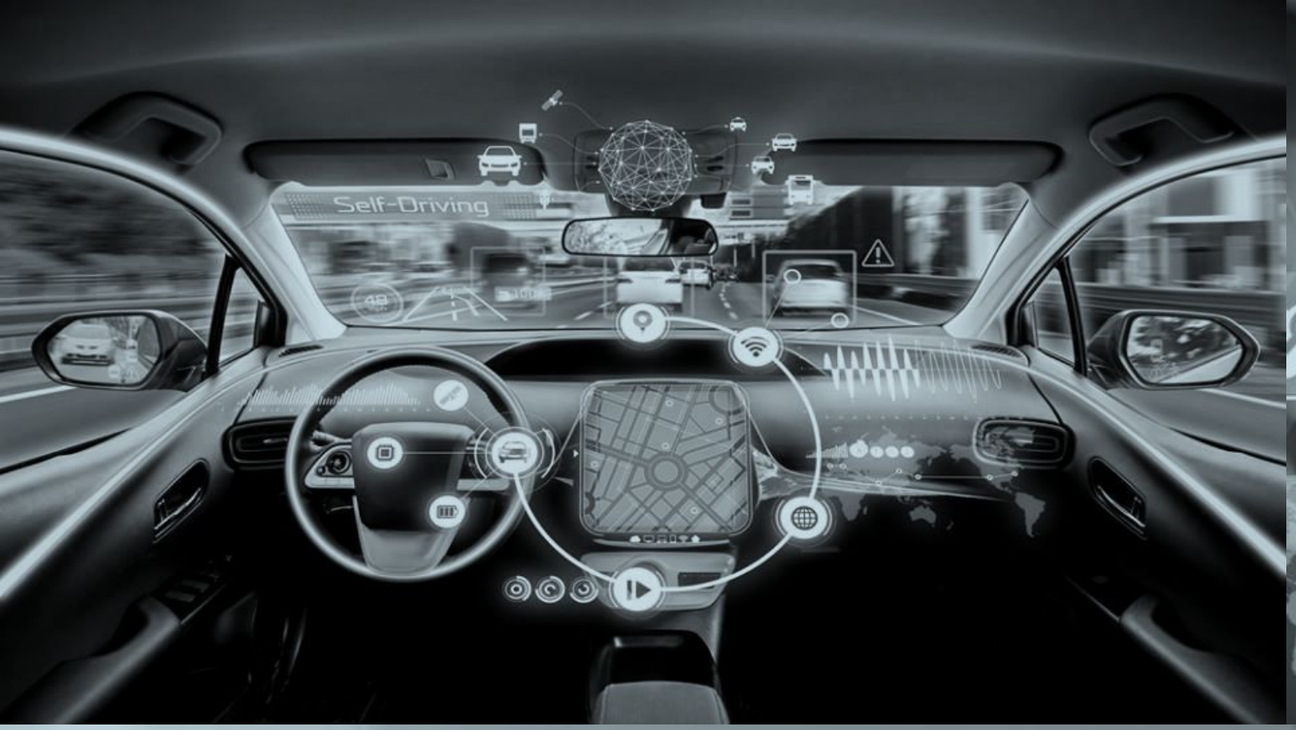
Bao Demos
Live demos on real hardware platforms (Raspberry Pi)

Advanced Bao Configuration
List of advanced features and configurations beyond this workshop

Conclusion and Roadmap
Takeaway points and roadmap

Bao Hypervisor

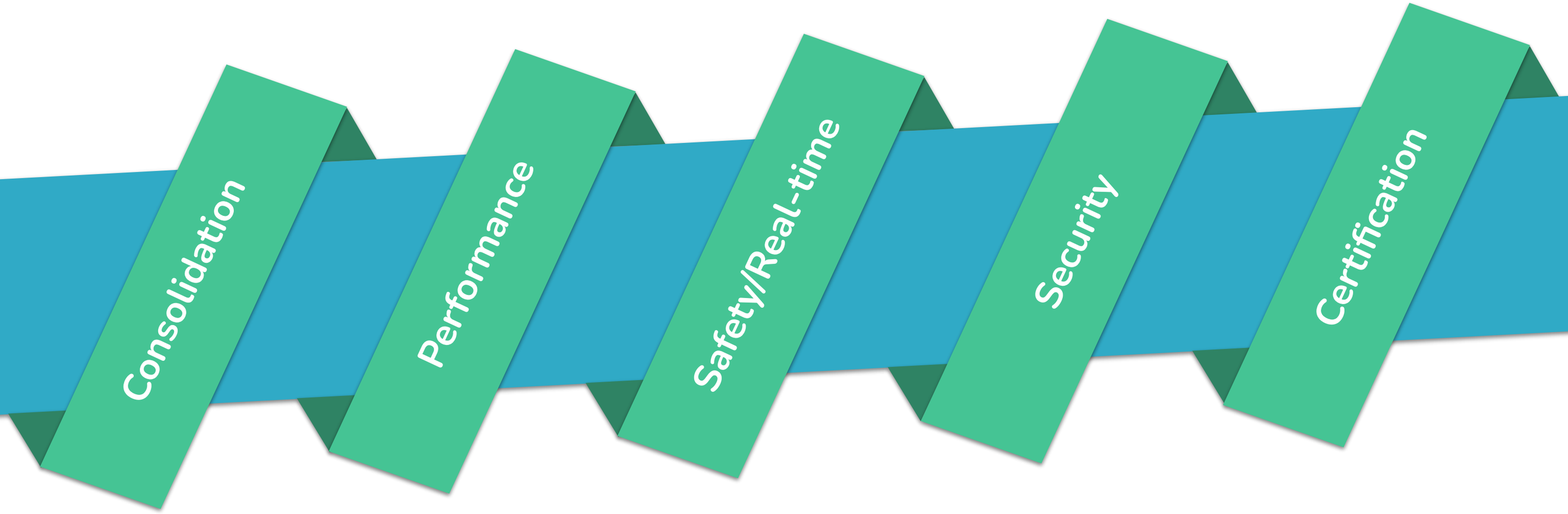
Bao in a nutshell: architecture and features


Self-Driving


MEDICINE
HEALTH
TREATMENT
DOCTOR
SURVEY
RECIPE
MEDICINE


Process
Concept
Automatic mode

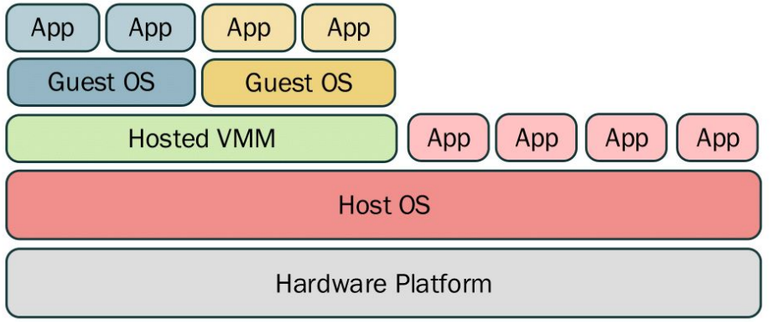
Mixed-Criticality Systems

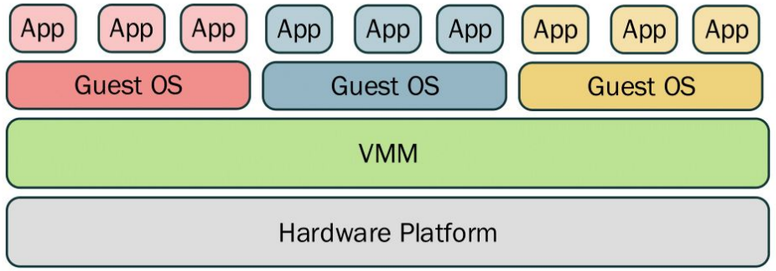


Virtualization?

Allows the execution of multiple Operating Systems in the same hardware platform.
*An **Hypervisor or VMM** is to an OS, as an OS is to a process.*

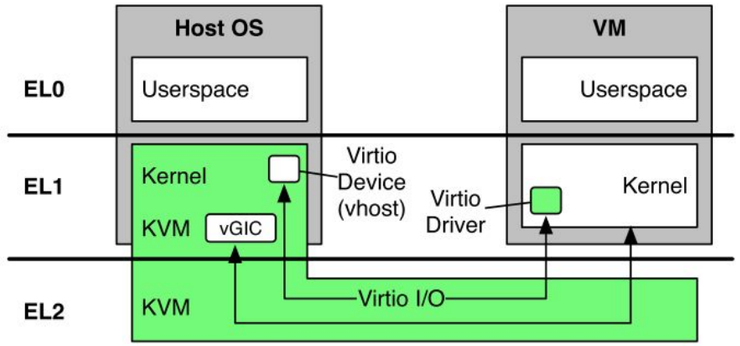
- **Main functions**
  - Resource Management
  - Abstraction
  - Protection / Isolation
- **The hypervisor provides a Virtual Machine (VM) abstraction for guest OSes**
- **Well-established**
  - Cloud (load balancing, power savings)
  - Desktops (cross-platform, systems development)





Hypervisors Spectrum

01



Source: C. Dall, "The Design, Implementation, and Evaluation of Software and Architectural Support for ARM Virtualization"

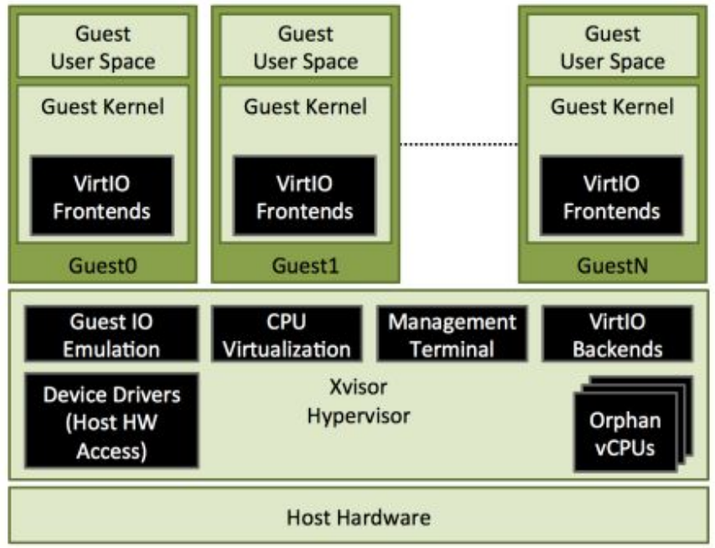
“Traditional” Hypervisors

- Large code base / TCB (Millions SLoC, e.g., Linux)
- Fully-featured (comm, networking, drivers)
- High-overhead I/O (emulation, paravirtualization)
- Cloud, server, and mobile
- **KVM, Xen**

02

- Intermediate code base / TCB (100K-10K SLoC)
- Soft real-time
- Scheduling and N:1 virtual-to-physical CPU mapping
- Generic embedded systems
- **XVisor, ACRN**

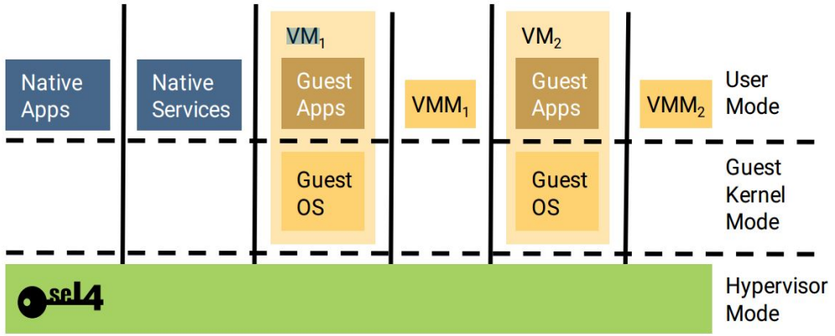
Embedded Hypervisors



Source: *A. Patel et al. “Embedded Hypervisor Xvisor: A comparative analysis”*

Hypervisors Spectrum

03



Source: *Gernot Heiser, "The seL4® Microkernel: An Introduction"*

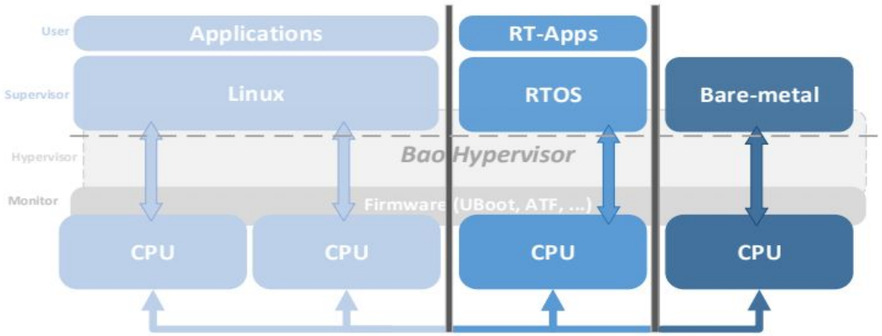
Microkernel Hypervisors

- Secure by design (formal verification)
- Capabilities and User-level VMMs
- Performance / Interrupt latency / Complexity
- Mobile, embedded, and mixed-criticality systems
- **seL4, NOVA**

- Small code base / TCB (5K-10K SLoC)
- Strong isolation & real-time
- Ineficiente resource usage
- Mixed-criticality systems
- **Jailhouse, Bao, Xen Dom0-less**

Static Partitioning Hypervisors

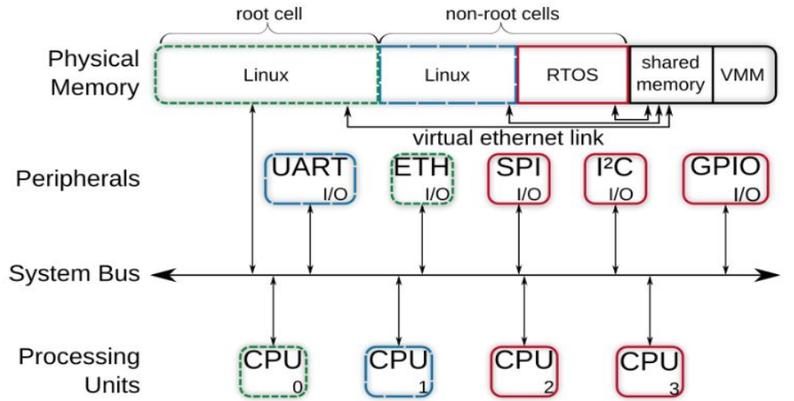
04



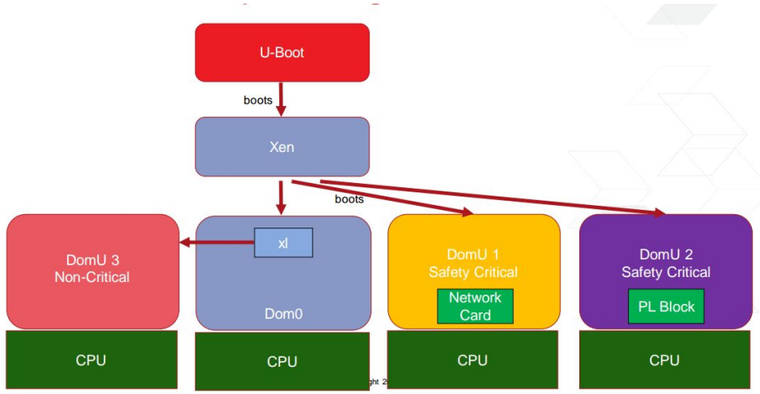
Source: *J. Martins et al. "Bao: A Lightweight Static Partitioning Hypervisor for Modern Multi-Core Embedded Systems"*

Static Partitioning Virtualization

- **Main Goals**
  - Consolidation
  - Strong isolation / Fault Encapsulation
  - Real-time guarantees
  - Minimal code base
- **No sharing, Static resource assignment, No abstractions**
  - 1:1 virtual-to-physical CPU mapping
  - No dynamic memory allocation
  - Device passthrough
  - Hardware interrupts
- **Reliance on HW virtualization**



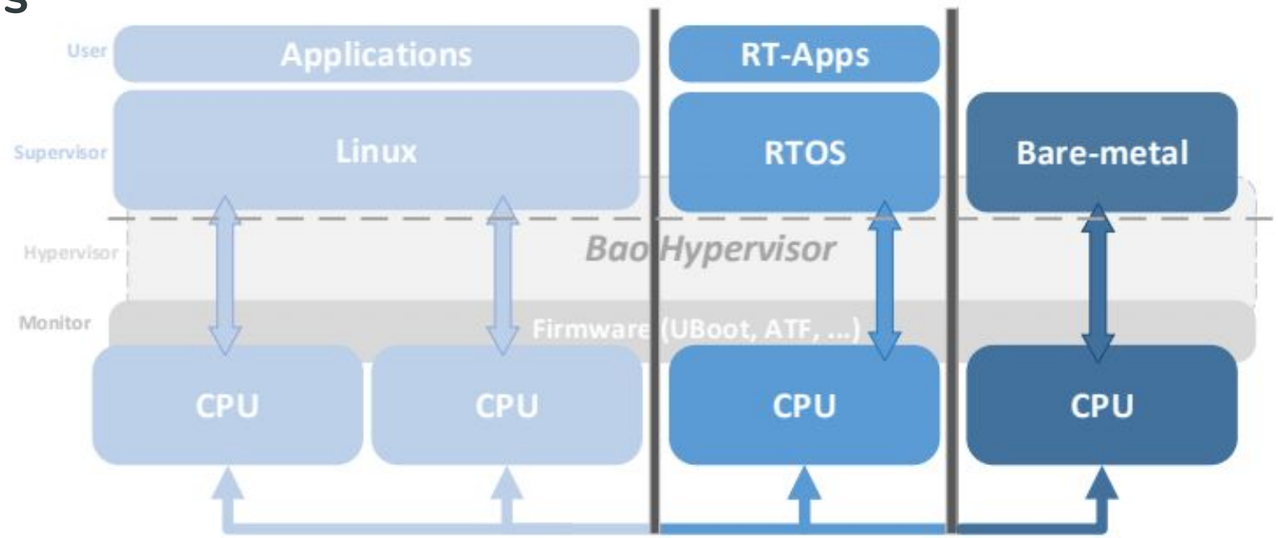
Source: *R. Ramsaur et al. "Look Mum, no VM Exits! (Almost)"*



Source: *S. Stabellini,"Static Partitioning with Xen"*

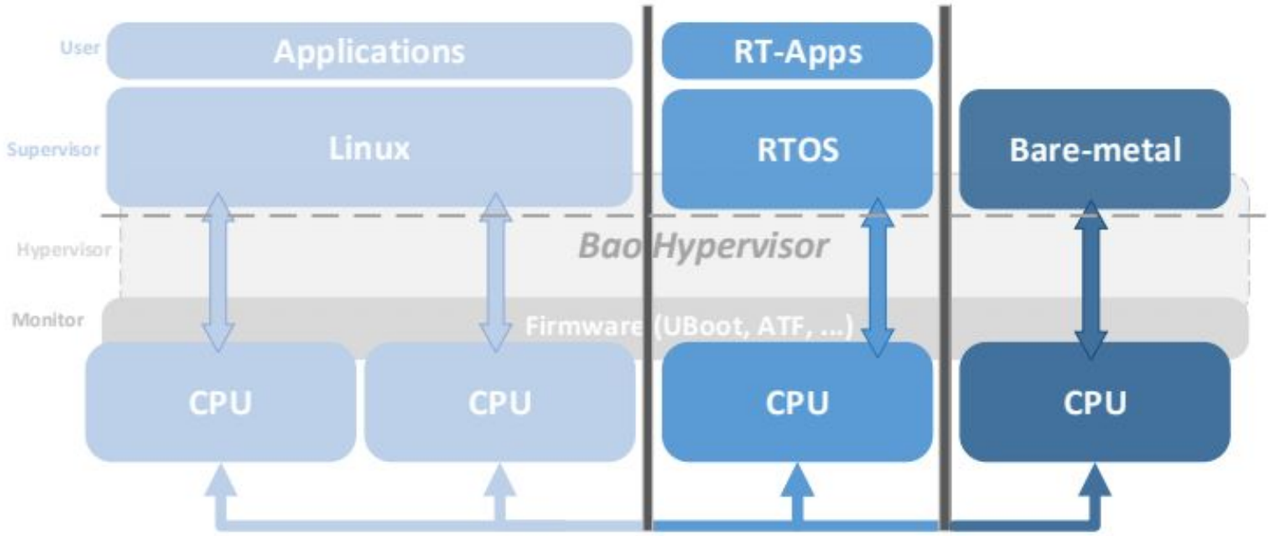
Bao Hypervisor

- **Mixed-criticality systems**
  - Meet real-time and security requirements
- **Type-1 / Bare-metal**
- **Support for MMU- and MPU-based systems**
- **Static Partitioning:**
  - 1:1 vCPU-to-pCPU mapping
  - Static memory assignment
  - Device Pass-through
  - Hardware interrupts
- **Inter-VM communication:**
  - Shared memory
  - Notifications (Inter-VM interrupts w/ access-control)
- **Dependencies**:
  - No external libraries
  - No privileged VMs/Oss
  - Simple drivers for UART log



Bao Hypervisor

- **Hardware-assisted:**
  - 2nd-stage translation
  - Interrupt virtualization support
  - IOMMU
- **Internal Isolation:**
  - Private CPU mappings
  - No mapping of VM memory
- **Superpages:**
  - Reduced TLB pressure
  - Reduced page-table memory
- **Cache Coloring (Partitioning)**:
  - Avoid interference on LLC
  - Guests and hypervisor
- **Permissive License: Apache v2.0**
  - https://github.com/bao-project/bao-hypervisor

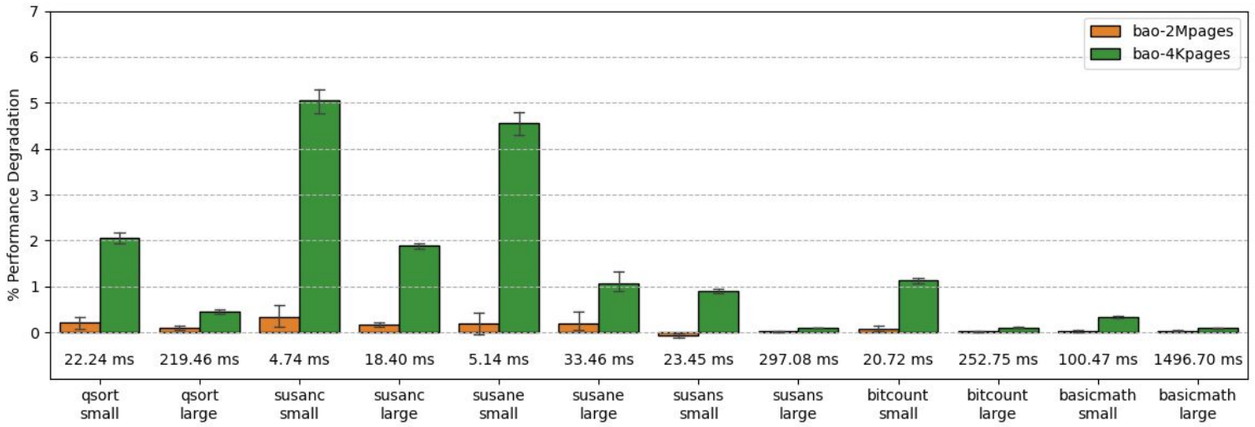


Minimal
Trusted Computing Base

~8K SLoC*

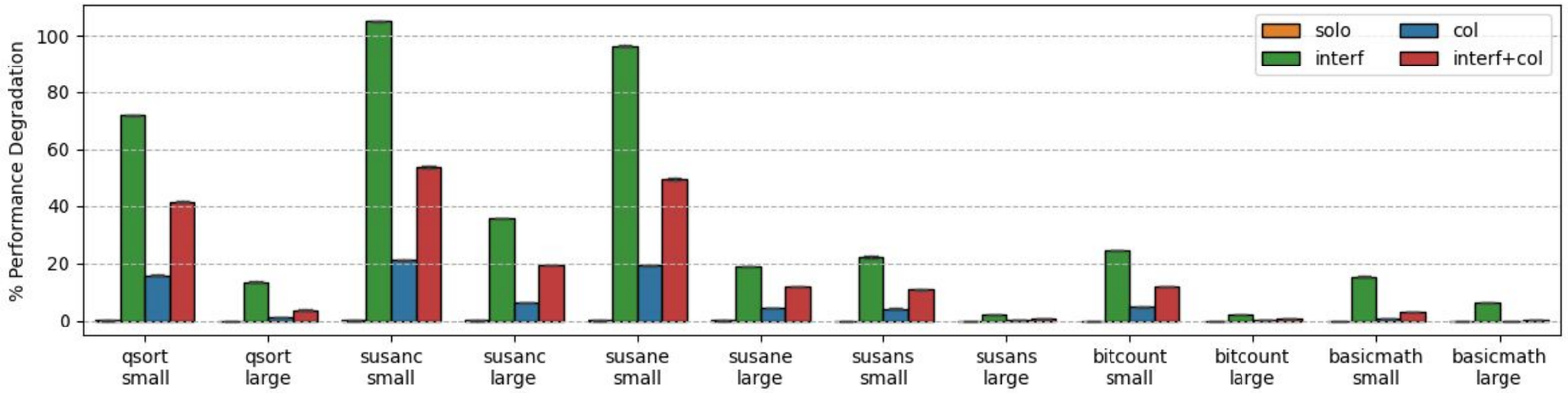
*Depending on target architecture and platform

Reduced
Performance Overhead



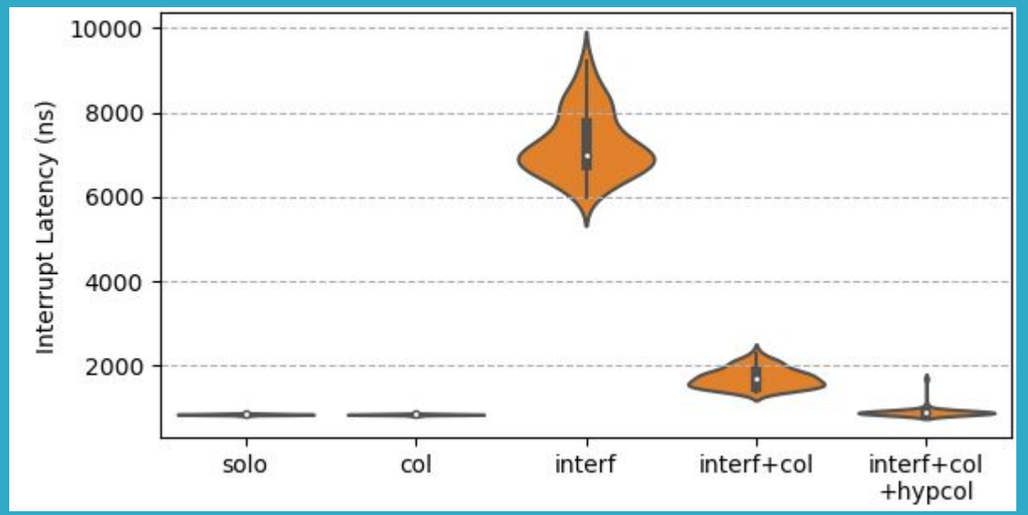
Interference Mitigation

Cache Coloring



Low IRQ Latency (sub µs)

Interference Resistant



2023 IEEE 29th Real-Time and Embedded Technology and Applications Symposium (RTAS)

Shedding Light on Static Partitioning Hypervisors for Arm-based Mixed-Criticality Systems

José Martins
Centro ALGORITMI/LASI, Universidade do Minho
jose.martins@dei.uminho.pt

Sandro Pinto
Centro ALGORITMI/LASI, Universidade do Minho
sandro.pinto@dei.uminho.pt

***Abstract*—In this paper, we aim to understand the properties and guarantees of static partitioning hypervisors (SPH) for Arm-based mixed-criticality systems (MCS). To this end, we performed a comprehensive empirical evaluation of popular open-source SPH, i.e., Jailhouse, Xen (Dom0-less), Bao, and seL4 CAmkES VMM, focusing on two key requirements of modern MCS: real-time and safety. The goal of this study is twofold. Firstly, to empower industrial practitioners with hard data to reason about the different trade-offs of SPH. Secondly, we aim to raise awareness of the research and open-source communities to the still open problems in SPH by unveiling new insights regarding lingering weaknesses. All artifacts will be open-sourced to enable independent validation of results and encourage further exploration on SPH.**

***Index Terms*—Virtualization, Static Partitioning, Hypervisor, Mixed-Criticality, Arm.**

I. Introduction

The explosion in the number of functional requirements in solutions, i.e., Jailhouse, Xen (Dom0-less), Bao, and the seL4 CAmkES virtual machine monitor (VMM). We drive our study based on two key requirements of modern MCS, i.e., real-time and safety, focusing on (i) performance, (ii) interrupt latency, (iii) inter-VM communication, (iv) boot time, and (v) code size. For each metric, we assess the effectiveness of the cache coloring technique, pervasive in SPH, for inter-VM interference mitigation.

The goal of this study is twofold. Firstly, we aim at empowering industrial practitioners with hard data to understand the trade-offs and limits of modern Arm-based SPH as well as how best configure these systems for their use case and requirements. For example, the use of superpages significantly decreases the number of TLB misses, resulting in negligible performance overhead (<1% without interference), but it is precluded by enabling page coloring, a widely adopted cache partitioning technique in these hypervisors. Also, experiments

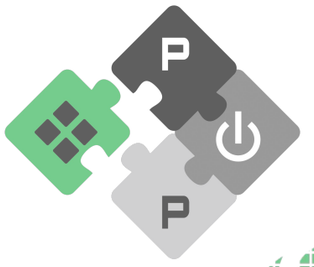
https://arxiv.org/abs/2303.11186
https://ieeexplore.ieee.org/document/10155684

Bao Support

- **Architectures:**
  - 32- and 64-bit
  - Armv7-A, Armv8-A, Armv8-R
  - RISC-V
- **Platform Highlight:**
  - Zynq US+ (ZCUx and Ultra96)
  - NXP i.MX8
  - Nvidia Tegra TX2
  - QEMU
  - Rocket @ ZCU
  - CVA6 @ Genesys2/VCU118
  - NXP S32Z/E
  - +++
- **Firmware:**
  - Arm Trusted Firmware (PSCI) on Arm
  - Supervisor Binary Interface (SBI) on RISC-V
- **Guests:**
  - Bare-metal
  - Linux / Android
  - RTOSs (Zephyr, FreeRTOS, Nuttx, Erika)
- **Tools:**
  - Lauterbach










Short Q&A

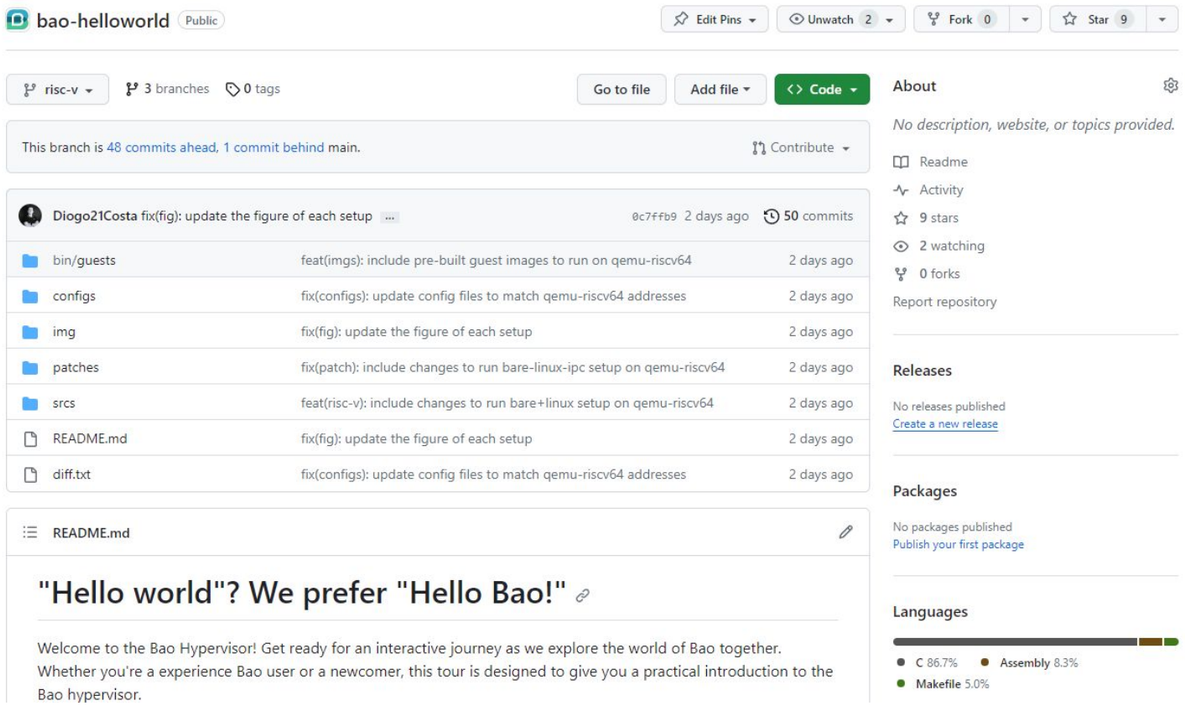
System Setup

Tools & Container

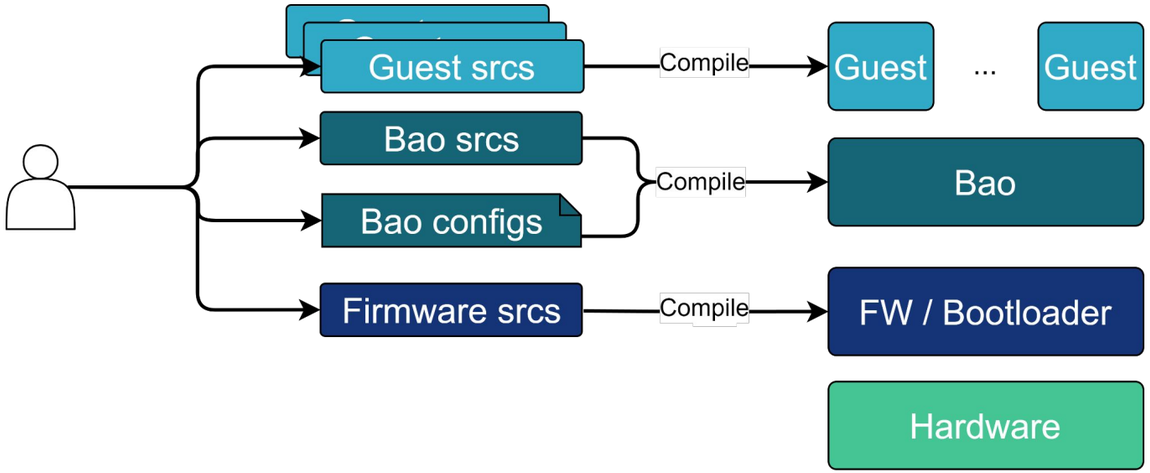
00 Cloning the repository

Clone the Bao Hello World repository

```
git clone https://github.com/bao-project/bao-helloworld.git
cd bao-helloworld
```



00 Overview of the workflow



00 Running the docker container

Use and run the Docker container

```
docker run --rm -it -u bao -v `pwd`:`pwd` -w `pwd` baoproject/bao:latest
```

00 How to setup work directory?

Set the **CROSS_COMPILE** environment variable

```
export CROSS_COMPILE=riscv64-unknown-elf-
```

Define environment variables for directories

```
export ROOT_DIR=$(realpath .)
export PRE_BUILT_IMGS=$ROOT_DIR/bin
export PATCHES_DIR=$ROOT_DIR/patches
export SETUP_BUILD=$ROOT_DIR/build
export BUILD_GUESTS_DIR=$SETUP_BUILD/guests
export BUILD_BAO_DIR=$SETUP_BUILD/bao
export TOOLS_DIR=$ROOT_DIR/tools
```

00 How to setup work directory?

```
root@a992b95c6080:/bao-helloworld# tree . -L 3
.
|-- README.md
|-- bin
|   `-- guests
|       |-- baremetal-freeRTOS-setup
|       |-- baremetal-linux-setup
|       |-- baremetal-linux-shmem-setup
|       `-- baremetal-setup
|-- build
|   |-- bao
|   |-- firmware
|   `-- guests
|-- configs
|   |-- baremetal-freeRTOS.c
|   |-- baremetal-linux-shmem.c
|   |-- baremetal-linux.c
|   `-- baremetal.c
|-- diff.txt
|-- img
|   |-- baremetal-freertos-setup.svg
|   |-- baremetal-linux-ipc-setup.svg
|   |-- baremetal-linux-setup.svg
|   `-- baremetal-setup.svg
|-- patches
|   |-- baremetal.patch
|   |-- baremetal_shmem.patch
|   `-- freeRTOS.patch
|-- srcs
|   |-- buildroot
|   |   |-- aarch32.config
|   |   |-- aarch64-rel.config
|   |   |-- aarch64.config
|   |   `-- riscv64.config
|   |-- configs
|   |   |-- aarch64.config
|   |   |-- base.config
|   |   |-- imx8qm.config
|   |   |-- linux.config
|   |   |-- rpi4.config
|   |   `-- tx2.config
|   |-- devicetrees
|   |   |-- qemu-aarch64-virt
|   |   `-- qemu-riscv64-virt
|   |-- loader
|   |   |-- LICENSE
|   |   |-- Makefile
|   |   |-- aarch32.S
|   |   |-- aarch64.S
|   |   |-- loader_aarch32.ld
|   |   |-- loader_aarch64.ld
|   |   |-- loader_riscv64.ld
|   |   `-- riscv64.S
|   `-- patches
|       |-- rel_imx_5.4.24_2.1.0
|       |-- v5.11
|       `-- v6.1
`-- tools
    `-- bin
```

Create working directories

```
mkdir -p $BUILD_GUESTS_DIR
mkdir -p $BUILD_BAO_DIR
mkdir -p $TOOLS_DIR/bin
```

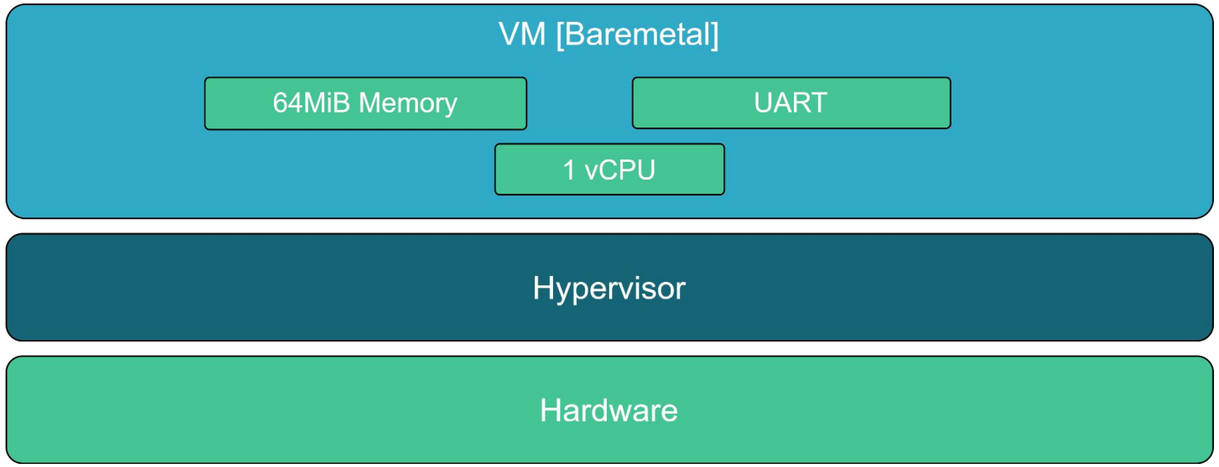
Single-guest Bao setup

1x Baremetal with 1x vCPUs

01 What are we building?

Single-guest configuration

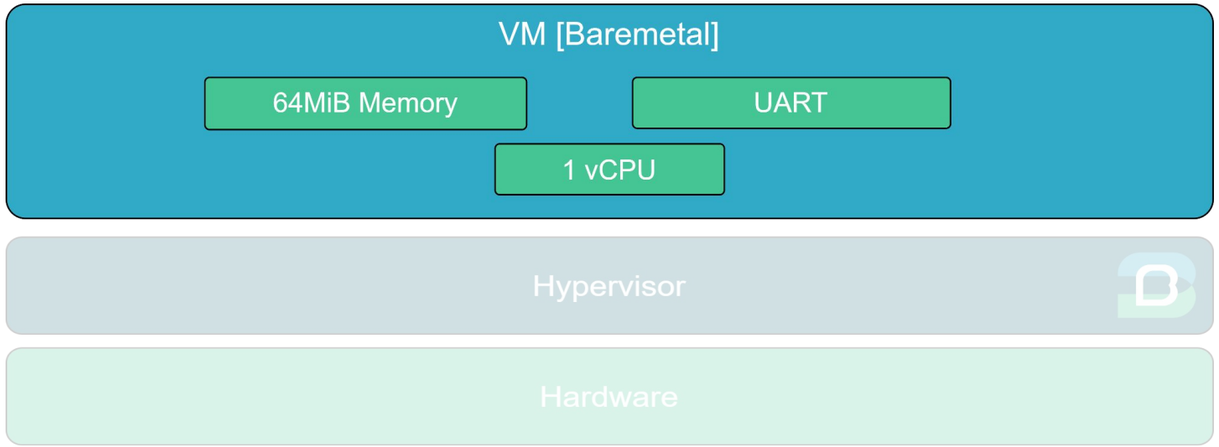
- Baremetal application
- 1 vCPUs
- 64 MiB Memory
- UART



02 How to setup the guest?

configs/baremetal.c

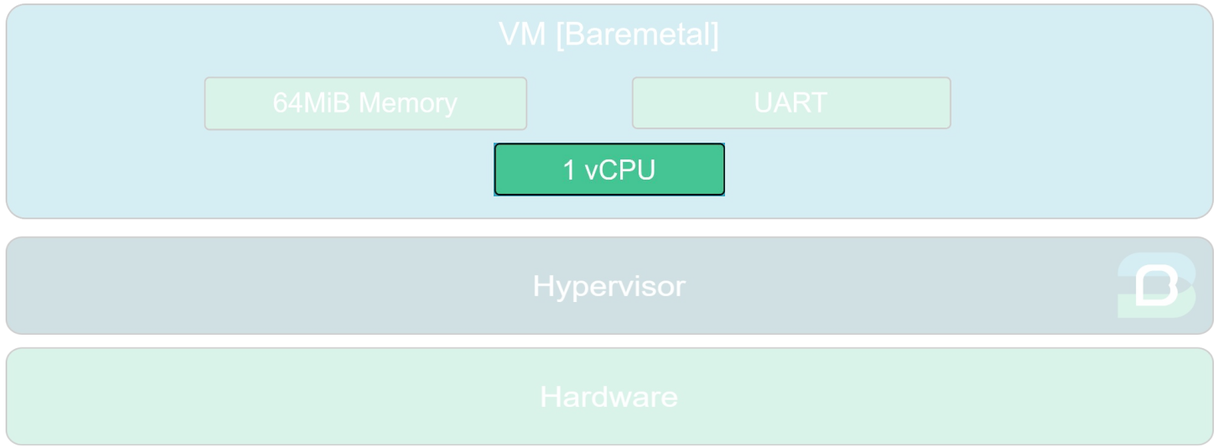
```
#include <config.h>

VM_IMAGE(baremetal_image,
XSTR(BAO_WRKDIR_IMGS/baremetal-setup/baremetal.bin));

struct config config = {
    .vmlist_size = 1,
    .vmlist = {
        {
            .image = VM_IMAGE_BUILTIN(baremetal_image, 0x80200000),

            .entry = 0x80200000,


(...)
```



02 How to setup the guest?

configs/baremetal.c

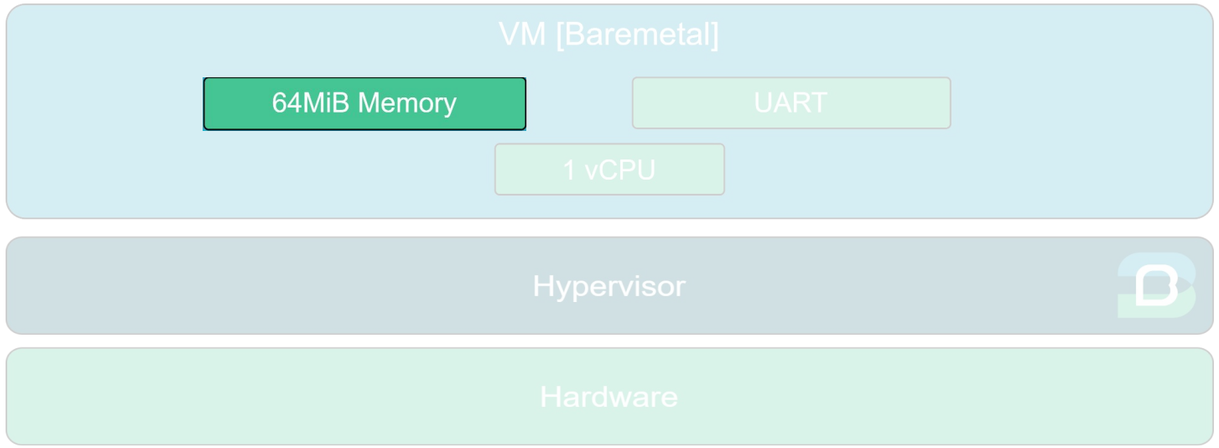
```
(...)

            .entry = 0x80200000,

             .platform = {
                 .cpu_num = 1,


(...)
```



02 How to setup the guest?

configs/baremetal.c

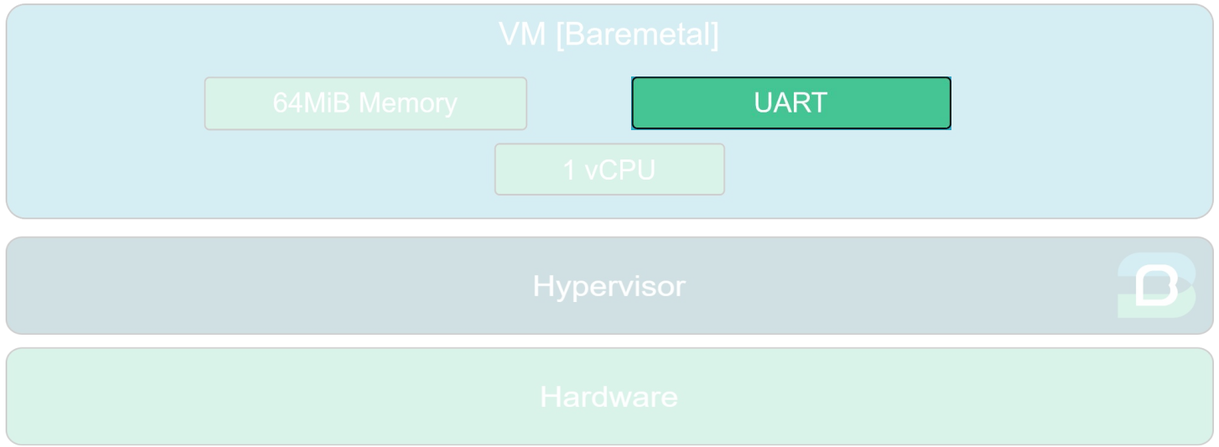
```
(...)

            .entry = 0x80200000,

             .platform = {
                 .cpu_num = 1,

                 .region_num = 1,
                 .regions =  (struct vm_mem_region[]) {
                     {
                         .base = 0x80200000,
                         .size = 0x4000000
                     }
                 },
```



02 How to setup the guest?

configs/baremetal.c

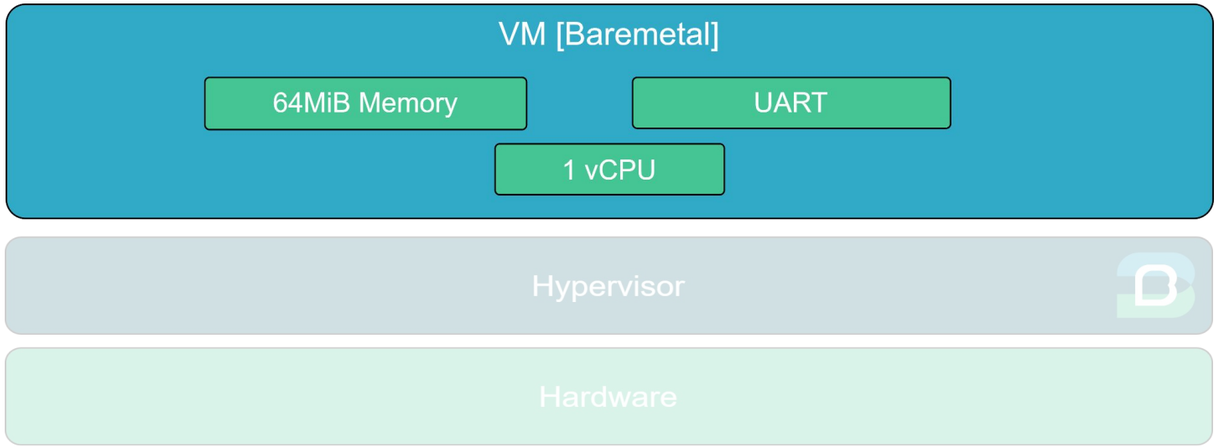
```
                .dev_num = 1,
                .devs =  (struct vm_dev_region[]) {
                    {
                        /* 8250 */
                        .pa = 0x10000000,
                        .va = 0x10000000,
                        .size = 0x1000,
                        .interrupt_num = 1,
                        .interrupts = (irqid_t[]) {10}
                    },
                },
(...)
                .arch =  {
                        .plic_base = 0xc000000,
                },
```



03 How to build the guest?

Define an environment variable for the baremetal guest application:

```
export BAREMETAL_SRCS=$ROOT_DIR/baremetal
```



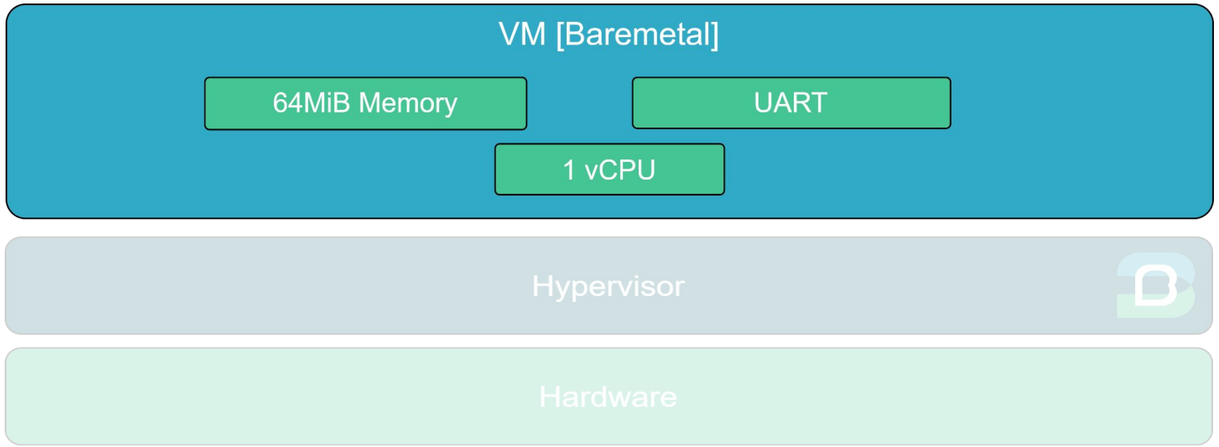
03 How to build the guest?

Define an environment variable for the baremetal guest application:

```
export BAREMETAL_SRCS=$ROOT_DIR/baremetal
```

Clone the baremetal:

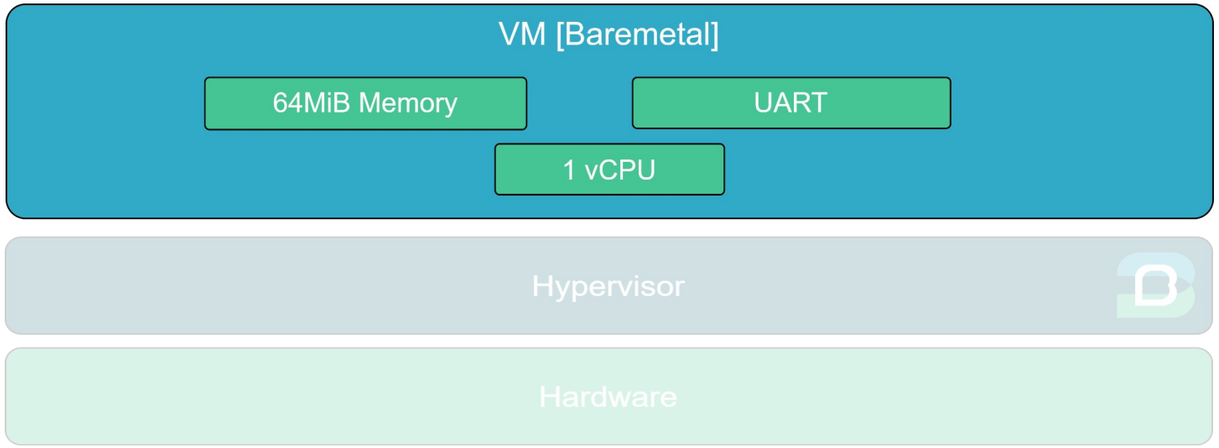
```
git clone https://github.com/bao-project/bao-baremetal-guest.git \
    --branch demo $BAREMETAL_SRCS
```



03 How to build the guest?

Compile the baremetal

```
git -C $BAREMETAL_SRCS apply $PATCHES_DIR/baremetal.patch
make -C $BAREMETAL_SRCS PLATFORM=qemu-riscv64-virt
```



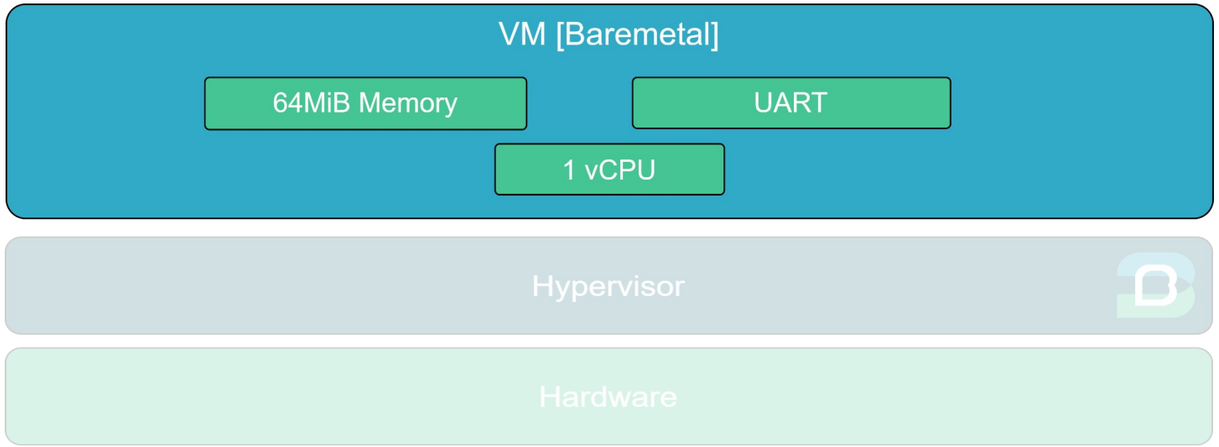
03 How to build the guest?

Compile the baremetal

```
git -C $BAREMETAL_SRCS apply $PATCHES_DIR/baremetal.patch
make -C $BAREMETAL_SRCS PLATFORM=qemu-riscvh64-virt
```

Move the binary file (baremetal.bin)

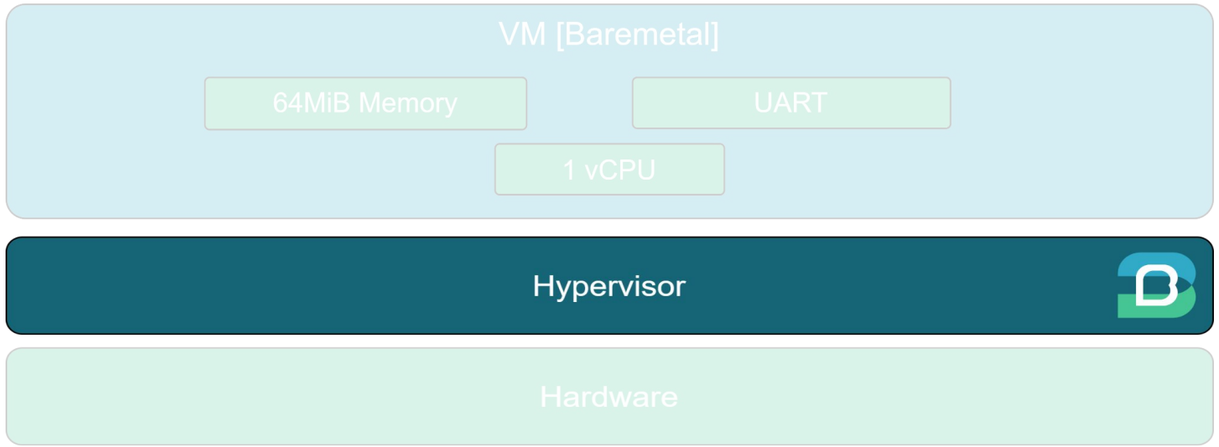
```
mkdir -p $BUILD_GUESTS_DIR/baremetal-setup
cp $BAREMETAL_SRCS/build/qemu-riscv64-virt/baremetal.bin \
    $BUILD_GUESTS_DIR/baremetal-setup/baremetal.bin
```



03 How to build Bao?

Define an environment variable for the Bao hypervisor

```
export BAO_SRCS=$ROOT_DIR/bao
```



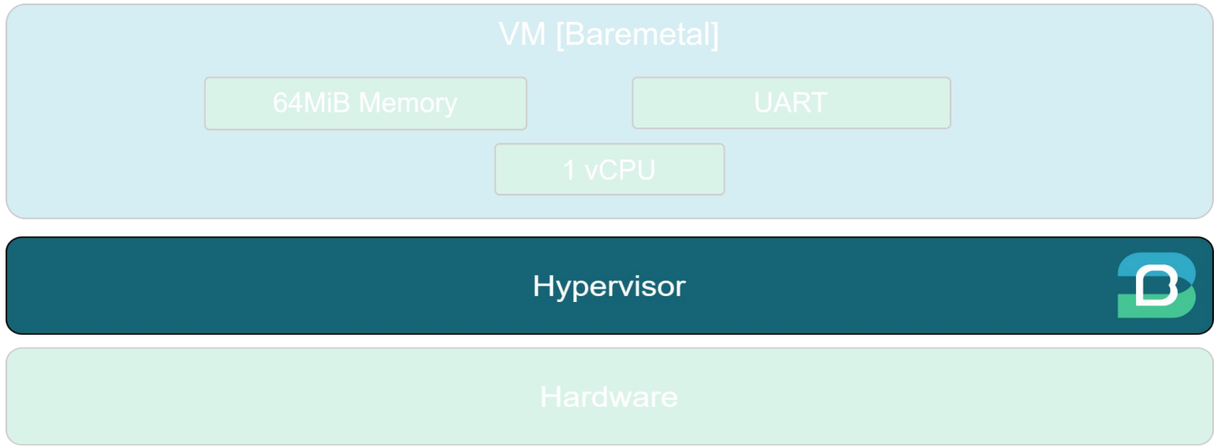
03 How to build Bao?

Define an environment variable for the Bao hypervisor:

```
export BAO_SRCS=$ROOT_DIR/bao
```

Clone the hypervisor

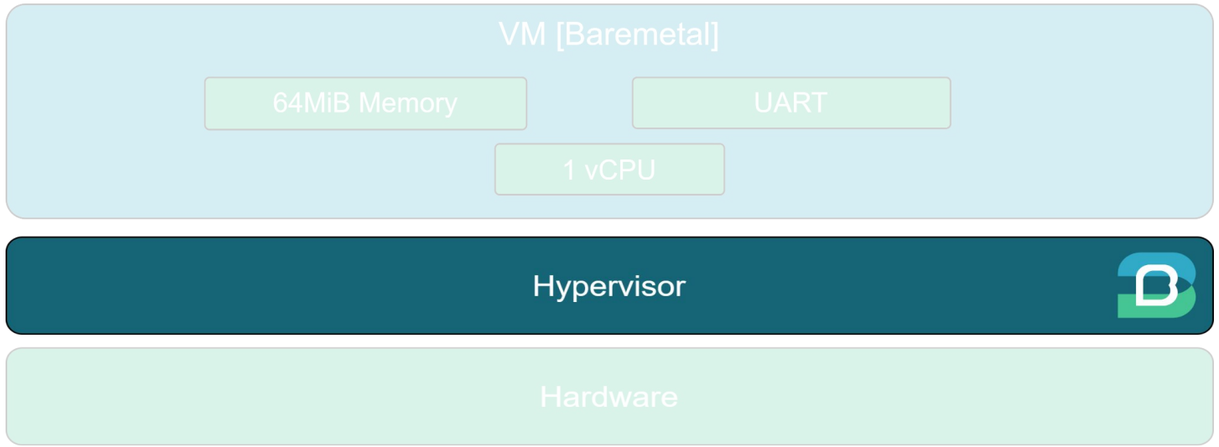
```
git clone https://github.com/bao-project/bao-hypervisor \
    $BAO_SRCS --branch demo
```



03 How to build Bao?

Compile the hypervisor

```
make -C $BAO_SRCS \
    PLATFORM=qemu-riscv64-virt \
    CONFIG_REPO=$ROOT_DIR/configs \
    CONFIG=baremetal \
    CPPFLAGS=-DBAO_WRKDIR_IMGS=$BUILD_GUESTS_DIR
```

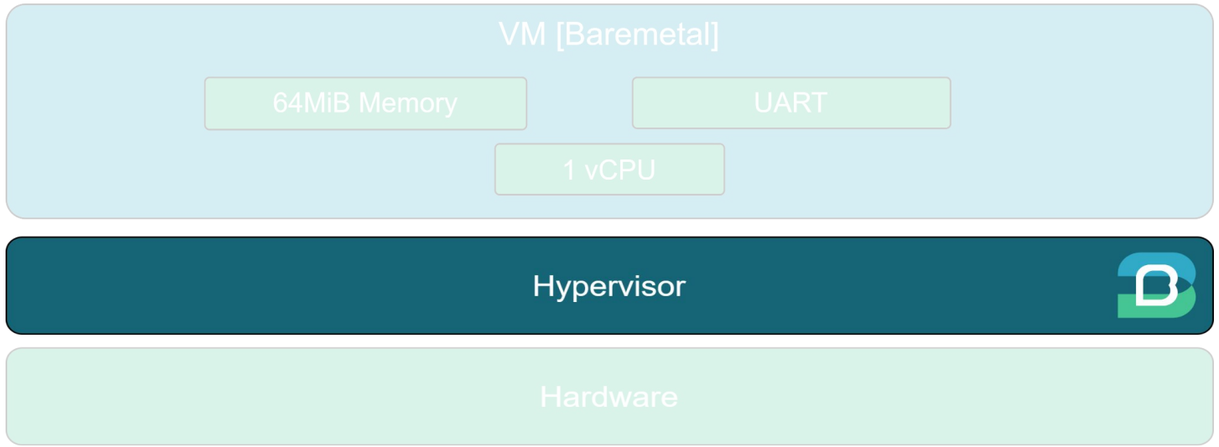


03 How to build Bao?

Compile the hypervisor

```
make -C $BAO_SRCS \
    PLATFORM=qemu-riscv64-virt \
    CONFIG_REPO=$ROOT_DIR/configs \
    CONFIG=baremetal \
    CPPFLAGS=-DBAO_WRKDIR_IMGS=$BUILD_GUESTS_DIR
```

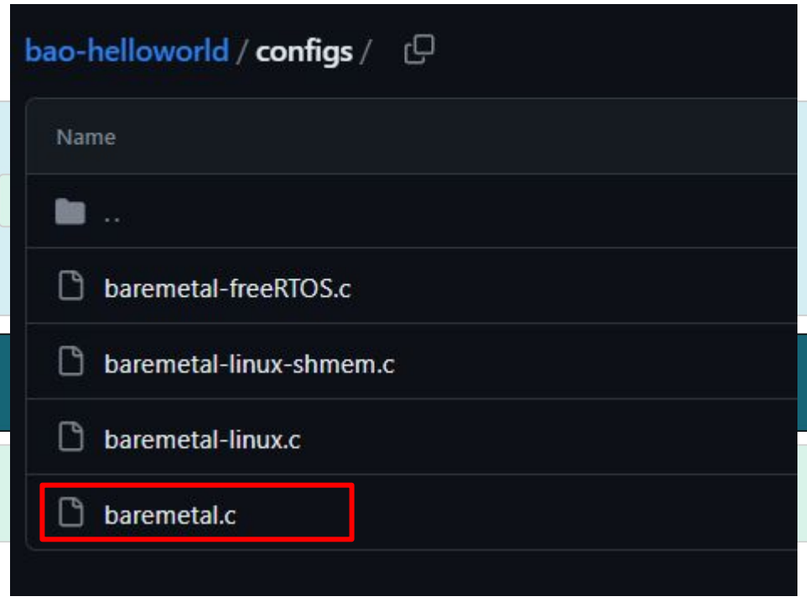
bao-helloworld / configs /



03 How to build Bao?

Compile the hypervisor

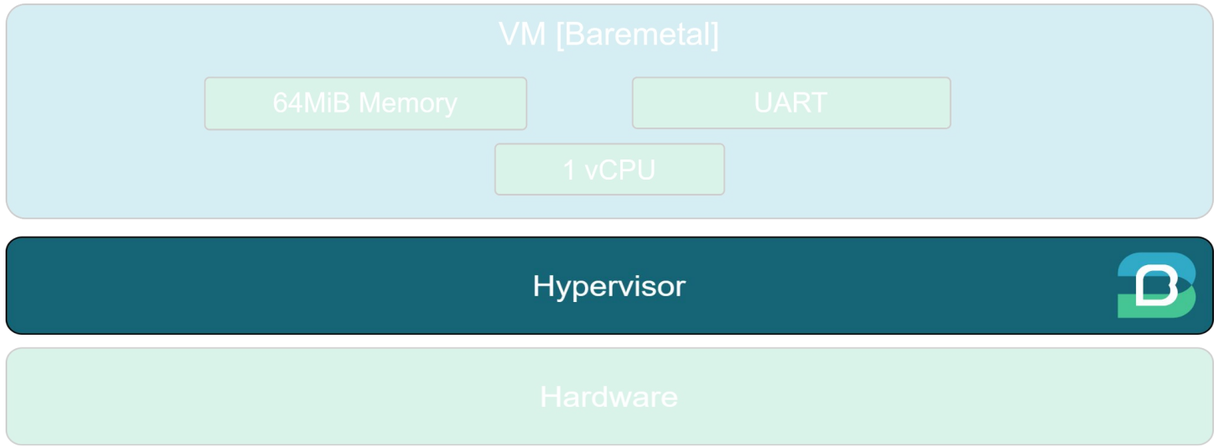
```
make -C $BAO_SRCS\
    PLATFORM=qemu-riscv64-virt\
    CONFIG_REPO=$ROOT_DIR/configs\
    CONFIG=baremetal\
    CPPFLAGS=-DBAO_WRKDIR_IMGS=$BUILD_GUESTS_DIR
```



03 How to build Bao?

Move the binary file (bao.bin)

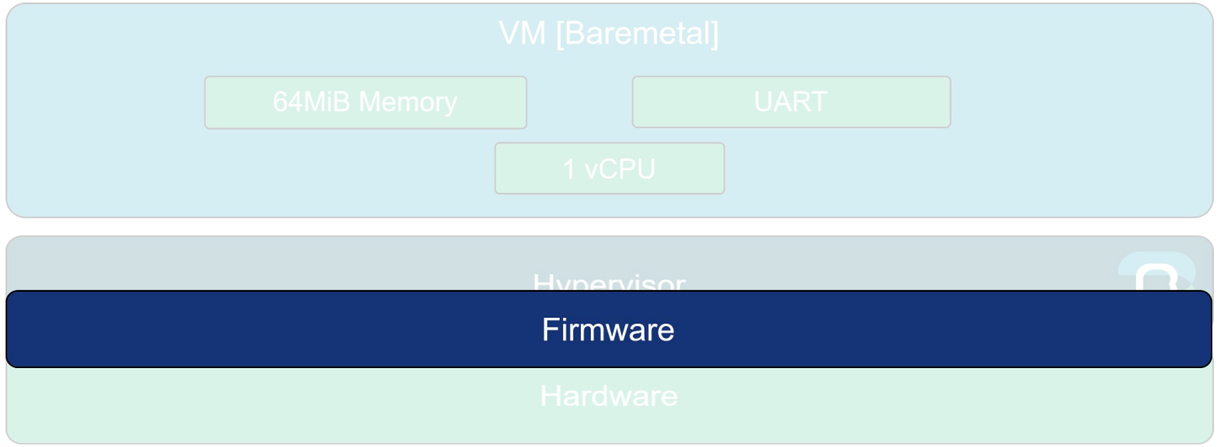
```
cp $BAO_SRCS/bin/qemu-riscv64-virt/baremetal/bao.bin \
    $BUILD_BAO_DIR/bao.bin
```



03 How to build firmware?

Clone OpenSBI

```
export OPENSBI_DIR=$TOOLS_DIR/OpenSBI
git clone https://github.com/bao-project/opensbi.git \
    $OPENSBI_DIR --depth 1 --branch bao/demo
```



03 How to build firmware?

Clone OpenSBI

```
export OPENSBI_DIR=$TOOLS_DIR/OpenSBI
git clone https://github.com/bao-project/opensbi.git \
OPENSBI_DIR --depth 1 --branch bao/demo
```

Compile OpenSBI

```
make -C $TOOLS_DIR/OpenSBI PLATFORM=generic \
    FW_PAYLOAD=y \
    FW_PAYLOAD_FDT_ADDR=0x80100000 \
    FW_PAYLOAD_PATH=$BUILD_BAO_DIR/bao.bin
```

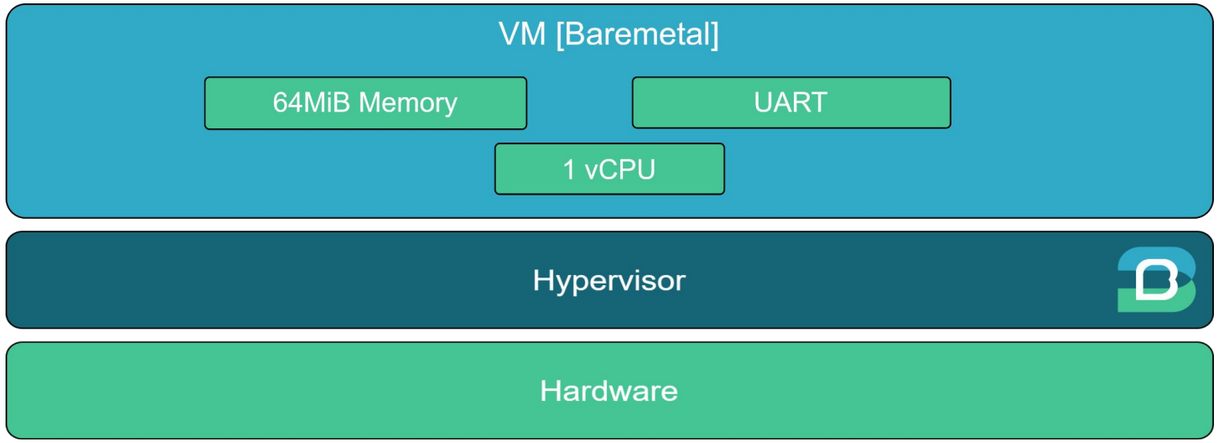
Copy the compiled image

```
cp $TOOLS_DIR/OpenSBI/build/platform/generic/firmware/fw_payload.elf \
    $TOOLS_DIR/bin/opensbi.elf
```

04 How to run the demo?

Launch QEMU

```
qemu-system-riscv64 -nographic \
    -M virt -cpu rv64 -m 4G -smp 4 \
    -bios $TOOLS_DIR/bin/opensbi.elf \
    -device virtio-net-device,netdev=net0 \
    -netdev user,id=net0,net=192.168.42.0/24,\
     hostfwd=tcp:127.0.0.1:5555-:22 \
    -device virtio-serial-device -chardev pty,id=serial3 \
    -device virtconsole,chardev=serial3
```



Short Q&A

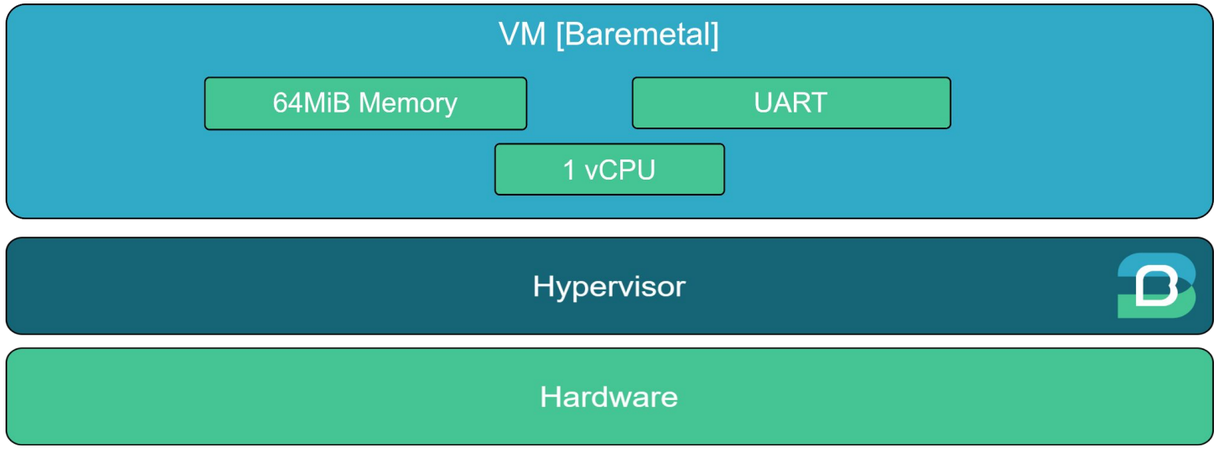
Fine-tune VM configuration

1x Baremetal with 4x vCPUs

01 What are we building?

Single-guest configuration

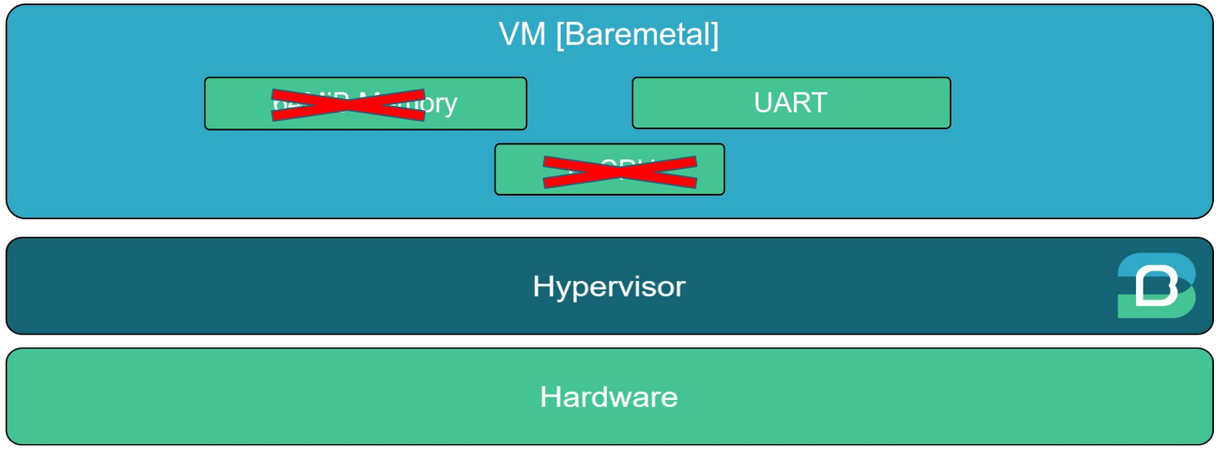
- Baremetal application
- 1 vCPUs
- 64 MiB Memory
- UART



01 What are we building?

Single-guest configuration

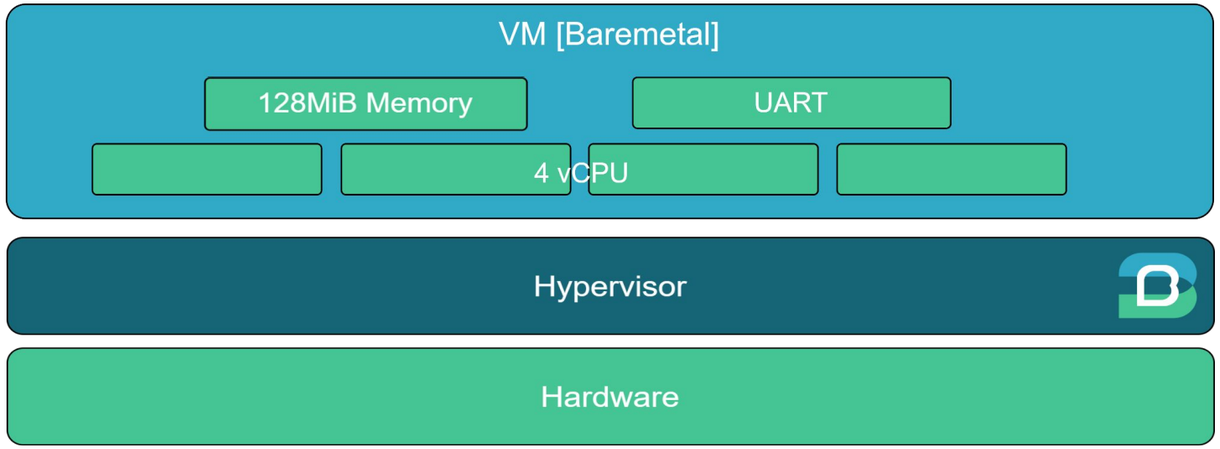
- Baremetal application
- ~~1 vCPU~~
- ~~64 MiB Memory~~
- UART



01 What are we building?

Single-guest configuration

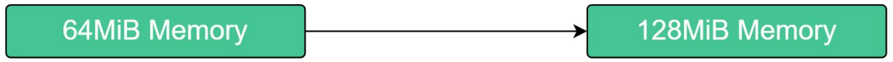
- Baremetal application
- 4 vCPUs
- 128 MiB Memory
- UART

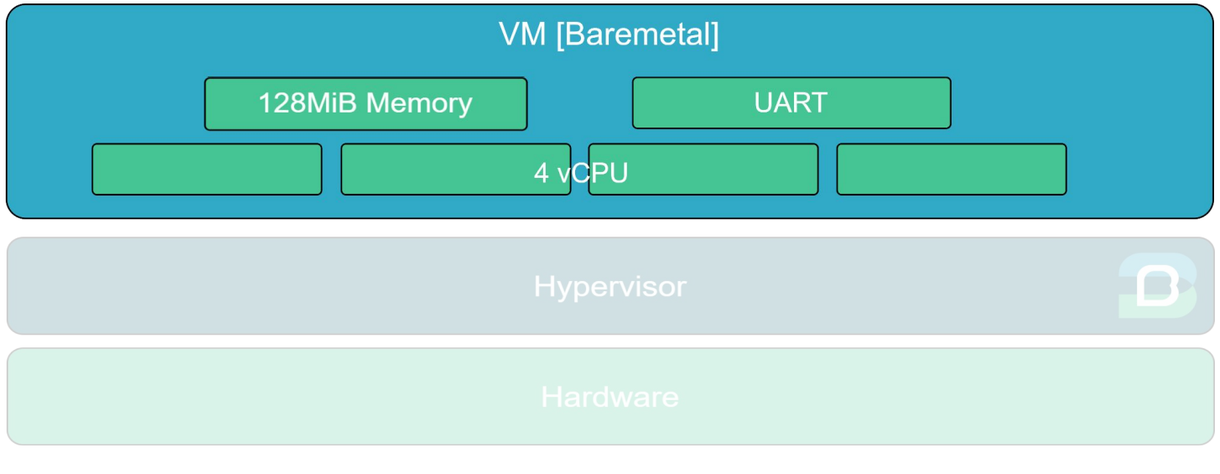


02 What to modify? Guest

baremetal/src/platform/qemu-riscv64-virt/inc/plat.h

```
#define PLAT_MEM_BASE 0x80200000
-#define PLAT_MEM_SIZE 0x4000000
+#define PLAT_MEM_SIZE 0x8000000
```

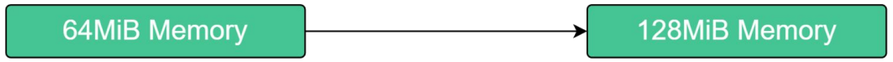


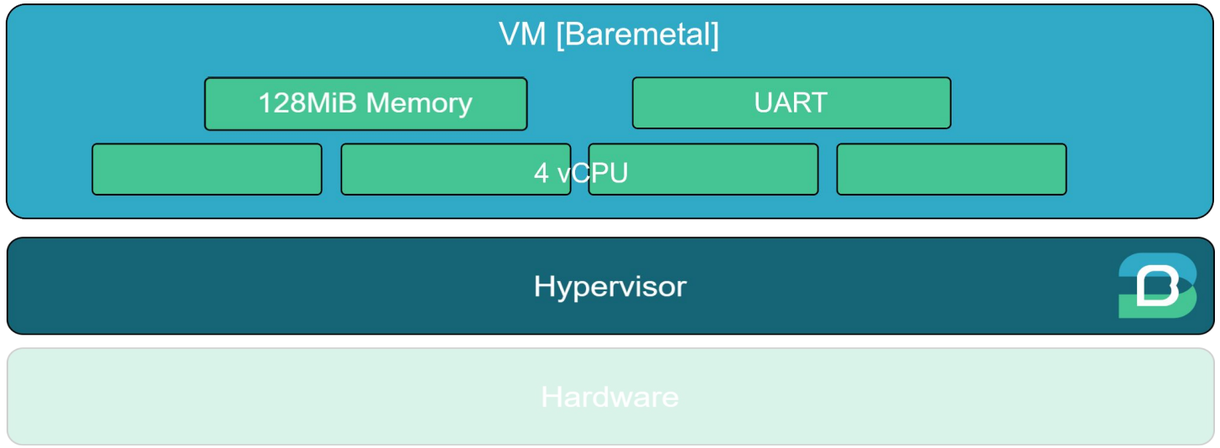


03 What to modify? Bao config

configs/baremetal.c

```
            .regions =  (struct vm_mem_region[]) {
                {
                    .base = 0x50000000,
-                   .size = 0x4000000
+                   .size = 0x8000000
                }
            }
```

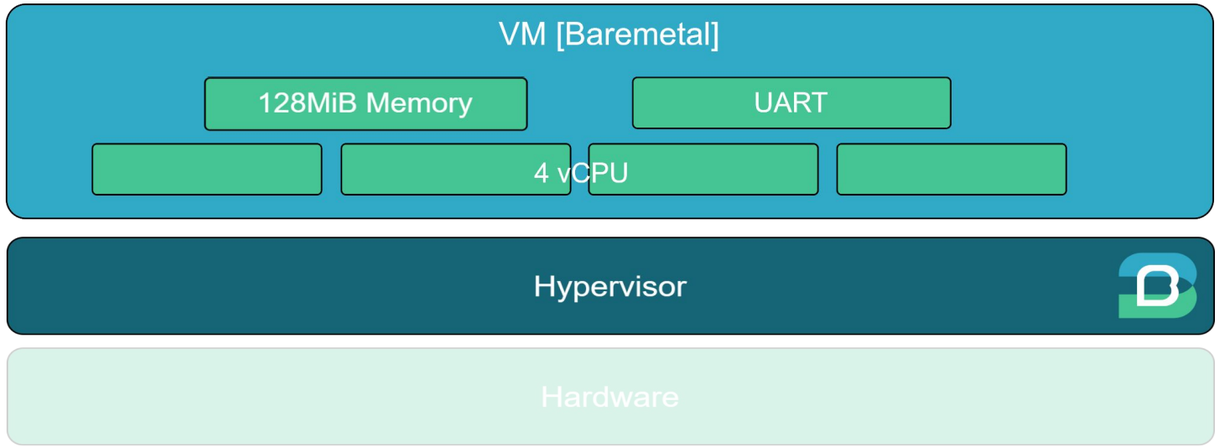




03 What to modify? Bao config

configs/baremetal.c

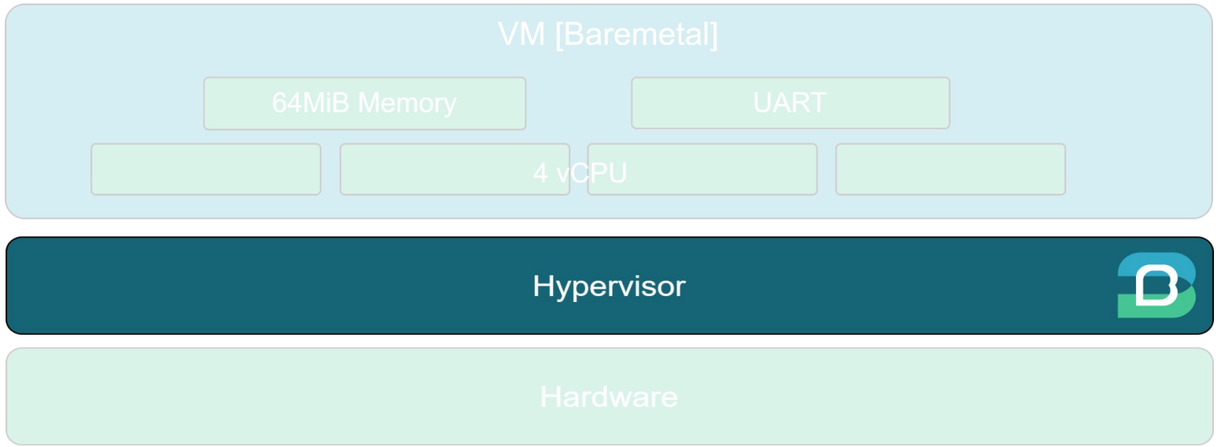
```
                .platform = {
-                       .cpu_num = 1
+                       .cpu_num = 4
```



03 How to build Bao?

Compile the hypervisor:

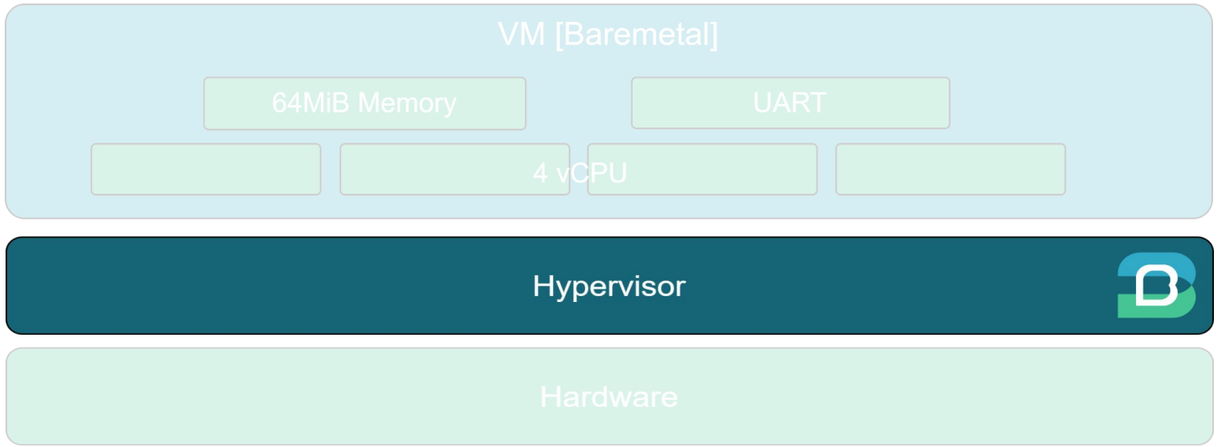
```
make -C $BAO_SRCS \
    PLATFORM=qemu-riscv64-virt \
    CONFIG_REPO=$ROOT_DIR/configs \
    CONFIG=baremetal_mod \
    CPPFLAGS=-DBAO_WRKDIR_IMGS=$BUILD_GUESTS_DIR
```



03 How to build Bao?

Move the binary file (bao.bin):

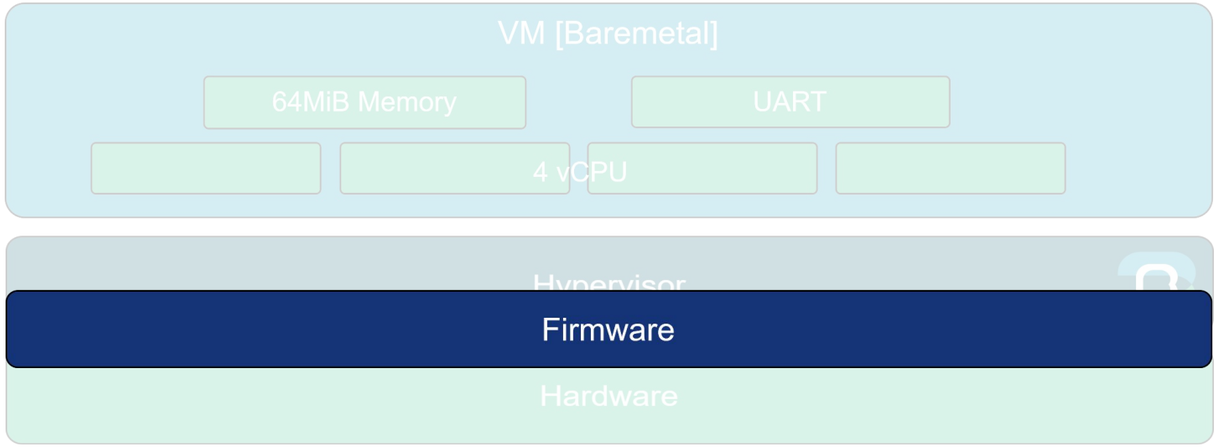
```
cp $BAO_SRCS/bin/qemu-riscv64-virt/baremetal_mod/bao.bin \
    $BUILD_BAO_DIR/bao.bin
```



04 How to run the demo?

Compile OpenSBI

```
make -C $TOOLS_DIR/OpenSBI PLATFORM=generic \
    FW_PAYLOAD=y \
    FW_PAYLOAD_FDT_ADDR=0x80100000 \
    FW_PAYLOAD_PATH=$BUILD_BAO_DIR/bao.bin
```



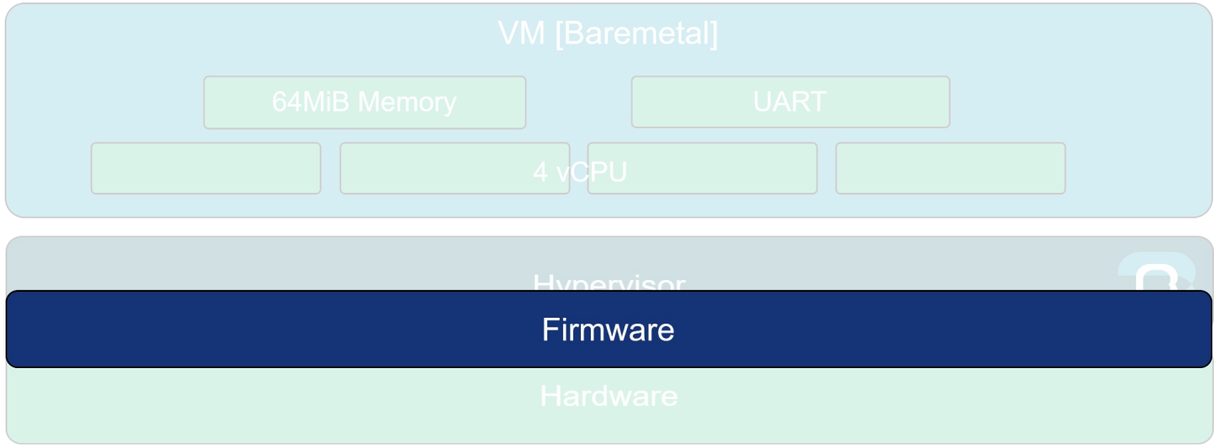
04 How to run the demo?

Compile OpenSBI

```
make -C $TOOLS_DIR/opensbi PLATFORM=generic \
    FW_PAYLOAD=y \
    FW_PAYLOAD_FDT_ADDR=0x80100000 \
    FW_PAYLOAD_PATH=$BUILD_BAO_DIR/bao.bin
```

Copy the compiled image

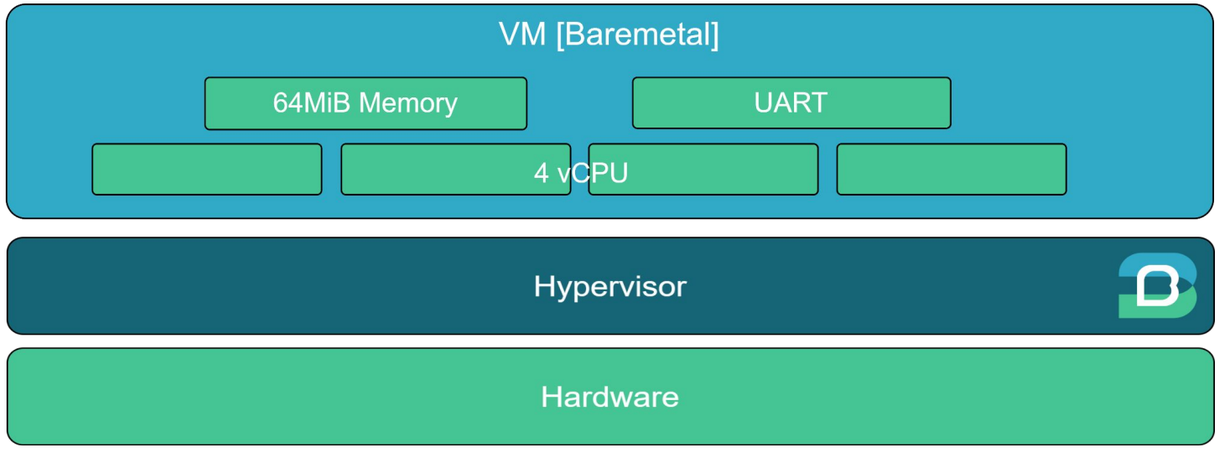
```
cp $TOOLS_DIR/OpenSBI/build/platform/generic/firmware/fw_payload.elf \
    $TOOLS_DIR/bin/opensbi.elf
```



04 How to run the demo?

Launch QEMU

```
qemu-system-riscv64 -nographic \
    -M virt -cpu rv64 -m 4G -smp 4 \
    -bios $TOOLS_DIR/bin/opensbi.elf \
    -device virtio-net-device,netdev=net0 \
    -netdev user,id=net0,net=192.168.42.0/24,\
     hostfwd=tcp:127.0.0.1:5555-:22 \
    -device virtio-serial-device -chardev pty,id=serial3 \
    -device virtconsole,chardev=serial3
```



Short Q&A

Dual-guest Bao setup with FreeRTOS

1x Baremetal with 3x vCPUs, 1x FreeRTOS with 1x vCPU

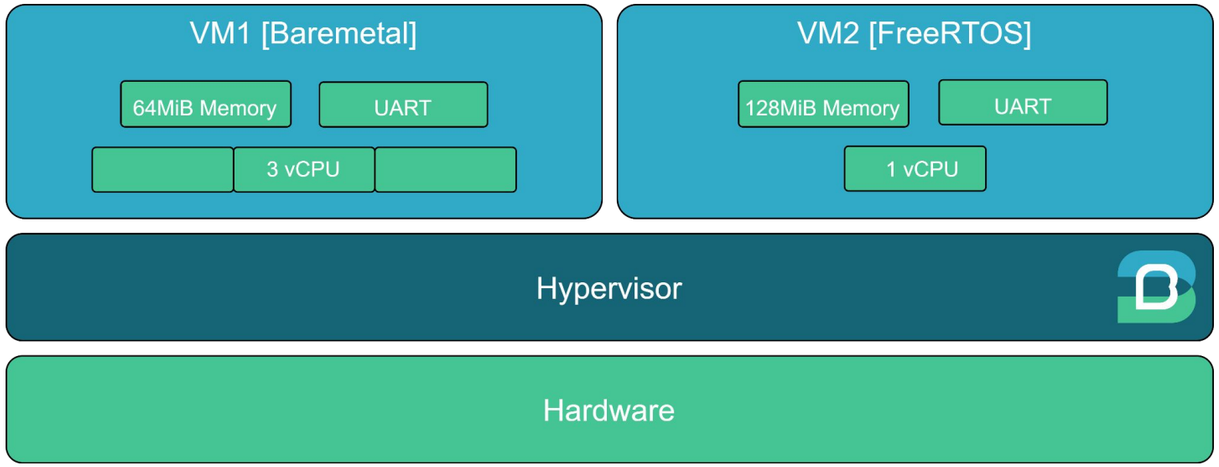
01 What are we building?

VM1

- Baremetal
- 3 vCPUs
- 64 MiB Memory
- UART

VM2

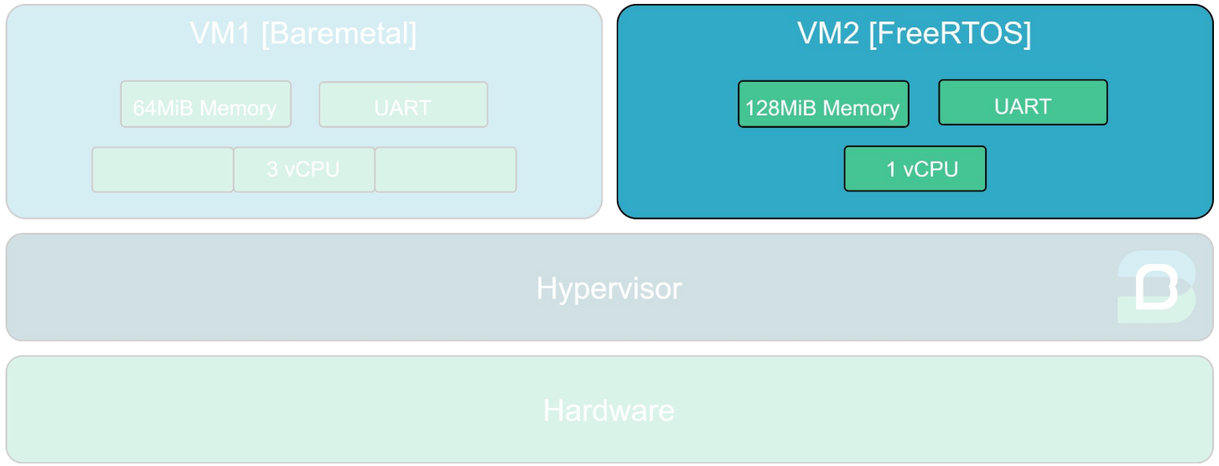
- FreeRTOS
- 1 vCPUs
- 128 MiB Memory
- UART



02 How to setup the guest?

configs/baremetal-freeRTOS.c

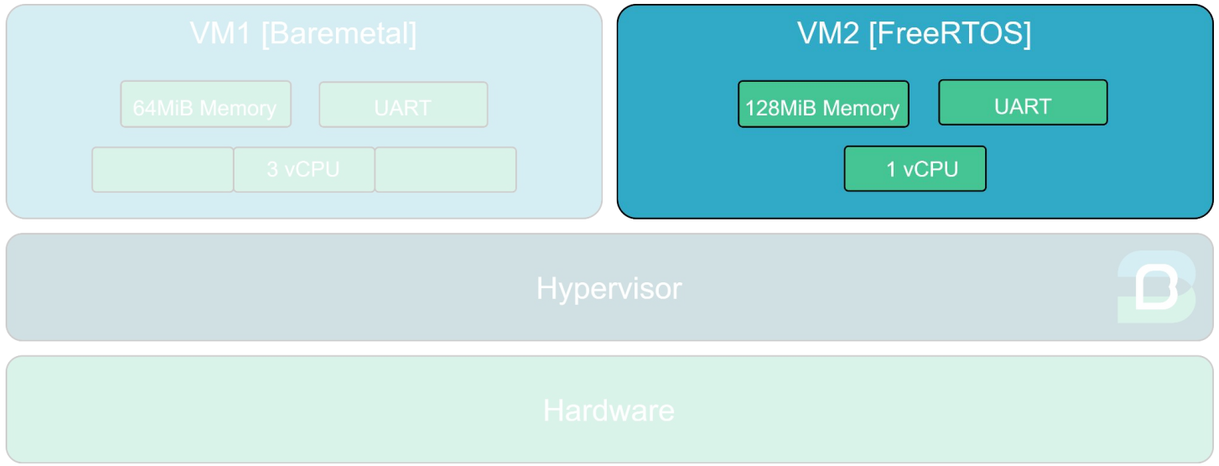
```
#include <config.h>

VM_IMAGE(baremetal_image,
XSTR(BAO_WRKDIR_IMGS/baremetal-freeRTOS-setup/baremetal.bin));
VM_IMAGE(freertos_image,
XSTR(BAO_WRKDIR_IMGS/baremetal-freeRTOS-setup/freertos.bin));

struct config config = {
    .vmlist_size = 2,
    .vmlist = {
        { //baremetal guest configuration},
        {
            .image = VM_IMAGE_BUILTIN(freertos_image, 0x00000000),
(...)
```



02 How to setup the guest?

configs/baremetal-freeRTOS.c

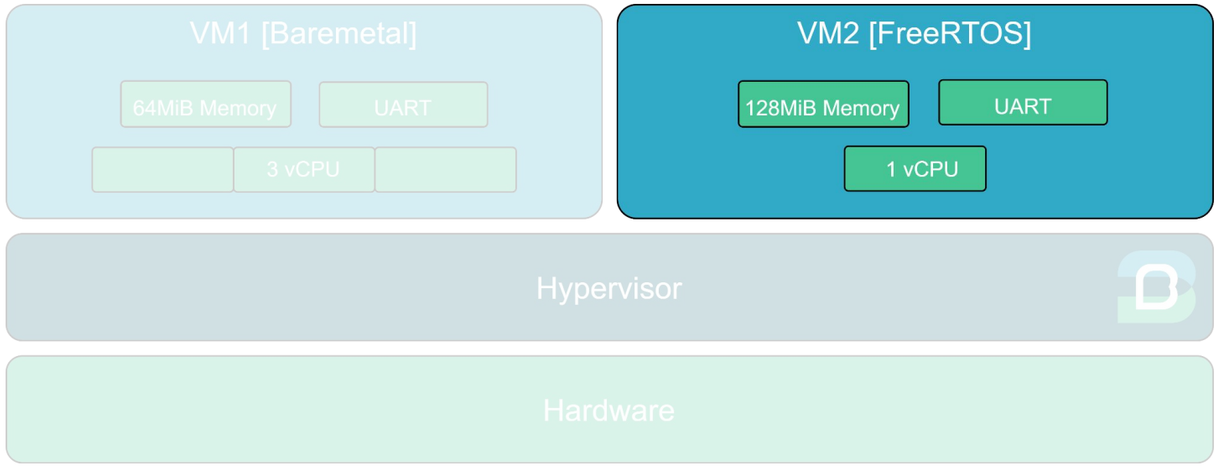
```
#include <config.h>

VM_IMAGE(baremetal_image,
XSTR(BAO_WRKDIR_IMGS/baremetal-freeRTOS-setup/baremetal.bin));
VM_IMAGE(freertos_image,
XSTR(BAO_WRKDIR_IMGS/baremetal-freeRTOS-setup/freertos.bin));

struct config config = {
    .vmlist_size = 2,
    .vmlist = {
        { //baremetal guest configuration},
        {
            .image = VM_IMAGE_BUILTIN(freertos_image, 0x00000000),
(...)
```



02 How to setup the guest?

configs/baremetal-freeRTOS.c

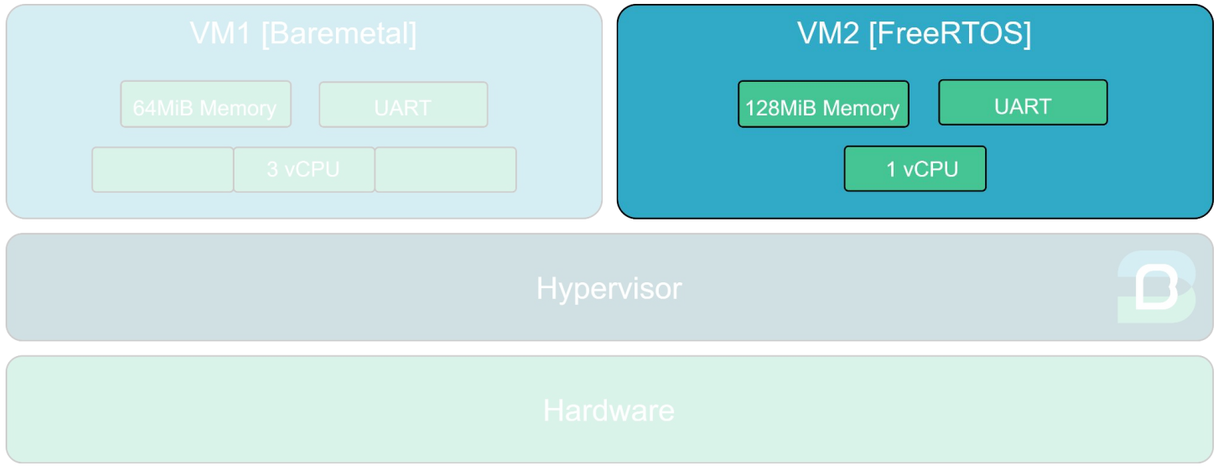
```
struct config config = {
    .vmlist_size = 2,
     .vmlist = {
         {
            // baremetal guest configuration
-           .cpu_num = 4,
+           .cpu_num = 3,
         },
         {
            // freeRTOS guest configuration
+           .cpu_num = 1,
         },
```



02 How to build the guest?

Define an environment variable for the FreeRTOS guest application

```
export FREERTOS_SRCS=$ROOT_DIR/freertos
```



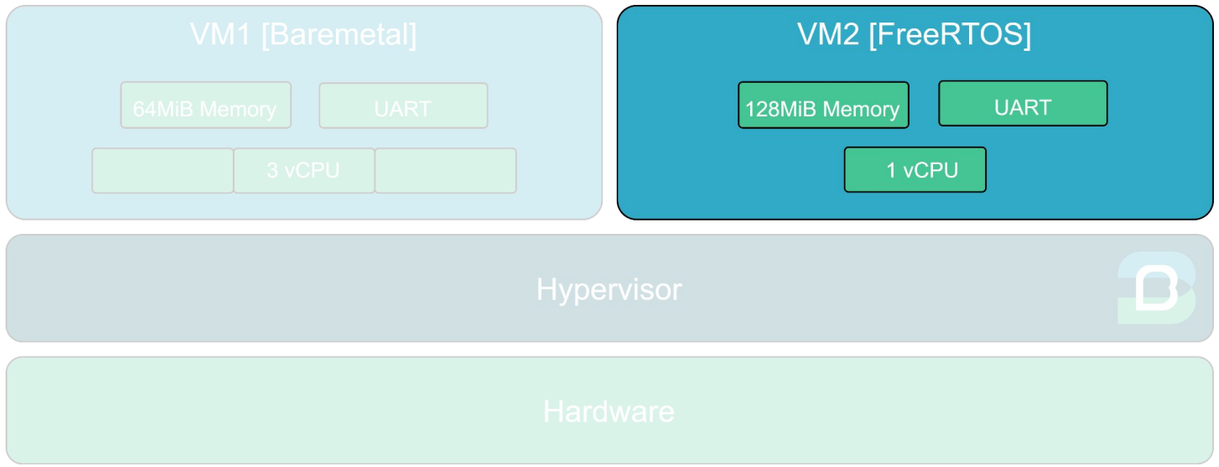
02 How to build the guest?

Define an environment variable for the FreeRTOS guest application:

```
export FREERTOS_SRCS=$ROOT_DIR/freertos
```

Clone the FreeRTOS guest

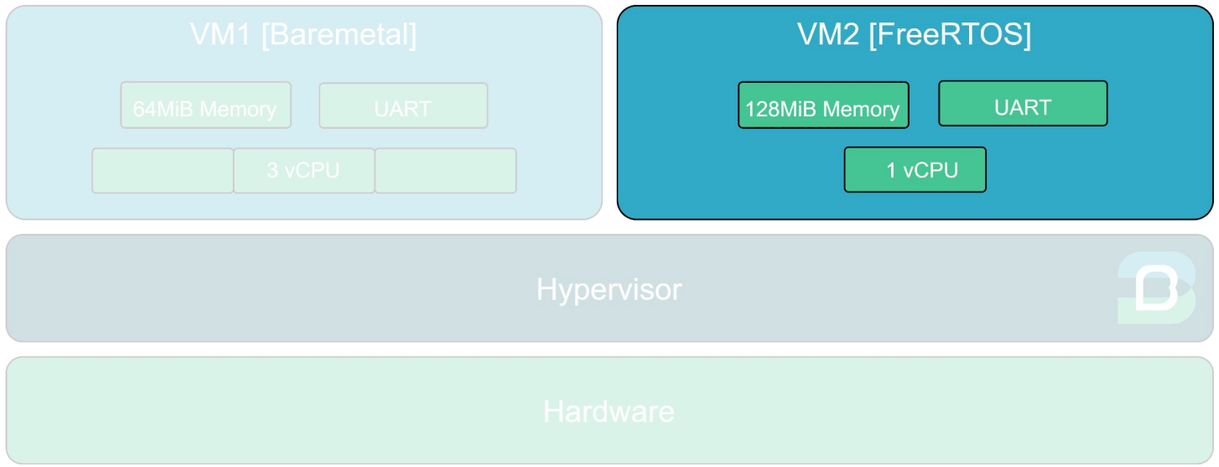
```
git clone --recursive --shallow-submodules \
    https://github.com/bao-project/freertos-over-bao.git \
    $FREERTOS_SRCS --branch demo
```



02 How to build the guest?

Compile the FreeRTOS Guest

```
export FREERTOS_PARAMS="STD_ADDR_SPACE=y"
git -C $FREERTOS_SRCS apply $PATCHES_DIR/freeRTOS.patch
make -C $FREERTOS_SRCS PLATFORM=qemu-riscv64-virt \
    $FREERTOS_PARAMS
```



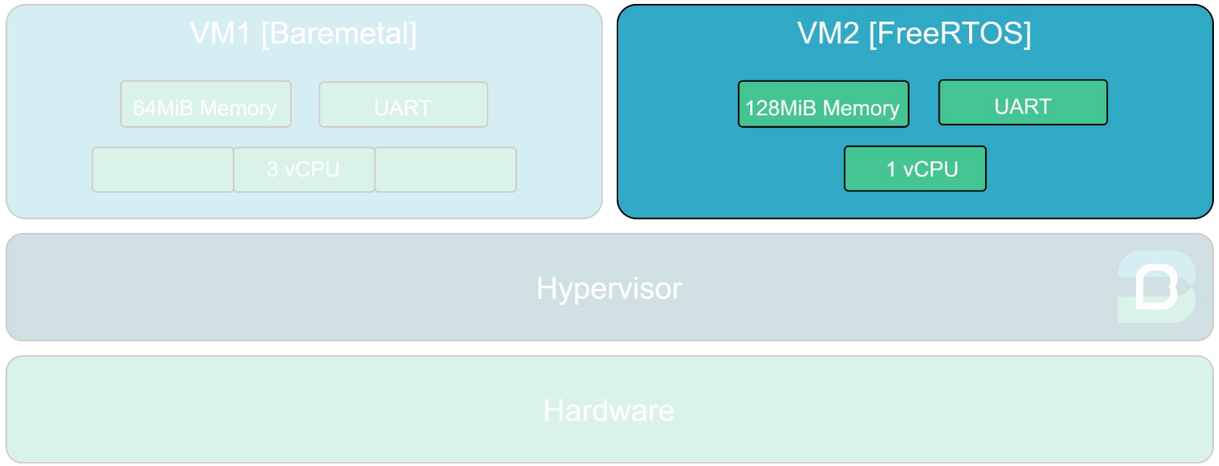
02 How to build the guest?

Compile the FreeRTOS Guest

```
export FREERTOS_PARAMS="STD_ADDR_SPACE=y"
git -C $FREERTOS_SRCS apply $PATCHES_DIR/freeRTOS.patch
make -C $FREERTOS_SRCS PLATFORM=qemu-riscv64-virt \
$FREERTOS_PARAMS
```

Move the binary file (free-rtos.bin)

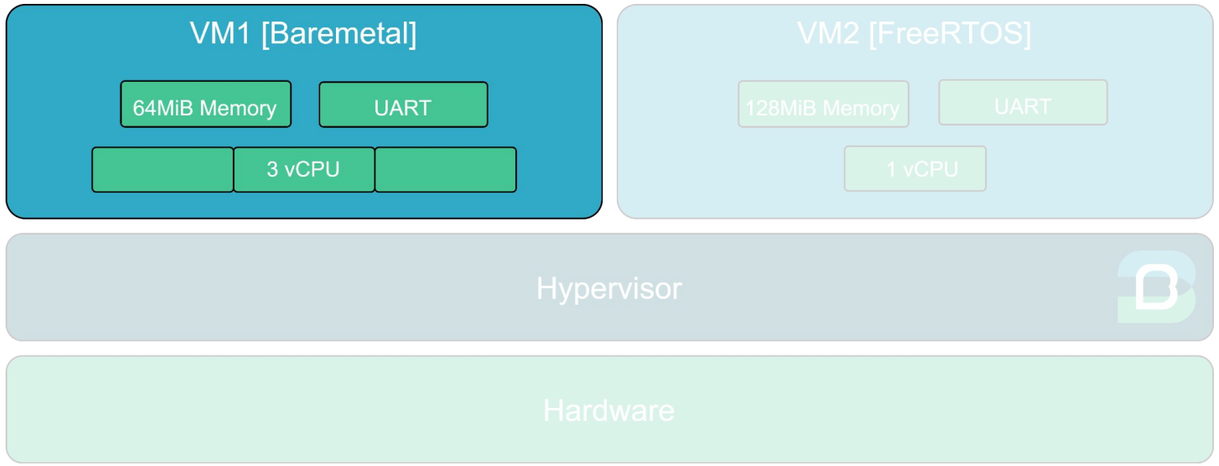
```
mkdir -p $BUILD_GUESTS_DIR/baremetal-freeRTOS-setup
cp $FREERTOS_SRCS/build/qemu-riscv64-virt/freertos.bin \
    $BUILD_GUESTS_DIR/baremetal-freeRTOS-setup/freertos.bin
```



02 How to build the guest?

Copy compiled baremetal image

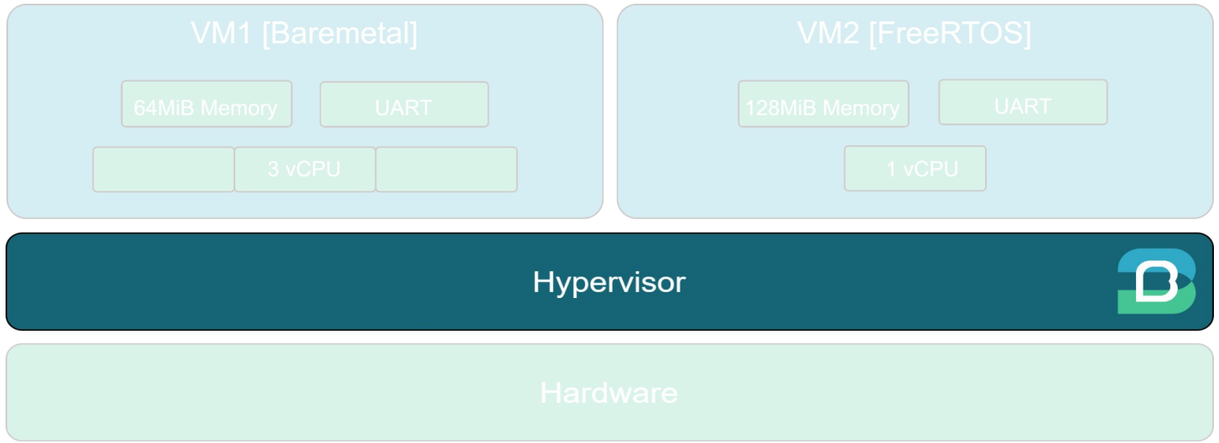
```
cp $BUILD_GUESTS_DIR/baremetal-setup/baremetal.bin \
    $BUILD_GUESTS_DIR/baremetal-freeRTOS-setup/baremetal.bin
```



03 How to build Bao?

Compile the hypervisor

```
make -C $BAO_SRCS \
    PLATFORM=qemu-riscv64-virt \
    CONFIG_REPO=$ROOT_DIR/configs \
    CONFIG=baremetal-freeRTOS \
    CPPFLAGS=-DBAO_WRKDIR_IMGS=$BUILD_GUESTS_DIR
```

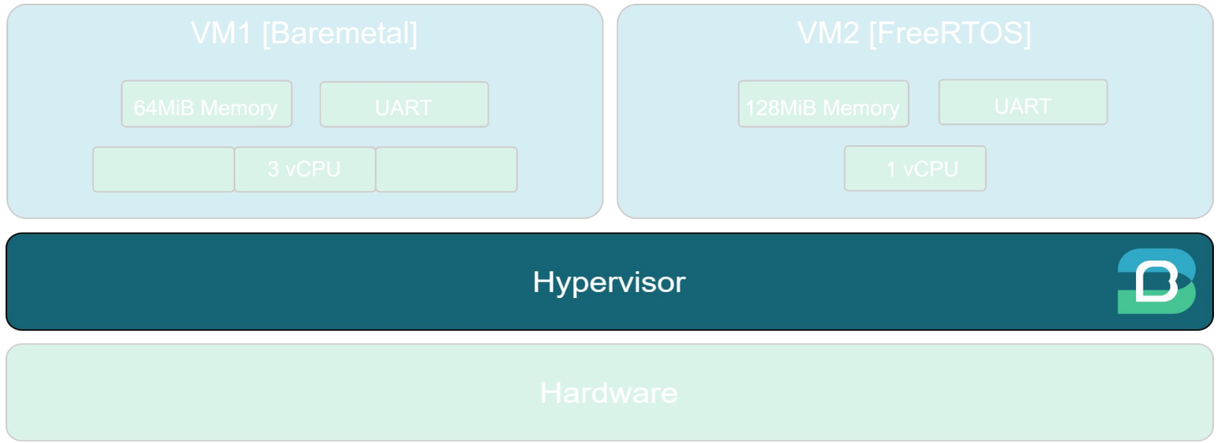


03 How to build Bao?

Compile the hypervisor

```
make -C $BAO_SRCS \
    PLATFORM=qemu-riscv64-virt \
    CONFIG_REPO=$ROOT_DIR/configs \
    CONFIG=baremetal-freeRTOS \
    CPPFLAGS=-DBAO_WRKDIR_IMGS=$BUILD_GUESTS_DIR
```

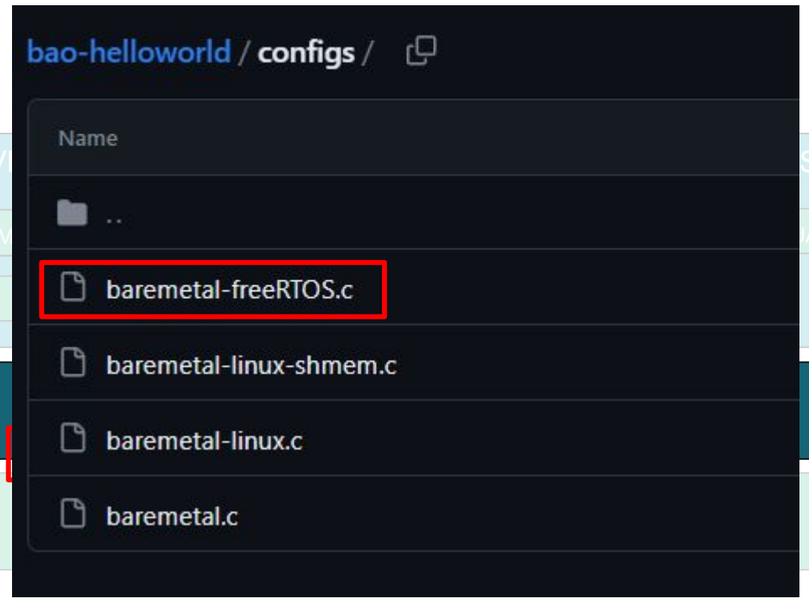
bao-helloworld / configs /



03 How to build Bao?

Compile the hypervisor

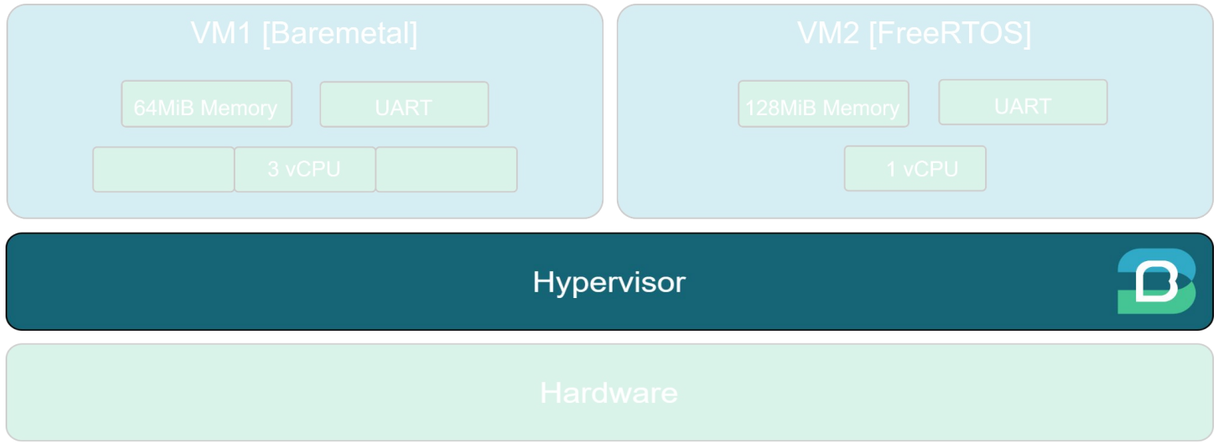
```
make -C $BAO_SRCS \
    PLATFORM=qemu-riscv64-virt \
    CONFIG_REPO=$ROOT_DIR/configs \
    CONFIG=baremetal-freeRTOS \
    CPPFLAGS=-DBAO_WRKDIR_IMGS=$BUILD_GUESTS_DIR
```



03 How to build Bao?

Move the binary file (bao.bin)

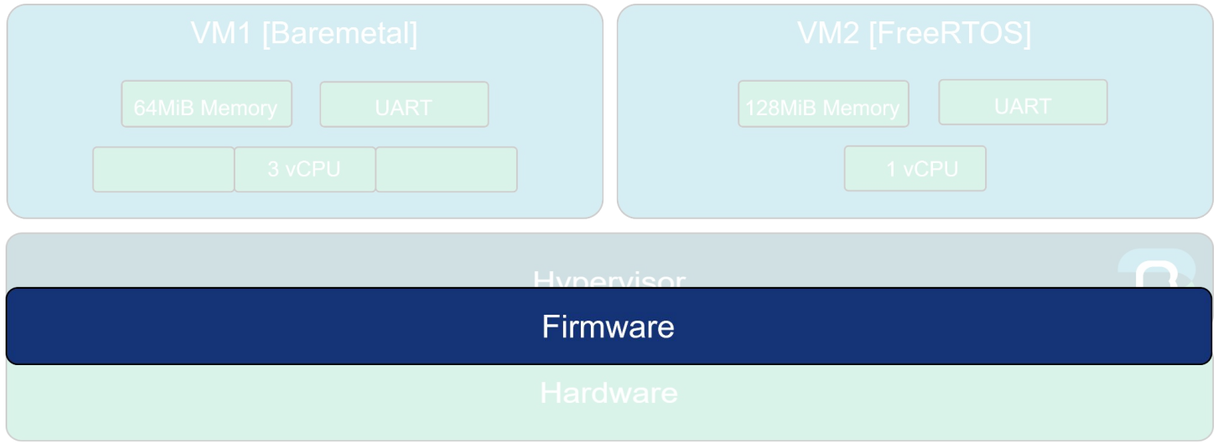
```
cp $BAO_SRCS/bin/qemu-riscv64-virt/baremetal-freeRTOS/bao.bin \
    $BUILD_BAO_DIR/bao.bin
```



04 How to build firmware?

Compile OpenSBI

```
make -C $TOOLS_DIR/OpenSBI PLATFORM=generic \
    FW_PAYLOAD=y \
    FW_PAYLOAD_FDT_ADDR=0x80100000 \
    FW_PAYLOAD_PATH=$BUILD_BAO_DIR/bao.bin
```



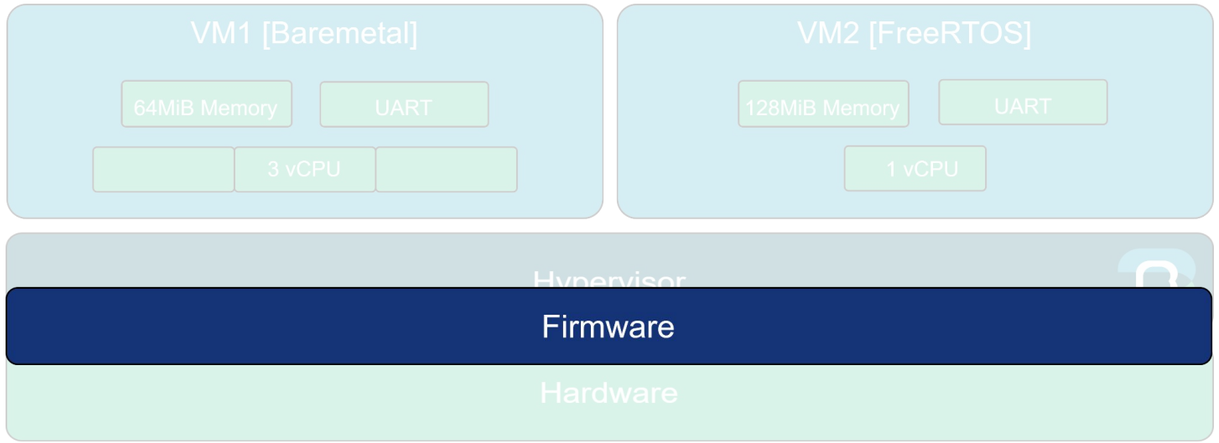
04 How to build firmware?

Compile OpenSBI

```
make -C $TOOLS_DIR/OpenSBI PLATFORM=generic \
    FW_PAYLOAD=y \
    FW_PAYLOAD_FDT_ADDR=0x80100000 \
    FW_PAYLOAD_PATH=$BUILD_BAO_DIR/bao.bin
```

Copy the compiled image

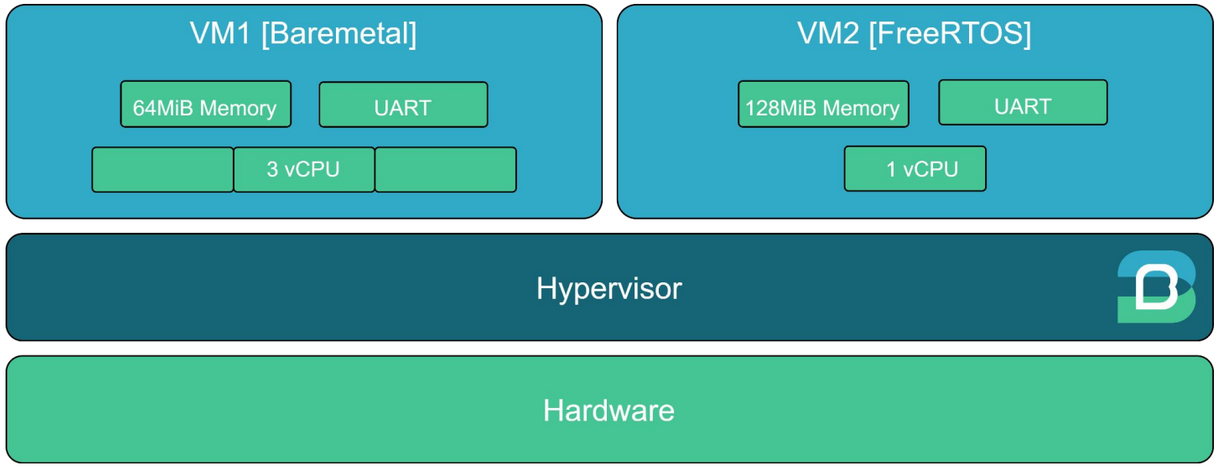
```
cp $TOOLS_DIR/OpenSBI/build/platform/generic/firmware/fw_payload.elf \
$TOOLS_DIR/bin/opensbi.elf
```



04 How to run the demo?

Launch QEMU

```
qemu-system-riscv64 -nographic \
    -M virt -cpu rv64 -m 4G -smp 4 \
    -bios $TOOLS_DIR/bin/opensbi.elf \
    -device virtio-net-device,netdev=net0 \
    -netdev user,id=net0,net=192.168.42.0/24,\
     hostfwd=tcp:127.0.0.1:5555-:22 \
    -device virtio-serial-device -chardev pty,id=serial3 \
    -device virtconsole,chardev=serial3
```



Short Q&A

IT'S COFFEE BREAK TIME | 5 minutes

Dual-guest Bao setup with Linux

1x baremetal with 1x vCPUs; 1x Linux with 3 vCPUs

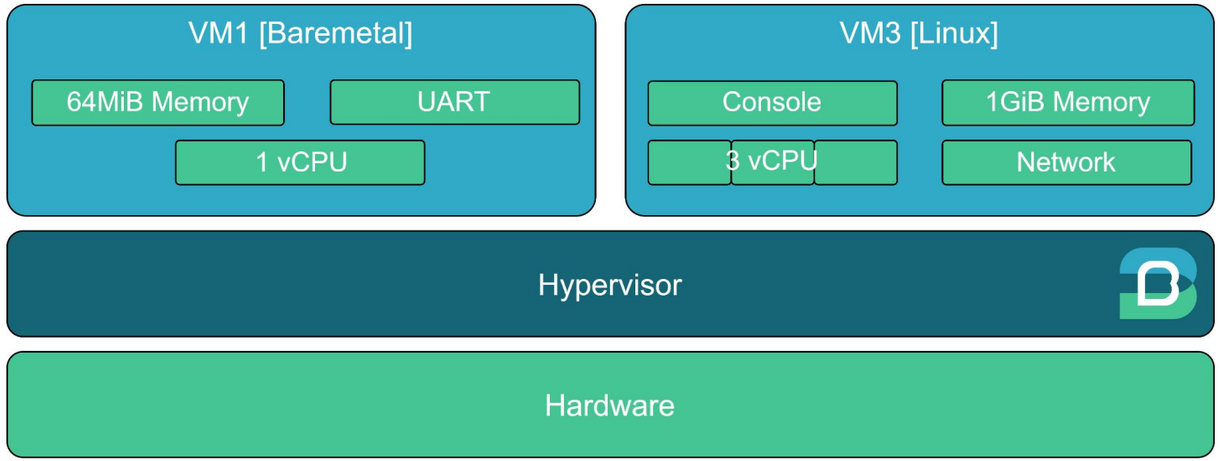
01 What are we building?

VM1

- Baremetal
- 1 vCPUs
- 64 MiB Memory
- UART

VM2

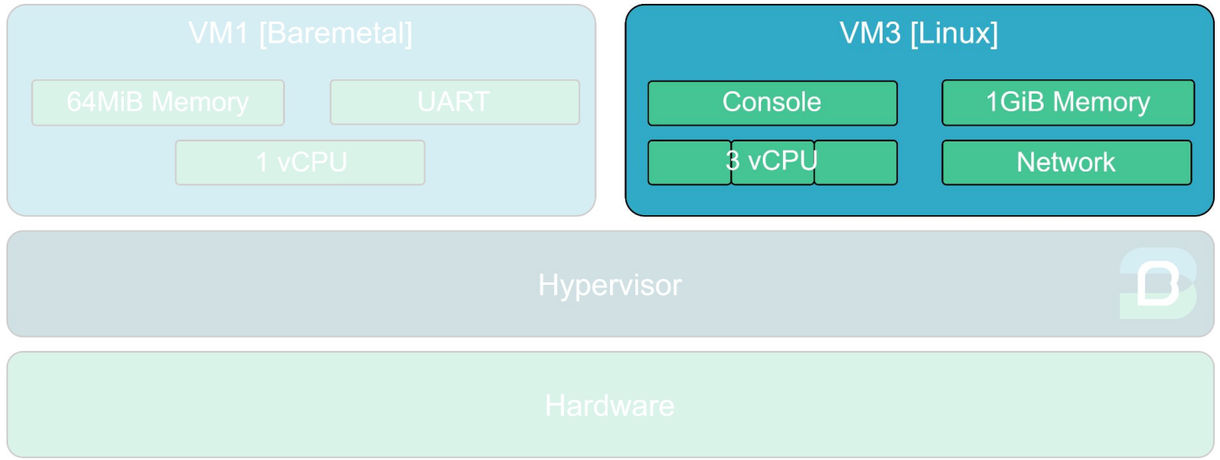
- Linux
- 3 vCPUs
- 1 GiB Memory
- Console
- Network



02 How to setup the guest?

configs/baremetal-linux.c

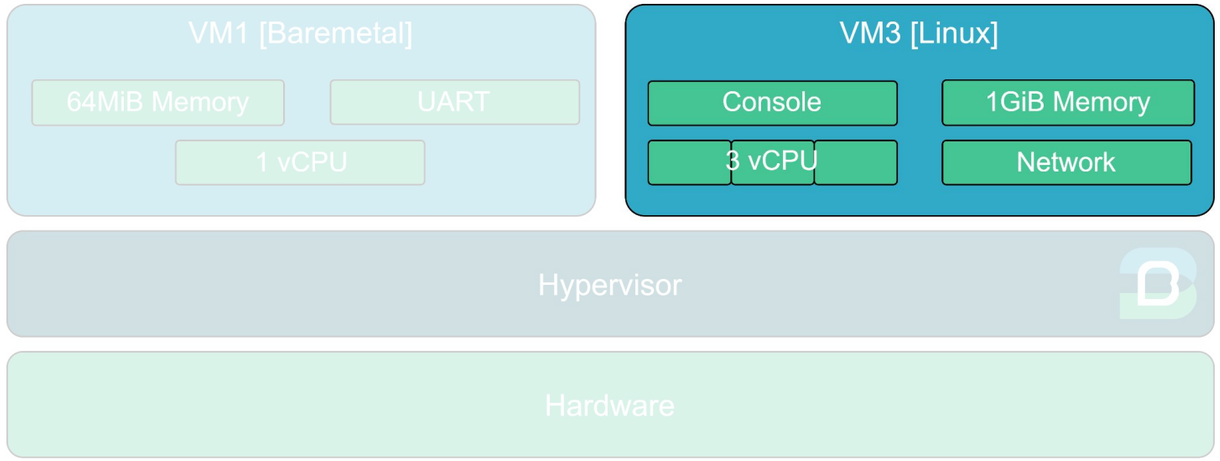
```
#include <config.h>

VM_IMAGE(baremetal_image,
XSTR(BAO_WRKDIR_IMGS/baremetal-linux-setup/baremetal.bin));
VM_IMAGE(linux_image, XSTR(BAO_WRKDIR_IMGS/baremetal-linux-setup/linux.bin));

struct config config = {
   .vmlist_size = 2,
    .vmlist = {
        {
     //baremetal guest configuration
        },
        {
                .image = VM_IMAGE_BUILTIN(linux_image,  0x90200000),
(…)
```



02 How to setup the guest?

configs/baremetal-linux.c

```
#include <config.h>

VM_IMAGE(baremetal_image, /path/to/baremetal.bin));
VM_IMAGE(linux_image, /path/to/linux.bin));


struct config config = {
   .vmlist_size = 2,
    .vmlist = {
        {
     //baremetal guest configuration
        },
        {
                .image = VM_IMAGE_BUILTIN(linux_image,  0x90200000),
(…)
```



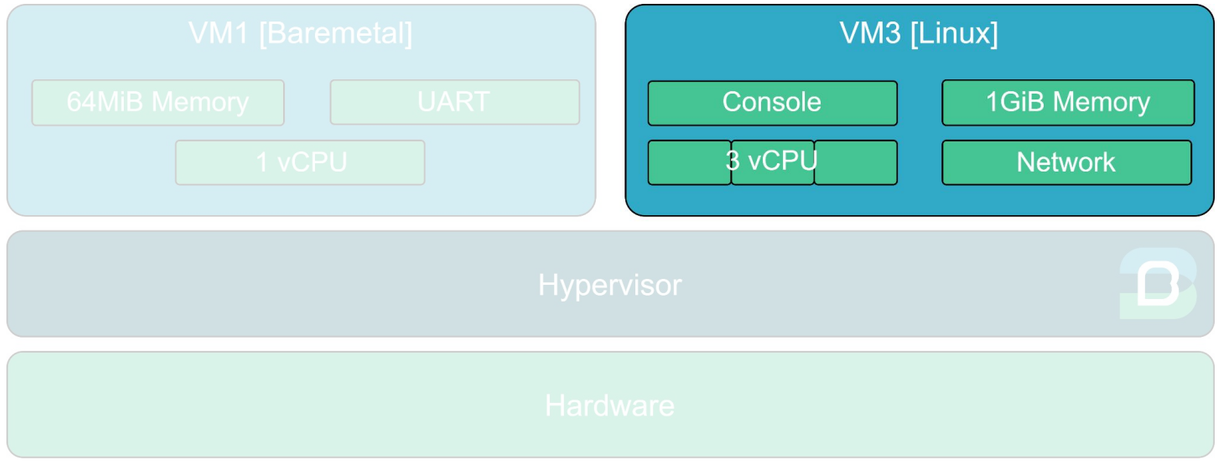
02 How to setup the guest?

configs/baremetal-linux.c

```
struct config config = {
    .vmlist_size = 2,
    .vmlist = {
        {
            // baremetal guest configuration
            …
-           .cpu_num = 4,
+           .cpu_num = 1,
            …
        },
+       {
            // Linux guest configuration
+           …
+           .cpu_num = 3,
+           …
+       },
```

srcs/devicetrees/qemu-riscv64-virt/**linux.dts**

```
…
        cpus {
                …
                cpu@0 {
                        …
                };
                cpu@1 {
                        …
                };
                cpu@2 {
                        …
                };
        };
…
```



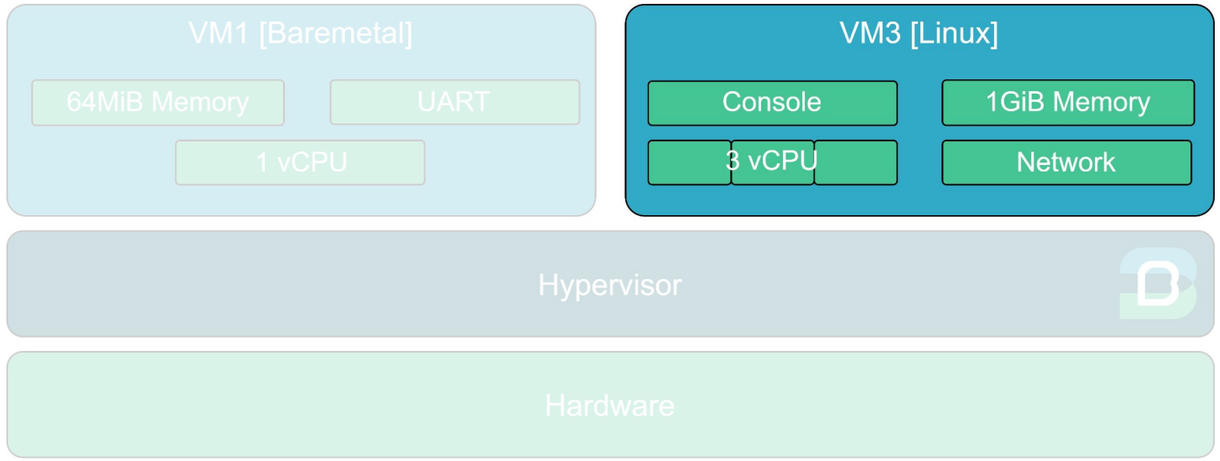
02 How to setup the guest?

configs/baremetal-linux.c

```
struct config config = {
   .vmlist_size = 2,
    .vmlist = {
        {// baremetal guest configuration },
        {
            // Linux guest config    uration
+           …
+           .region_num = 1,
+           .regions = (struct vm_mem_region[]) {
+                   {
+                           .base = 0x90000000,
+                           .size = 0x40000000,
+                           …
+                   }
+           },
+           …
        },
```

/srcs/devicetrees/qemu-riscv64-virt/**linux.dts**

```
(…)

memory@90000000 {
   device_type = "memory";
   reg = <0x0 0x90000000 0x0 0x40000000>;
};

(…)
```



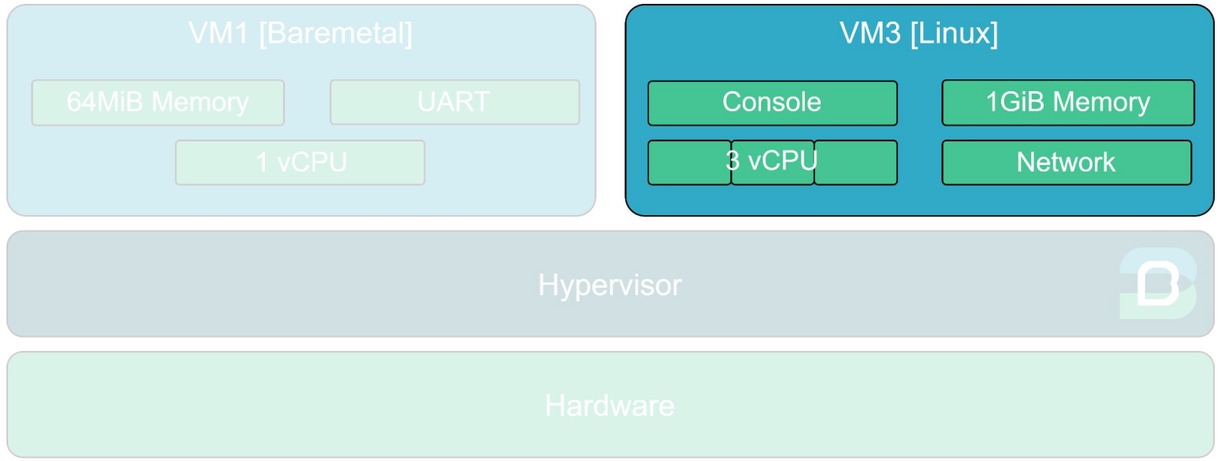
02 How to setup the guest?

configs/baremetal-linux.c

```
      {
        // Linux guest configuration
+           …
+.dev_num = 1,
+.devs =  (struct vm_dev_region[]) {
+    {
+       /* virtio devices */
+       .pa = 0x10001000,
+       .va = 0x10001000,
+       .size = 0x00008000,
+       .interrupt_num = 8,
+       .interrupts = (irqid_t[]) {1, 2,3,4,5,6,7,8}
+    },
+},
+          …
```

/srcs/devicetrees/qemu-riscv64-virt/**linux.dts**

```
(...)
virtio_mmio@10002000 {
    interrupts = <0x2>;
    interrupt-parent = <&plic>;
    reg = <0x0 0x10002000 0x0 0x1000>;
    compatible = "virtio,mmio";
};
virtio_mmio@10001000 {
    interrupts = <0x1>;
    interrupt-parent = <&plic>;
    reg = <0x0 0x10001000 0x0 0x1000>;
    compatible = "virtio,mmio";
};
(...)
```



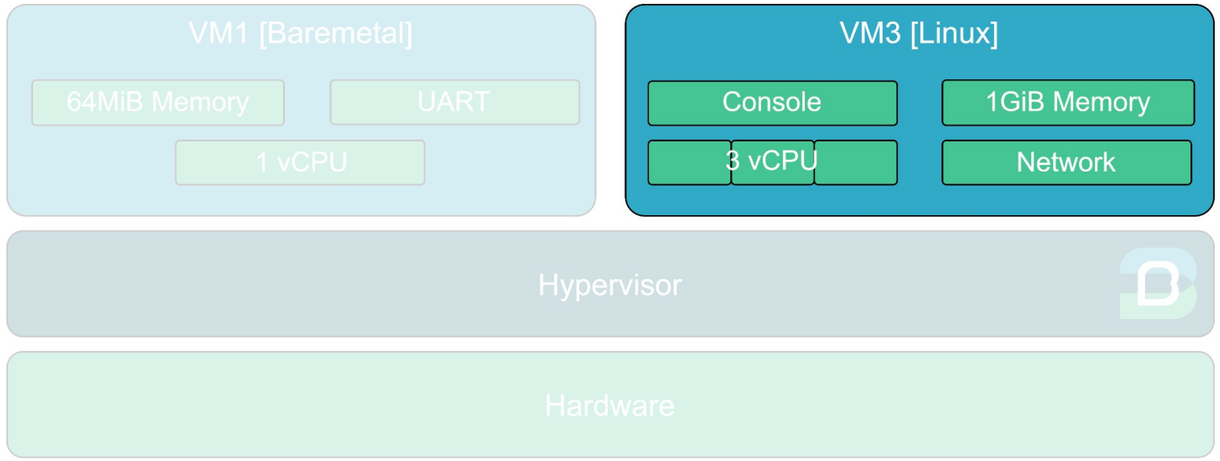
02 How to setup the guest?

configs/baremetal-linux.c

```
    {
        // Linux guest configuration
+           …
+           .arch = {
+               .plic_base = 0xc000000,
+           }
+           …
    }
```

/srcs/devicetrees/qemu-riscv64-virt/**linux.dts**

```
(…)
  plic: interrupt-controller@c000000 {
          riscv,ndev = <60>;
          reg = <0x0 0xc000000 0x0 0x4000000>;
          interrupts-extended = <
            &cpu0_intc 11 &cpu0_intc 9
            &cpu1_intc 11 &cpu1_intc 9
            &cpu2_intc 11 &cpu2_intc 9
          >;
          interrupt-controller;
          compatible = "riscv,plic0";
          #interrupt-cells = <0x1>;
  };
(…)
```



02 How to build the guest?

bao-project / **bao-helloworld** Public

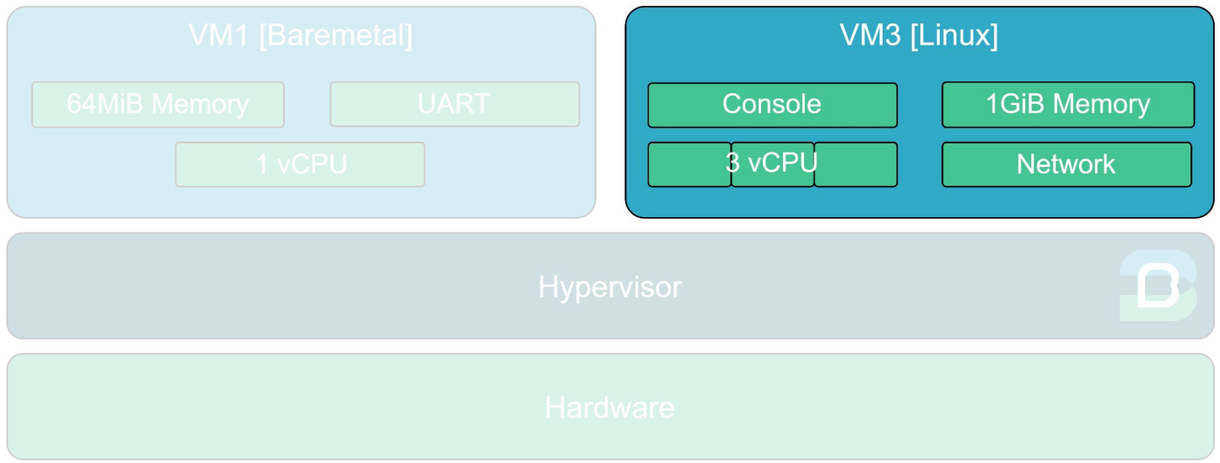
Name

- ..
- baremetal-freeRTOS-setup
- baremetal-linux-setup
- baremetal-linux-shmem-setup
- baremetal-setup

[Step-by-step Build Instructions]

Name

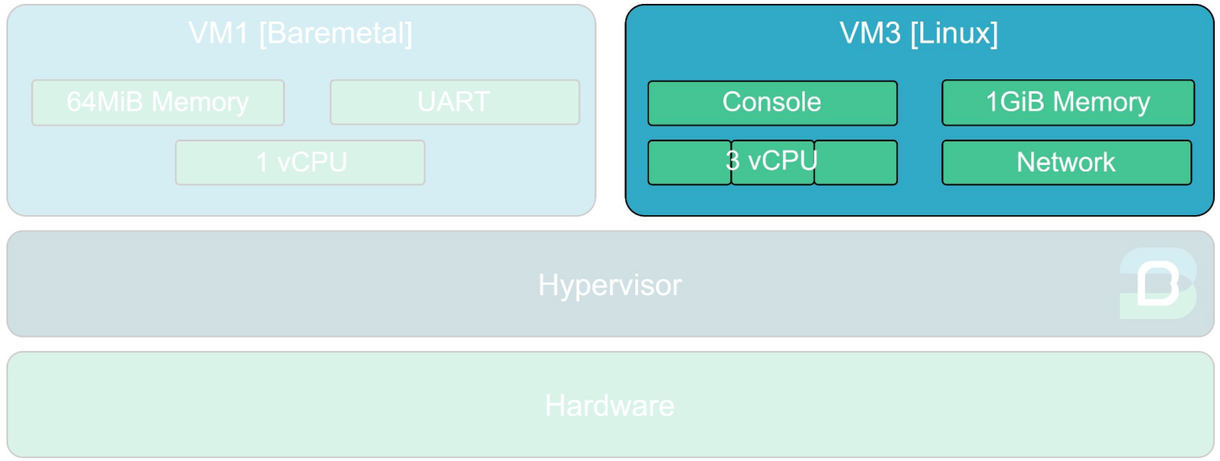
- ..
- baremetal.bin
- linux.bin



02 How to build the guest?

Copy pre-built Linux image

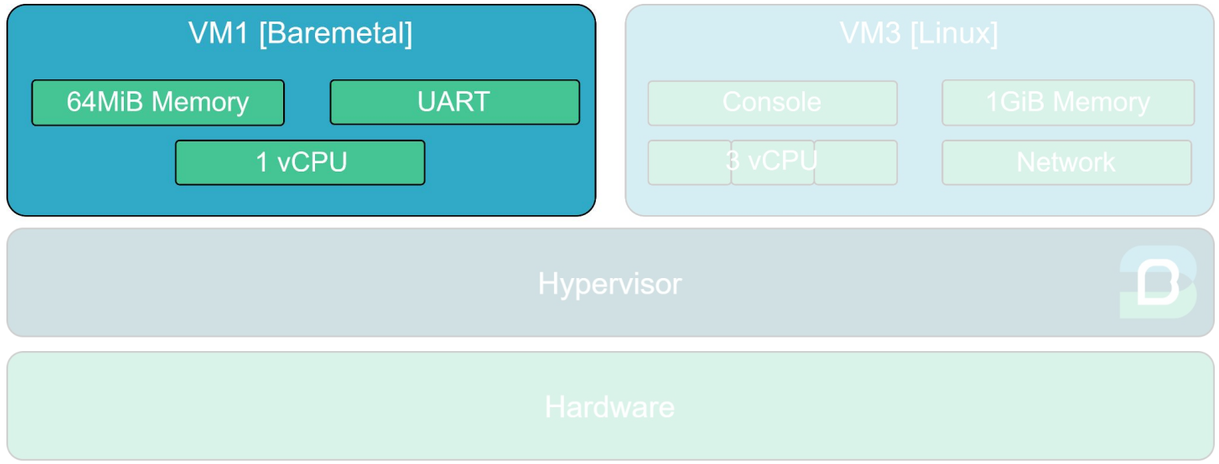
```
mkdir -p $BUILD_GUESTS_DIR/baremetal-linux-setup
cp $PRE_BUILT_IMGS/guests/baremetal-linux-setup/linux.bin \
    $BUILD_GUESTS_DIR/baremetal-linux-setup/linux.bin
```



02 How to build the guest?

Copy compiled baremetal image

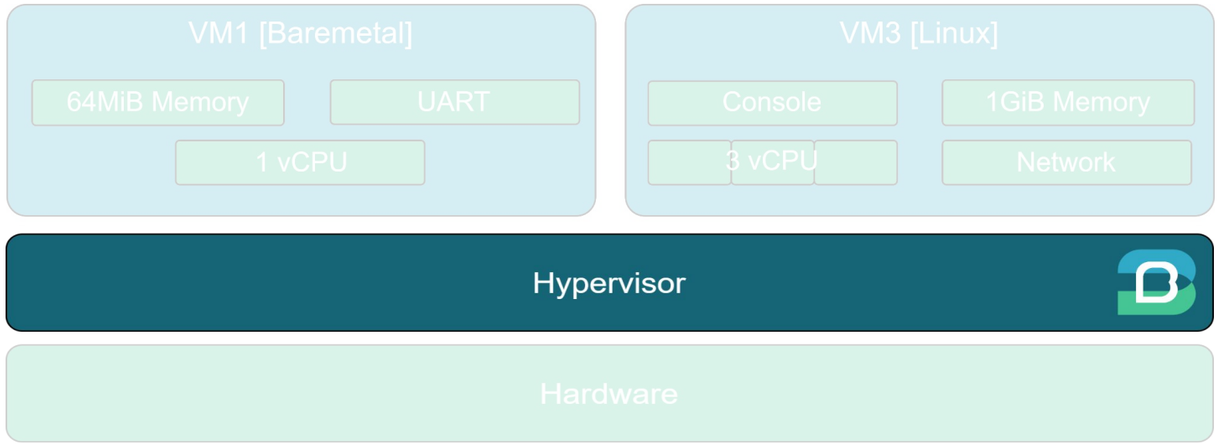
```
cp $BUILD_GUESTS_DIR/baremetal-setup/baremetal.bin \
    $BUILD_GUESTS_DIR/baremetal-linux-setup/baremetal.bin
```



03 How to build Bao?

Compile the hypervisor

```
make -C $BAO_SRCS \
    PLATFORM=qemu-riscv64-virt \
    CONFIG_REPO=$ROOT_DIR/configs \
    CONFIG=baremetal-linux \
    CPPFLAGS=-DBAO_WRKDIR_IMGS=$BUILD_GUESTS_DIR
```

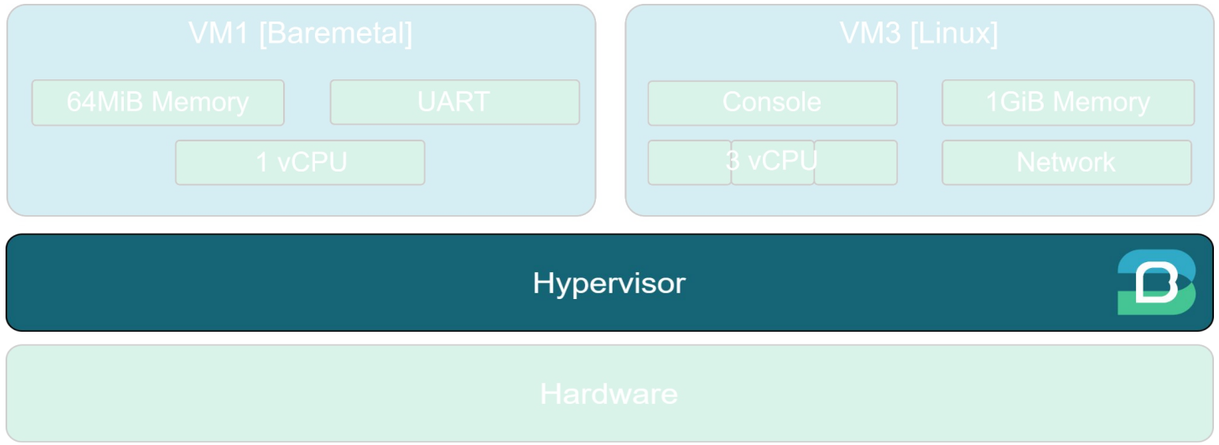


03 How to build Bao?

Compile the hypervisor

```
make -C $BAO_SRCS \
    PLATFORM=qemu-riscv64-virt \
    CONFIG_REPO=$ROOT_DIR/configs \
    CONFIG=baremetal-linux \
    CPPFLAGS=-DBAO_WRKDIR_IMGS=$BUILD_GUESTS_DIR
```

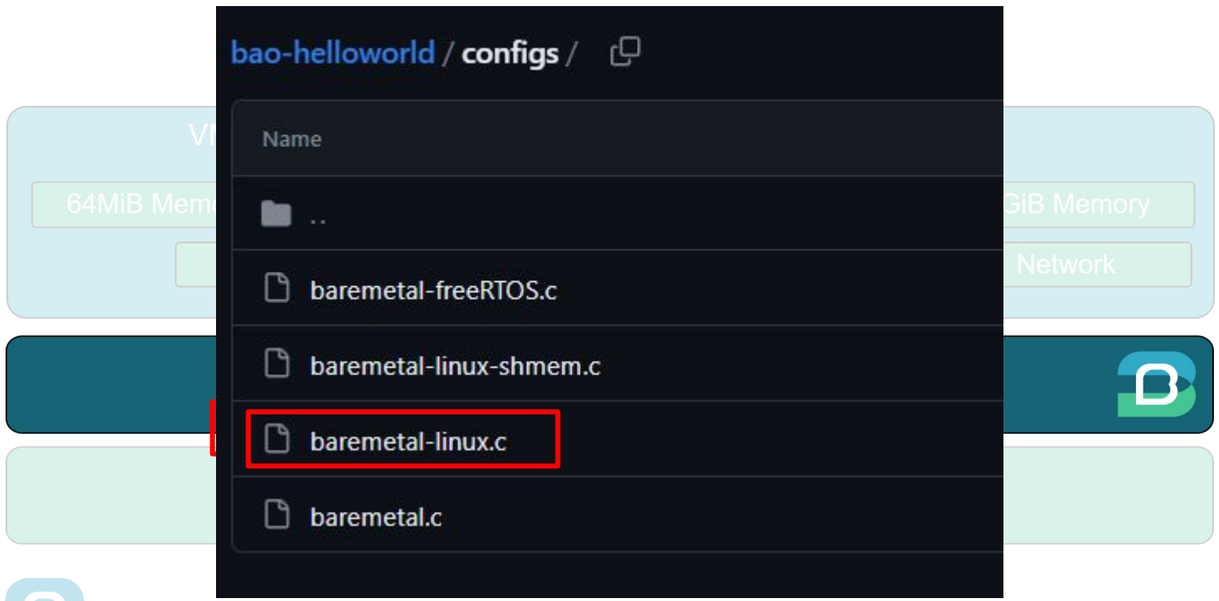
bao-helloworld / configs /



03 How to build Bao?

Compile the hypervisor

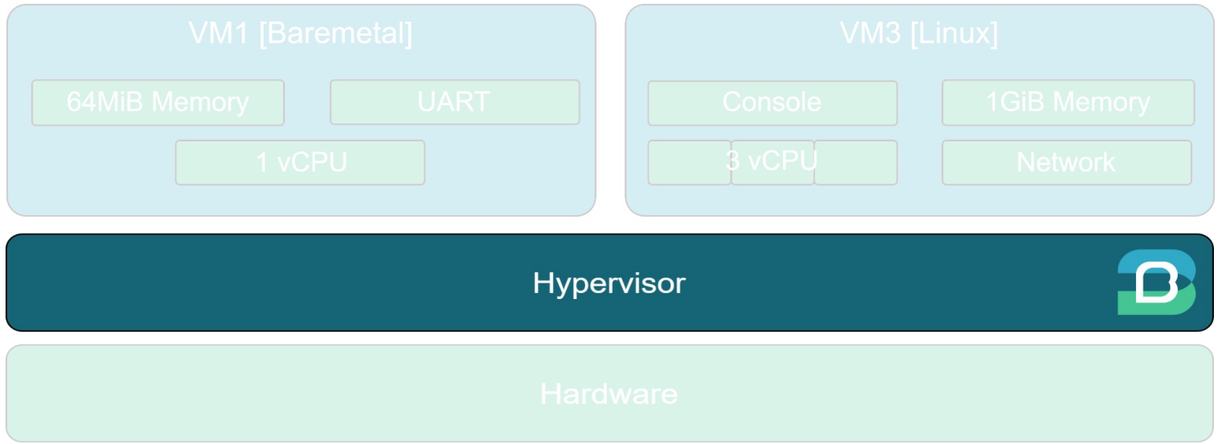
```
make -C $BAO_SRCS \
    PLATFORM=qemu-riscv64-virt \
    CONFIG_REPO=$ROOT_DIR/configs \
    CONFIG=baremetal-linux \
    CPPFLAGS=-DBAO_WRKDIR_IMGS=$BUILD_GUESTS_DIR
```



03 How to build Bao?

Move the binary file (bao.bin)

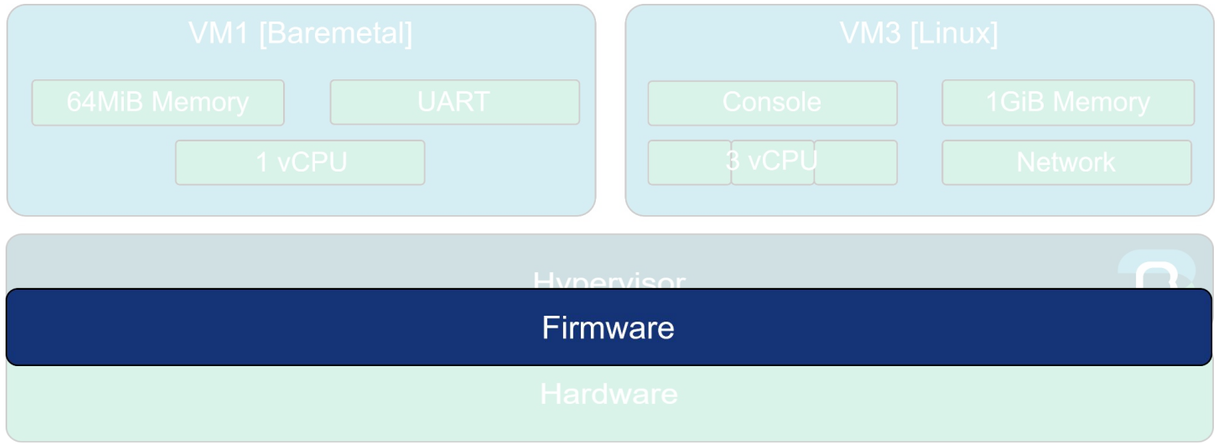
```
cp $BAO_SRCS/bin/qemu-riscv64-virt/baremetal-linux/bao.bin \
    $BUILD_BAO_DIR/bao.bin
```



04 How to build firmware?

Compile OpenSBI

```
make -C $TOOLS_DIR/OpenSBI PLATFORM=generic \
    FW_PAYLOAD=y \
    FW_PAYLOAD_FDT_ADDR=0x80100000 \
    FW_PAYLOAD_PATH=$BUILD_BAO_DIR/bao.bin
```



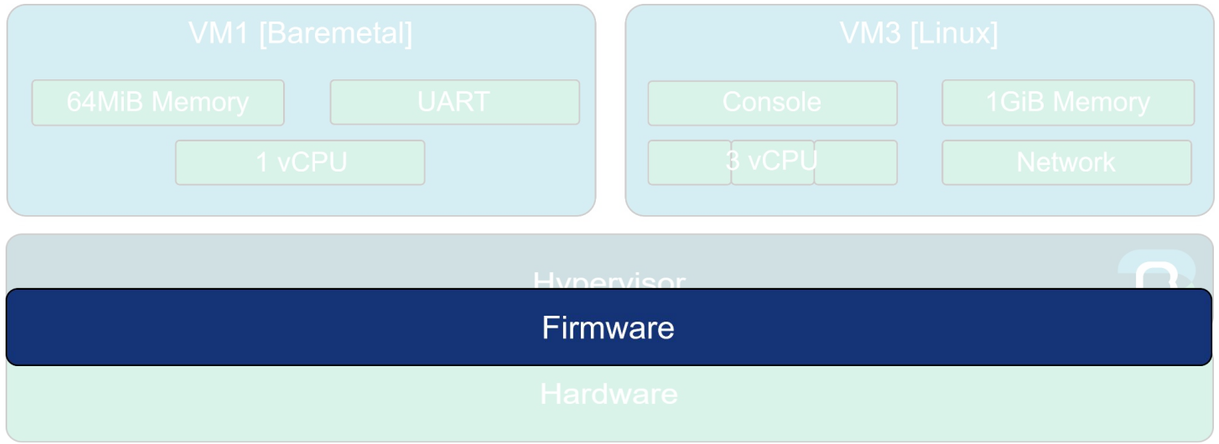
04 How to build firmware?

Compile OpenSBI

```
make -C $TOOLS_DIR/OpenSBI PLATFORM=generic \
    FW_PAYLOAD=y \
    FW_PAYLOAD_FDT_ADDR=0x80100000 \
    FW_PAYLOAD_PATH=$BUILD_BAO_DIR/bao.bin
```

Copy the compiled image

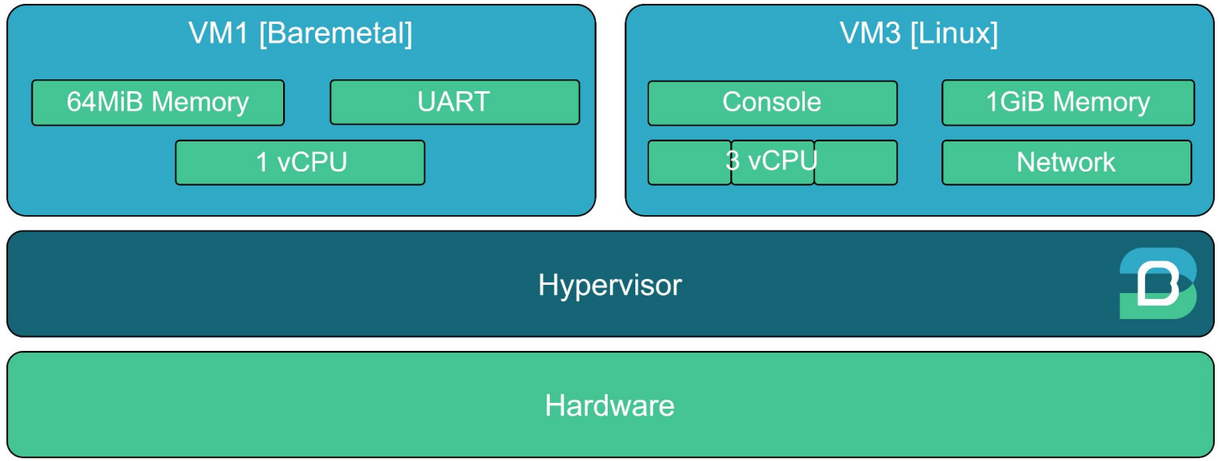
```
cp $TOOLS_DIR/OpenSBI/build/platform/generic/firmware/fw_payload.elf \
    $TOOLS_DIR/bin/opensbi.elf
```



04 How to run the demo?

Launch QEMU

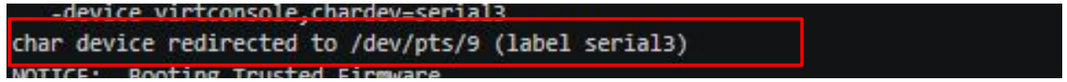
```
qemu-system-riscv64 -nographic \
    -M virt -cpu rv64 -m 4G -smp 4 \
    -bios $TOOLS_DIR/bin/opensbi.elf \
    -device virtio-net-device,netdev=net0 \
    -netdev user,id=net0,net=192.168.42.0/24,\
     hostfwd=tcp:127.0.0.1:5555-:22 \
    -device virtio-serial-device -chardev pty,id=serial3 \
    -device virtconsole,chardev=serial3
```



04 How to run the demo?

Launch QEMU

```
qemu-system-riscv64 -nographic\
    -M virt -cpu rv64 -m 4G -smp 4\
    -bios $TOOLS_DIR/OpenSBI.elf\
    -device virtio-net-device,netdev=net0 \
    -netdev user,id=net0,net=192.168.42.0/24,\
hostfwd=tcp:127.0.0.1:5555-:22\
    -device virtio-serial-device -chardev pty,id=serial3\
    -device virtconsole,chardev=serial3
```



Connect to the UART available to Linux

```
pyserial-miniterm --filter=direct /dev/pts/9
```

04 How to run the demo?

Check container ID

```
docker ps
```

```
CONTAINER ID   IMAGE                  COMMAND
9827a6c27e26   baoproject/bao:latest  "/bin/sh -c /bin/bash"
```

Open another terminal

```
# docker exec -it <container_id> bash
docker exec -it 9827a6c27e26 bash
```

Short Q&A

Inter-VM Communication

1x baremetal with 1x vCPUs; 1x Linux with 3 vCPUs, communicating through shared memory

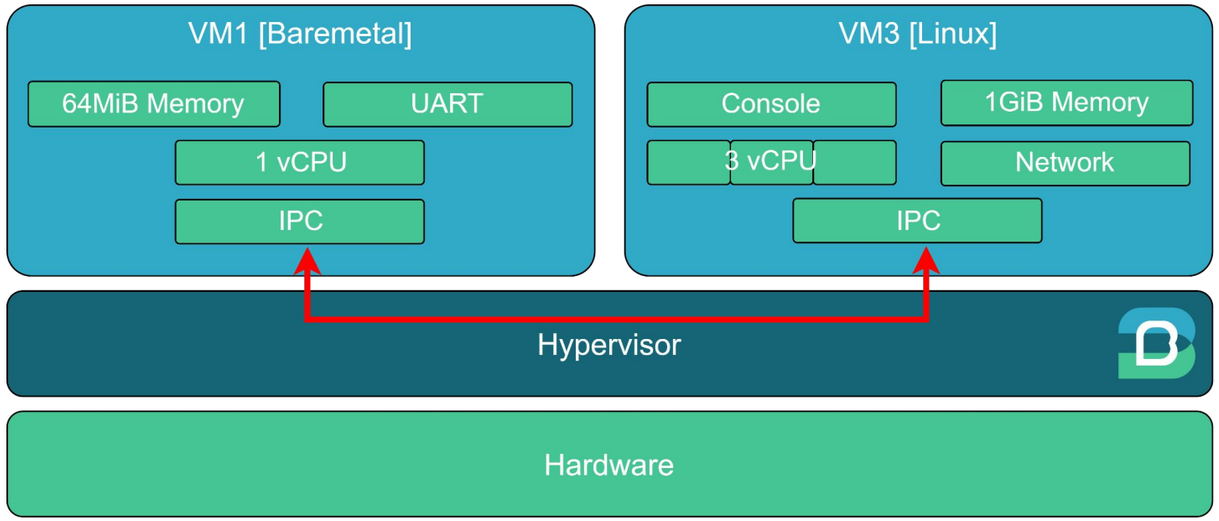
01 What are we building?

VM1

- Baremetal
- 1 vCPUs
- 64 MiB Memory
- UART
- IPC

VM2

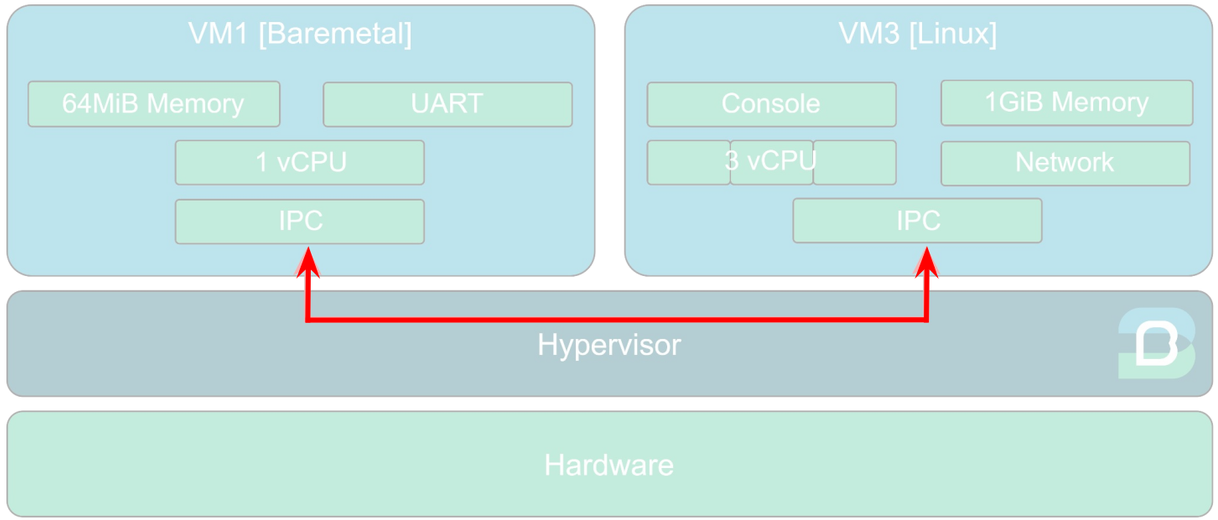
- Linux
- 3 vCPUs
- 1 GiB Memory
- VirtIO
- IPC



02 What to modify? Bao config

configs/baremetal-linux-shmem.c

```
struct config config = {
(...)
     // Create shared memory object/channel
+    .shmemlist_size = 1,
+    .shmemlist = (struct shmem[]) {
+        [0] = { .size = 0x00010000, }
+    },
```



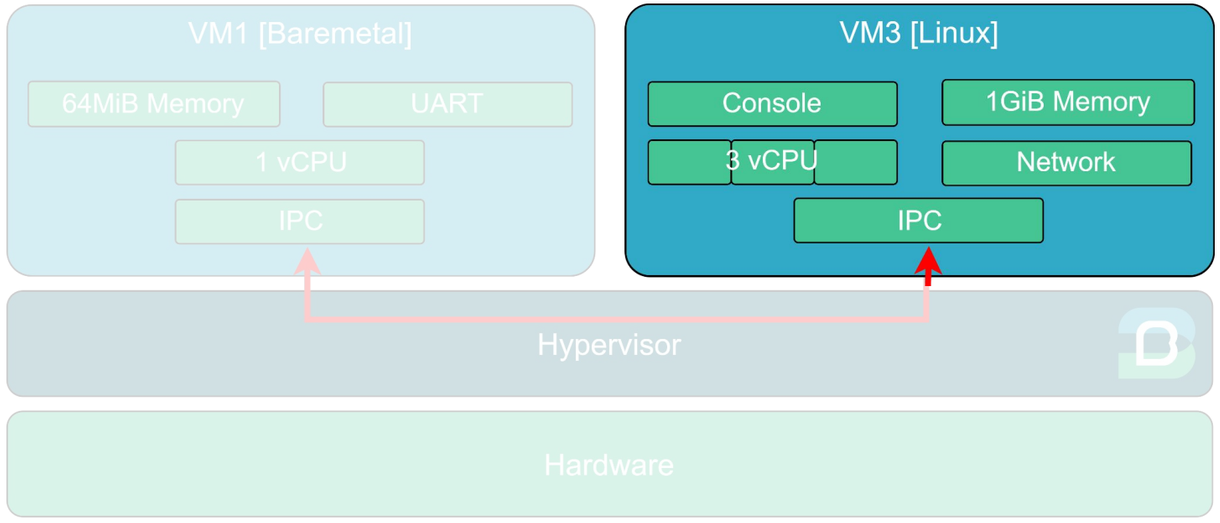
02 What to modify? Linux

configs/baremetal-linux-shmem.c

```
struct config config = {
(...)
     { // Linux guest configuration
+    .ipc_num = 1,
+    .ipcs = (struct ipc[]) {
+      {
+        .base = 0xf0000000,
+        .size = 0x00010000,
+        .shmem_id = 0,
+        .interrupt_num = 1,
+        .interrupts = (irqid_t[])
{52}
+      }
+    },
```

srcs/devicetrees/qemu-riscv64-virt/**linux.dts**

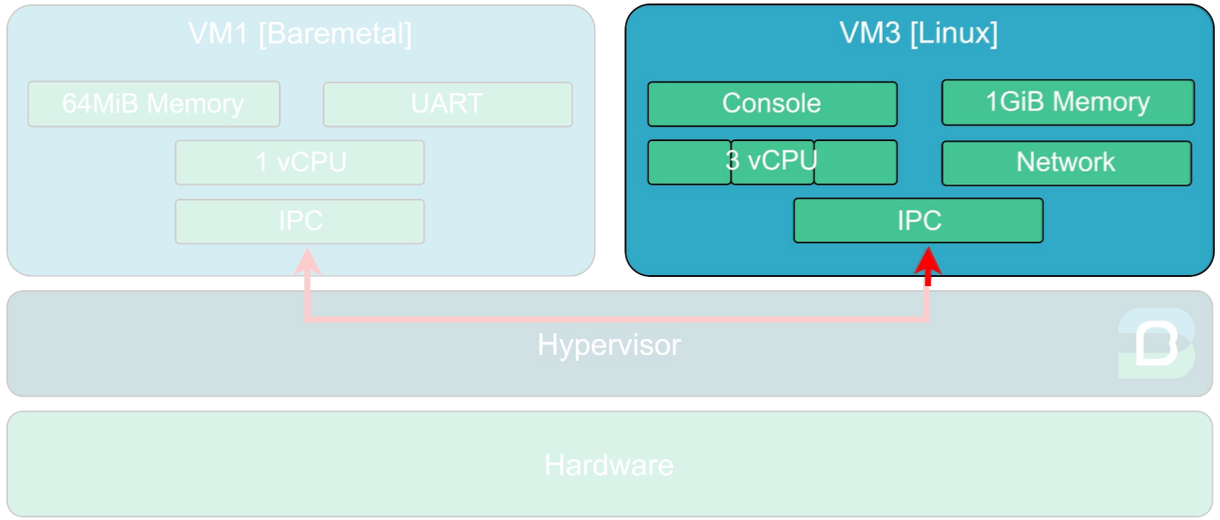
```
(...)
  bao-ipc@f0000000 {
     compatible = "bao,ipcshmem";
     reg = <0x0 0xf0000000 0x0 0x00010000>;
                read-channel = <0x0 0x2000>;
                write-channel = <0x2000 0x2000>;
     interrupts = <0 52 1>;
        id = <0>;
  };
(...)
```



02 What to integrate? Build

Generate the updated device tree

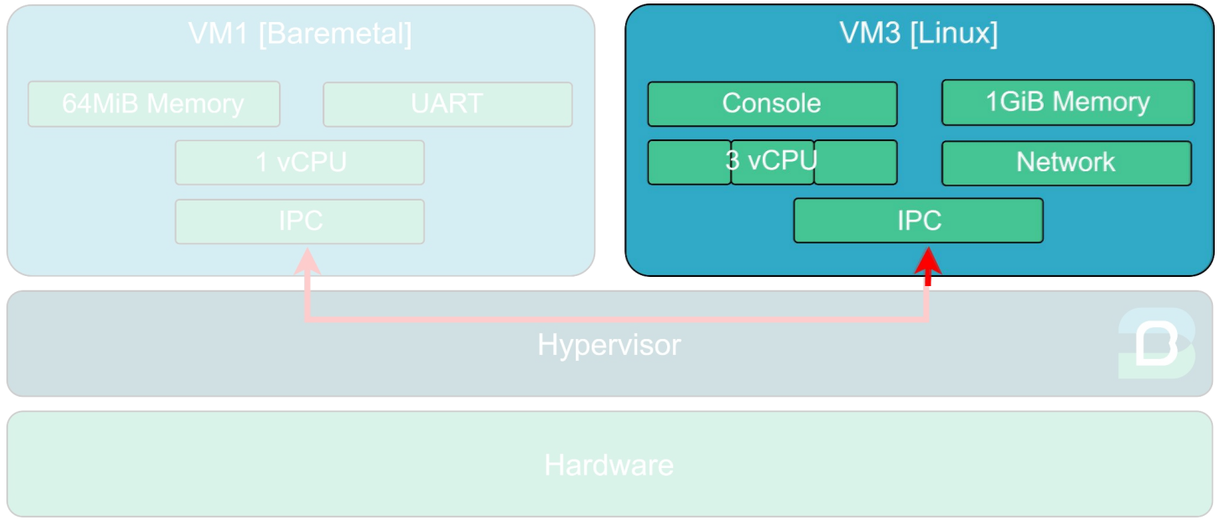
```
export LINUX_VM=linux-shmem
mkdir -p $BUILD_GUESTS_DIR/baremetal-linux-shmem-setup
dtc $ROOT_DIR/srcs/devicetrees/qemu-riscv64-virt/$LINUX_VM.dts > \
    $BUILD_GUESTS_DIR/baremetal-linux-shmem-setup/$LINUX_VM.dtb
```



02 What to integrate? Build

Wrap the kernel image and device tree blob into a single binary

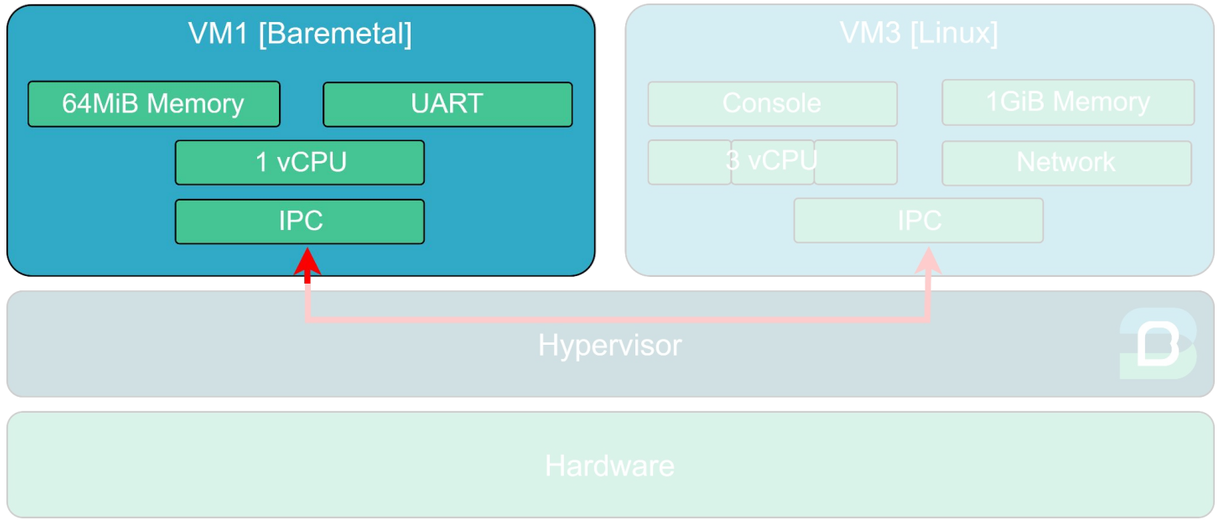
```
make -j $(nproc) -C $ROOT_DIR/srcs/lloader \
    ARCH=riscv64 \
    IMAGE=$PRE_BUILT_IMGS/guests/baremetal-linux-shmem-setup/Image-qemu-riscv64-virt \
    DTB=$BUILD_GUESTS_DIR/baremetal-linux-shmem-setup/$LINUX_VM.dtb \
    TARGET=$BUILD_GUESTS_DIR/baremetal-linux-shmem-setup/$LINUX_VM
```



03 What to modify? Bao config

configs/baremetal-linux-shmem.c

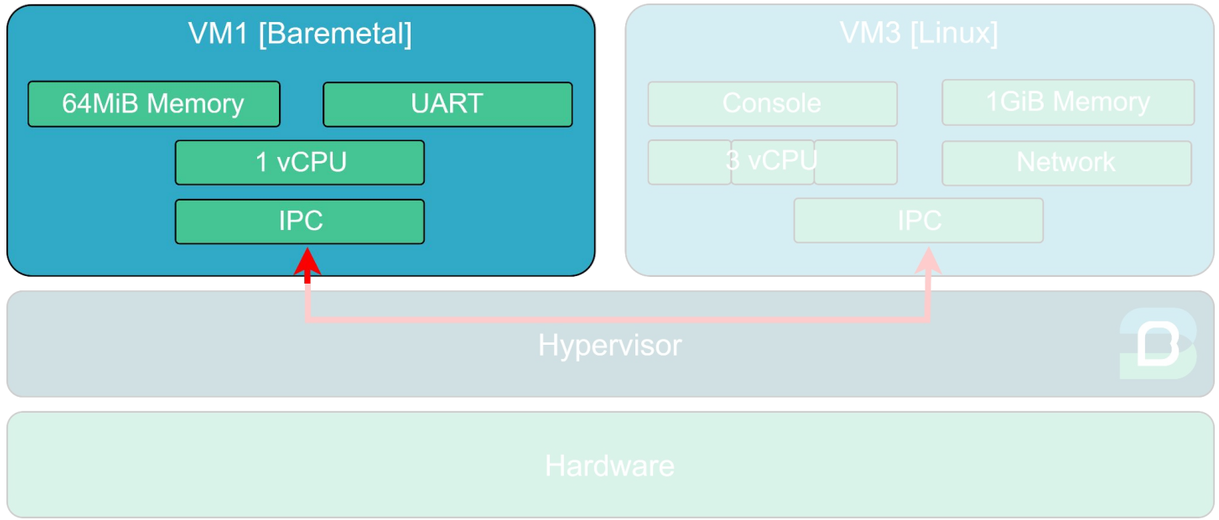
```
      // baremetal guest configuration
+                 .ipc num = 1,
+                 .ipcs = (struct ipc[]) {
+                     {
+                         .base = 0xf0000000,
+                         .size = 0x00010000,
+                         .shmem id = 0,
+                         .interrupt num = 1,
+                         .interrupts = (irqid_t[]) {52}
+                     }
+                 },
```



03 What to modify? Baremetal

Apply patch to handle IPC messages

```
git -C $BAREMETAL_SRCS apply $PATCHES_DIR/baremetal_shmem.patch
```



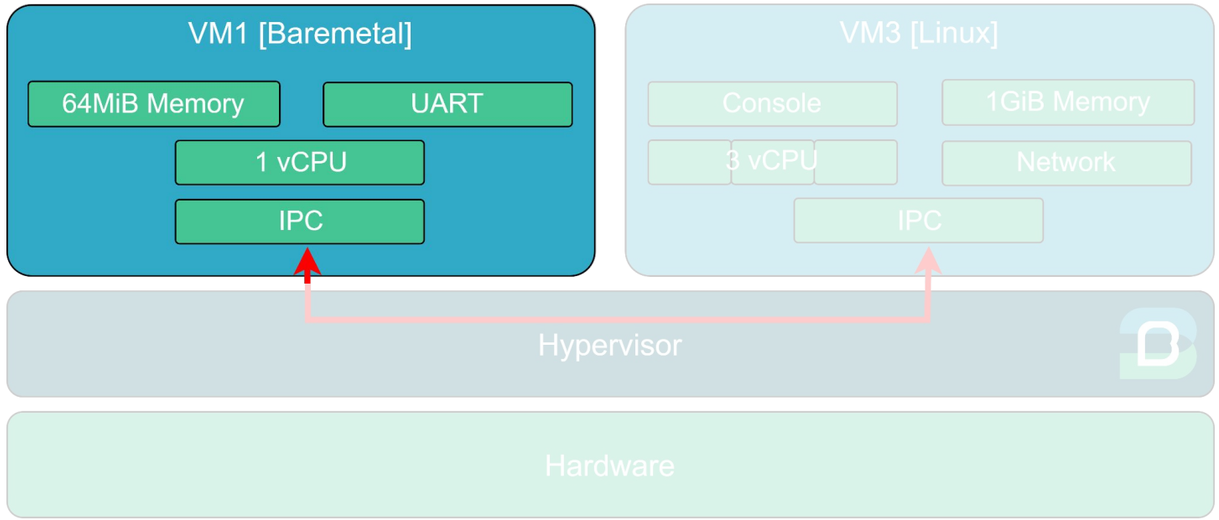
03 What to modify? Baremetal

Apply patch to handle IPC messages

```
git -C $BAREMETAL_SRCS apply $PATCHES_DIR/baremetal_shmem.patch
```

Recompile the baremetal

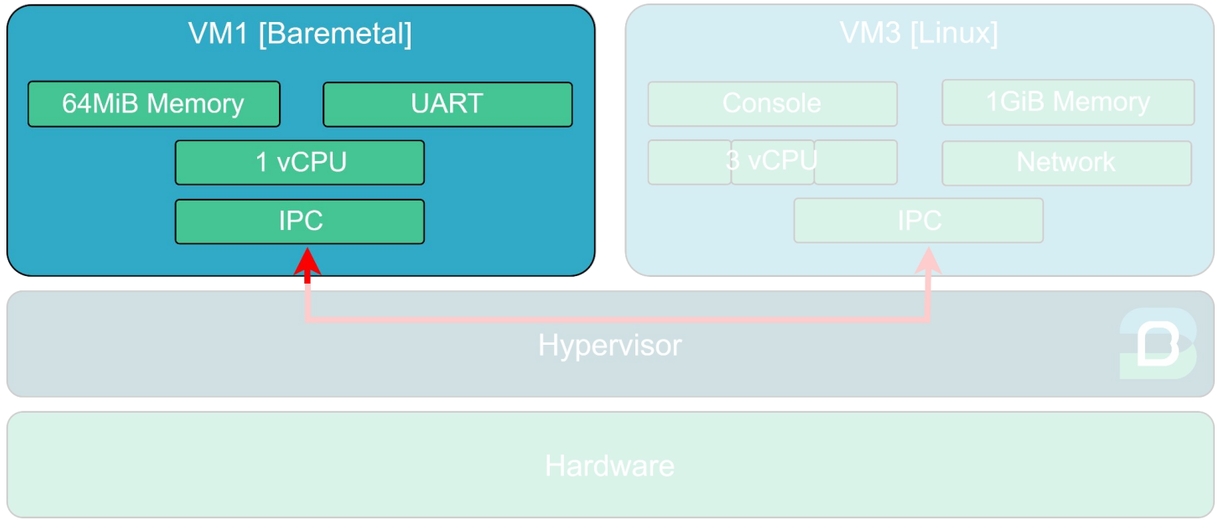
```
make -C $BAREMETAL_SRCS PLATFORM=qemu-riscv64-virt
```



03 What to modify? Baremetal

Move the binary file (baremetal.bin)

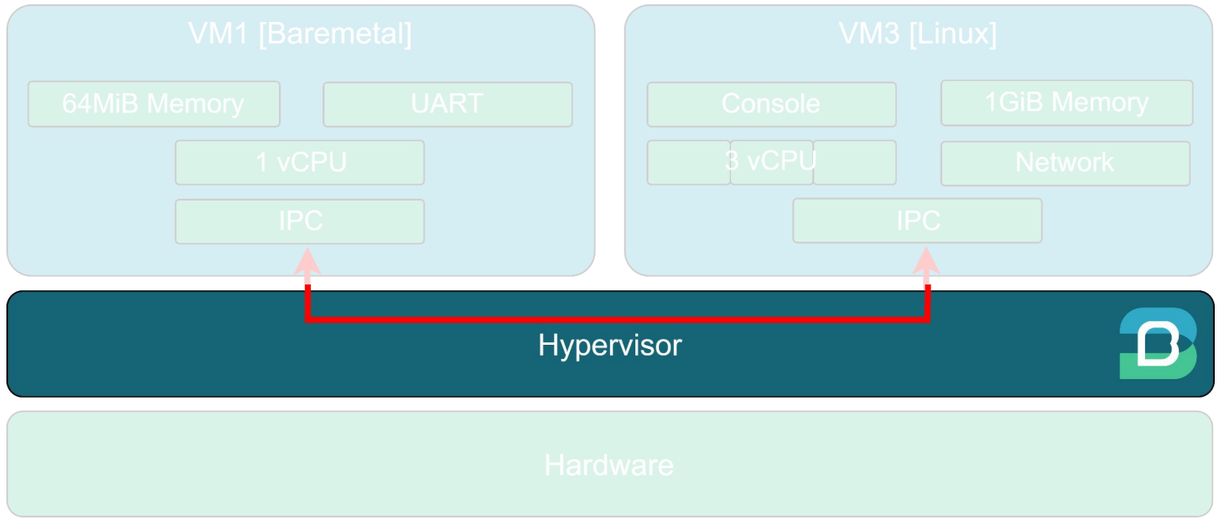
```
mkdir -p $BUILD_GUESTS_DIR/baremetal-setup
cp $BAREMETAL_SRCS/build/qemu-riscv64-virt/baremetal.bin \
    $BUILD_GUESTS_DIR/baremetal-linux-shmem-setup/baremetal.bin
```



03 How to build Bao?

Compile the hypervisor

```
make -C $BAO_SRCS \
    PLATFORM=qemu-riscv64-virt \
    CONFIG_REPO=$ROOT_DIR/configs \
    CONFIG=baremetal-linux-shmem \
    CPPFLAGS=-DBAO_WRKDIR_IMGS=$BUILD_GUESTS_DIR
```

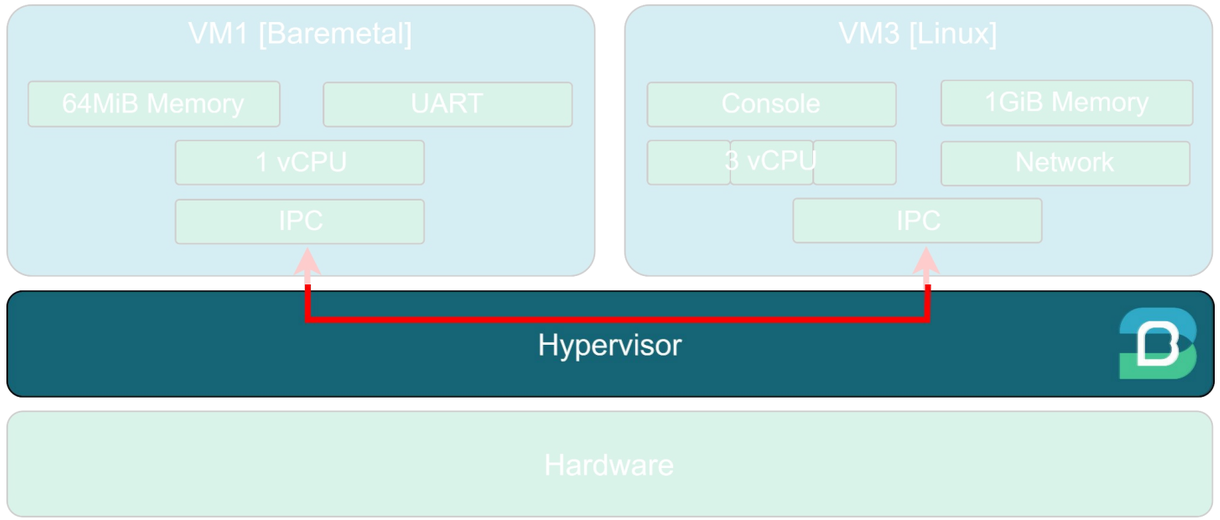


03 How to build Bao?

Compile the hypervisor

```
make -C $BAO_SRCS \
    PLATFORM=qemu-riscv64-virt \
    CONFIG_REPO=$ROOT_DIR/configs \
    CONFIG=baremetal-linux-shmem \
    CPPFLAGS=-DBAO_WRKDIR_IMGS=$BUILD_GUESTS_DIR
```

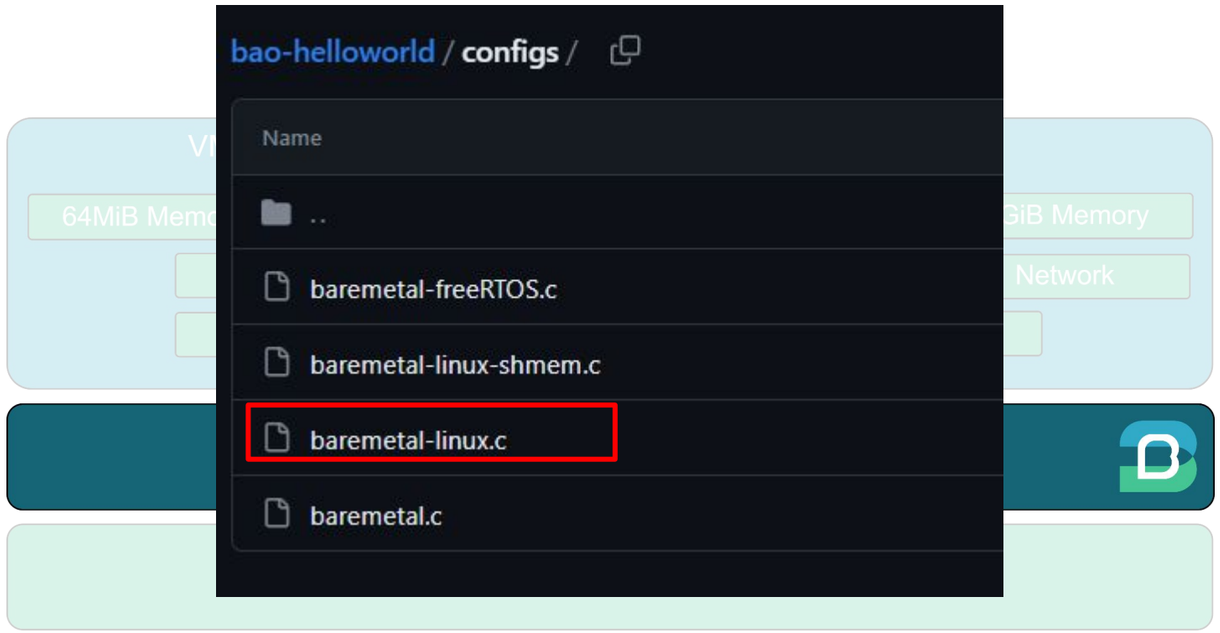
bao-helloworld / configs /



03 How to build Bao?

Compile the hypervisor

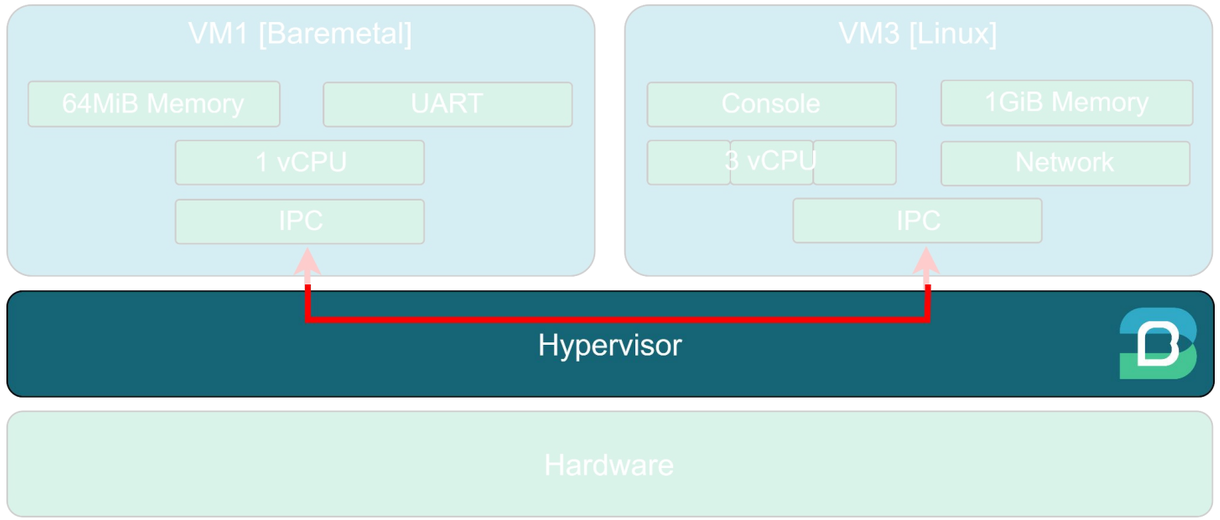
```
make -C $BAO_SRCS \
    PLATFORM=qemu-riscv64-virt \
    CONFIG_REPO=$ROOT_DIR/configs \
    CONFIG=baremetal-linux-shmem \
    CPPFLAGS=-DBAO_WRKDIR_IMGS=$BUILD_GUESTS_DIR
```



03 How to build Bao?

Move the binary file (bao.bin)

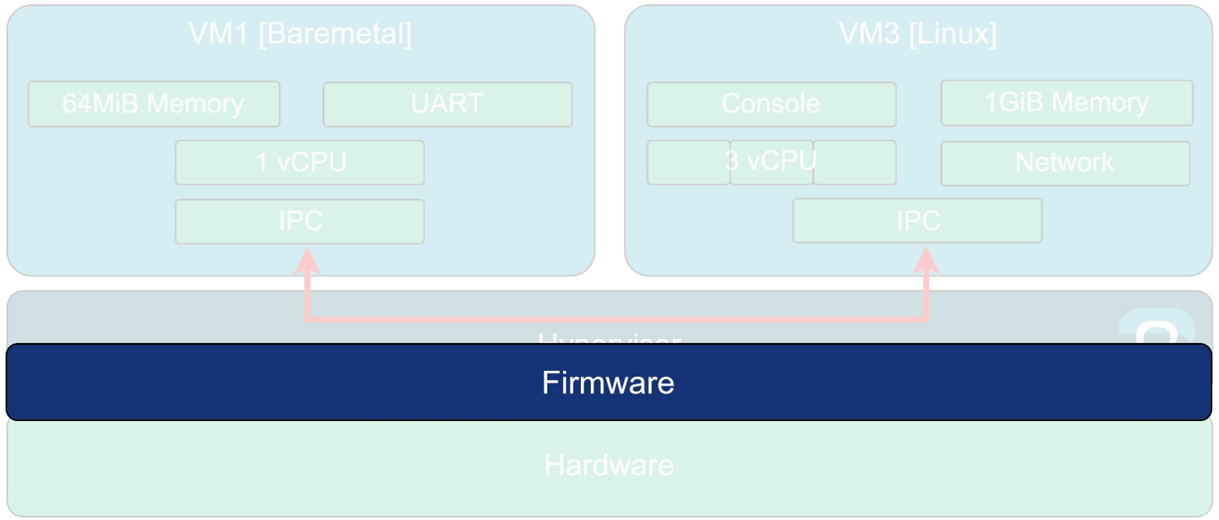
```
cp $BAO_SRCS/bin/qemu-riscv64-virt/baremetal-linux-shmem/bao.bin \
    $BUILD_BAO_DIR/bao.bin
```



04 How to build firmware?

Compile OpenSBI

```
make -C $TOOLS_DIR/OpenSBI PLATFORM=generic \
    FW_PAYLOAD=y \
    FW_PAYLOAD_FDT_ADDR=0x80100000 \
    FW_PAYLOAD_PATH=$BUILD_BAO_DIR/bao.bin
```



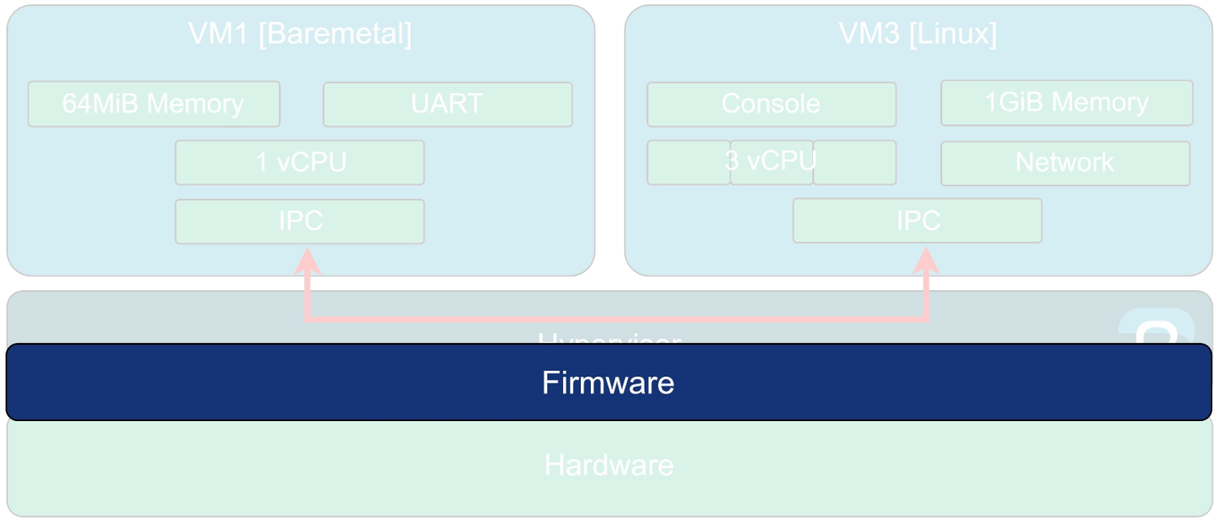
04 How to build firmware?

Compile OpenSBI

```
make -C $TOOLS_DIR/OpenSBI PLATFORM=generic \
    FW_PAYLOAD=y \
    FW_PAYLOAD_FDT_ADDR=0x80100000 \
    FW_PAYLOAD_PATH=$BUILD_BAO_DIR/bao.bin
```

Copy the compiled image

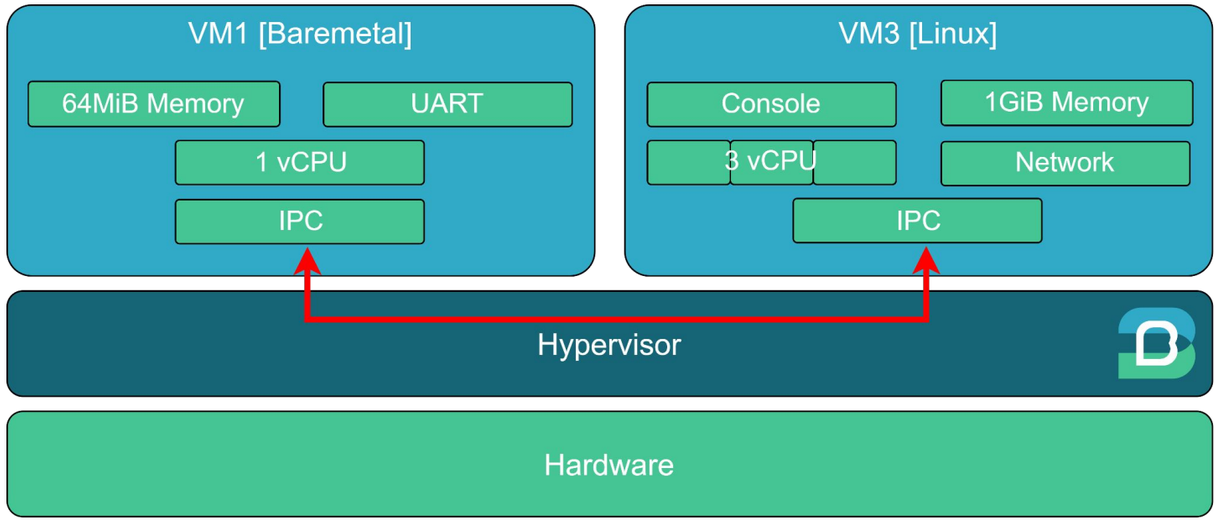
```
cp $TOOLS_DIR/OpenSBI/build/platform/generic/firmware/fw_payload.elf \
    $TOOLS_DIR/bin/opensbi.elf
```



04 How to run the demo?

Launch QEMU

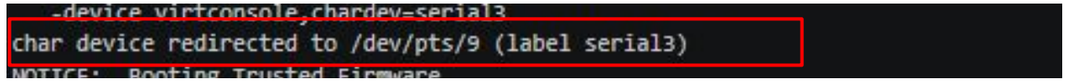
```
qemu-system-riscv64 -nographic \
    -M virt -cpu rv64 -m 4G -smp 4 \
    -bios $TOOLS_DIR/bin/opensbi.elf \
    -device virtio-net-device,netdev=net0 \
    -netdev user,id=net0,net=192.168.42.0/24,\
     hostfwd=tcp:127.0.0.1:5555-:22 \
    -device virtio-serial-device -chardev pty,id=serial3 \
    -device virtconsole,chardev=serial3
```



04 How to run the demo?

Launch QEMU

```
qemu-system-riscv64 -nographic \
    -M virt -cpu rv64 -m 4G -smp 4 \
    -bios $TOOLS_DIR/bin/opensbi.elf \
    -device virtio-net-device,netdev=net0 \
    -netdev user,id=net0,net=192.168.42.0/24,\
     hostfwd=tcp:127.0.0.1:5555-:22 \
    -device virtio-serial-device -chardev pty,id=serial3 \
    -device virtconsole,chardev=serial3
```



Connect to guest Linux UART

```
pyserial-miniterm --filter=direct /dev/pts/9
```

Short Q&A

Bao Demos

Live demos on real hardware platforms (Raspberry Pi)

bao-project / **bao-demos** Public

master **6** branches **0** tags Go to file Code

yanjiew1 and **josecm** fix(demos/linux+zephyr): fix linux+zephyr demo on fv... ... 40e843a 3 weeks ago **73** commits

demos	fix(demos/linux+zephyr): fix linux+zephyr demo on fvp-a	3 weeks ago
guests	Add support for linux+zephyr on rpi4	3 weeks ago
platforms	update(plat/qemu): cmd requires net=IP_ADDR/RANGE for SSH to r...	3 weeks ago
.gitignore	initial commit	2 years ago
COPYING	initial commit	2 years ago
Makefile	fix(makefile): correct platform variable on clean recipe	5 months ago
README.md	Add support for linux+zephyr on rpi4	3 weeks ago

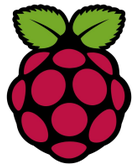
README.md

Bao Hypervisor Demo Guide

This tutorial provides a step-by-step guide on how to run different demo configurations of the Bao hypervisor featuring multiple guest operating systems and targeting several supported platforms. The available demos are:

- Single-guest Baremetal
- Dual-guest Linux+FreeRTOS
- Dual-Guest Linux+Zephyr
- Dual-Guest Zephyr+Baremetal

01 RPI: Bao+Linux+FreeRTOS



https://github.com/bao-project/bao-demos

1. Build Linux Kernel Guest
2. Build Freertos
3. Build Bao Hypervisor
4. Build Firmware (RPI4 FW, U-boot, & TF-A)
5. Setup SD
6. Setup board
7. Run U-boot commands to start

Advanced Bao Configuration

List of advanced features and configurations beyond this workshop

Static Allocations

- Allows to configure the physical region on which a **`vm_mem_region`** is mapped.
- Forces contiguous mappings
- Use the **`place_phys`** and **`phys`** fields in the region configuration.
- Straightforward way to obtain identity mappings
- Bao will check this is a valid address and is not allocated elsewhere
- Not to be used for OCM SRAMS! (these are considered devices)

```
(…)
            .regions =  (struct vm_mem_region[]) {
                {
                    .base =  0x80200000,
                    .size =  0x4000000,
                    .place_phys = true,
                    .phys = 0x80200000,
                }
            },
(…)
```

IOMMU DMA Devices

- DMA devices **need** to be **isolated** by the **IOMMU**.
  - In the Bao configuration, users need to add the Bus Master ID used by the IOMMU to control this device in the **`id`** field.
- **Master IDs** are either:
  - typically found in the SoC TRM.
  - **programmable** using **firmware** facilities.
- Specifically for Arm's SMMUv2, it is possible to **configure groups**.

```
(…)
                .devs =  (struct vm_dev_region[]) {
                        {
                            .pa =  0x20000000,
                            .va =  0x20000000,
                            .size =  0x1000,
                            .interrupt_num =  1,
                            .interrupts = ( irqid_t[]) {12},
                            .id = 0xde71d,
                        },
                    },

                    .arch = {
(…)
                        .smmu = {
                            .global_mask = 0xF,
                            .group_num = 0,
                            .smmu_groups = (struct smmu_group[]) {
                                {
                                    .group_mask = 0x70f,
                                    .group_id = 0x877
                                }
                            }
                        }
                    }
(…)
```

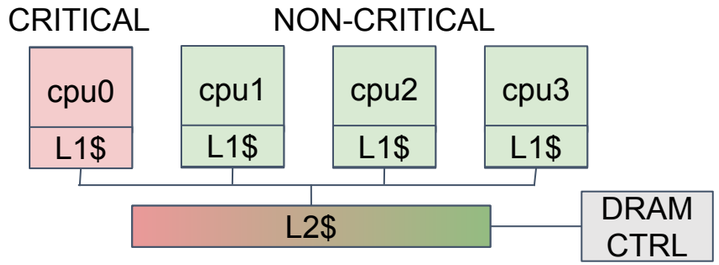
MPU Targets

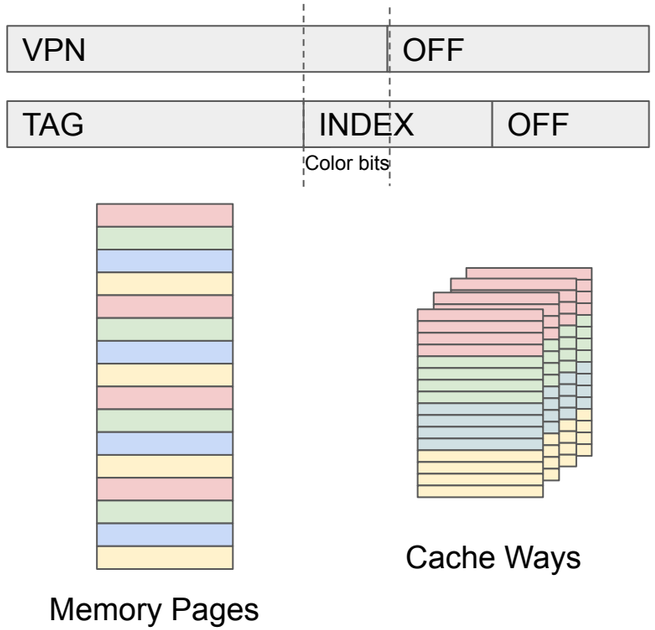
- For all memory regions identity map is implied. That is, **`place_phys`** with **`phys`** equal to **`base`**.
- For devices the **`va`** is ignored.
- Guest images should be separately loaded (use **`VM_IMAGE_LOADED`**)
- Possible to configure **Bao's base**.
- For Armv8-R AArch64, guest MMU can be enabled for each guest.

```
struct config config = {

    // configure hypervisor base address (optional)
    .hyp = {
        .relocate = true,
        .base_addr = 0x10000000,
    },

    .vmlist_size = 1,
    .vmlist = {
        {
           // indicate where the guest img was loaded and its
size
            .image = VM_IMAGE_LOADED(img, 0x80200000, 96 * 1024),

(…)

            // for Armv8-R AArch64 enable guest MMU
           .mmu = true,
(…)
```

Cache Coloring

- Cache or page coloring is a mechanism for shared cache set-partitioning.
- Avoids interference by eliminating inter-VM conflict misses
- **Possible to color individually:**
  - hypervisor
  - guests
  - shared memory
- **Disadvantages:**
  - Memory fragmentation;
  - Precludes the use of super-pages.
- Number of available colors is platform dependent.
- (Only for MMU platforms)






```
struct config config = {

    .hyp = {
        .colors = 0xf0,
    },

    .vmlist_size = 1,
    .vmlist = {
        {
        .colors = 0x0f,
(…)
```

Other Misc.

- **Mapping guest image inplace:**
  - Faster VM boot
  - May preclude use of superpages
- **Set cpu affinity:**
  - Which CPUs, vCPUs are assigned to;
  - Inorder bitmap:
  - each bit represents a physical CPU
- **Boot requirements (MMU systems):**
  - Load bao image on a 2MiB boundary.
  - Ideally at the base of a platform memory region

```
struct config config = {

(…)

    .vmlist = {
        {

            .image = VM_IMAGE_BUILTIN_INPLACE(img, 0x80200000),
            .cpu_affinity = 0x5

(…)
```

Conclusion and Roadmap

Takeaway points and roadmap

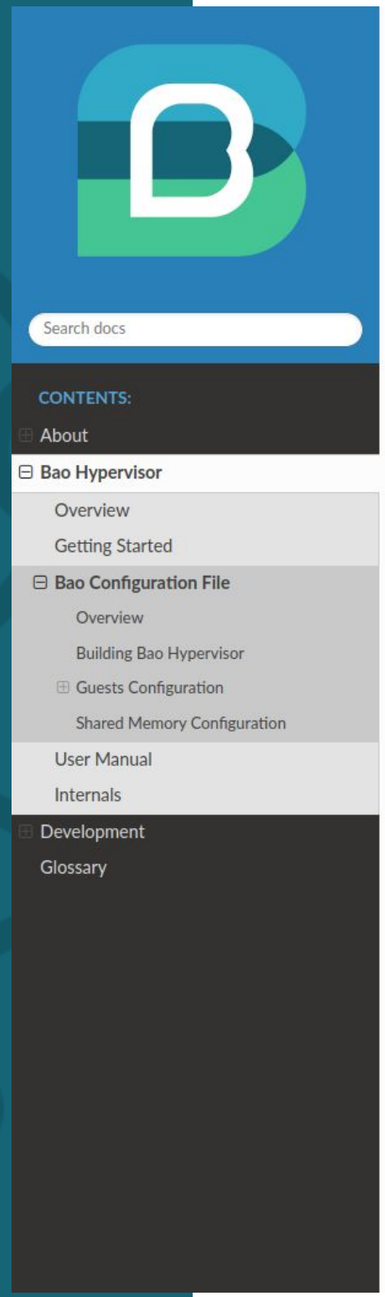
Conclusion

Takeaway Points

- Virtualization and static partitioning concepts
- Configure Bao VM resources
- Configure Bao Inter-VM communication
- Build Bao software stack components:
  - Guests
  - Hypervisor
  - Firmware
- Bao demonstration on a real hardware platform
- Bao advanced configurations

Roadmap

- **Documentation:**
  - User manual;
  - Troubleshooting;
  - Porting guide;
  - Internals



Bao Hypervisor / Bao Configuration File View page source

Bao Configuration File

Overview

The Bao hypervisor's configuration is managed through a dedicated configuration in the form of a C source file. This section provides an in-depth description of the various configuration options available.

The configuration file follows a specific structure, as outlined below:

```c
#include <config.h>

// Load guests' image
VM_IMAGE(img1_name, "/path/to/vm1/binary.bin");
VM_IMAGE(img2_name, "/path/to/vm2/binary.bin");

struct config config = {

    // Shared memory region configuration
    .shmemlist_size = N,
    .shmemlist = (struct shmem[]) {
        [0] = {/*shared memory config*/,},
        [1] = {/*shared memory config*/,},
        ...
        [N] = {/*shared memory config*/,}
    },

    // Guests Configuration
    .vmlist_size = NUM_VMs,
    .vmlist = {
        { /* VM 0 Config*/},
        { /* VM 1 Config*/},
        ...
        { /* VM N Config*/},
    }
};
```

Before the configuration itself, it is necessary to declare the VM images using the `VM_IMAGE` macro. This macro directly embeds the guest binary file into the hypervisor image. Here's an example usage of the `VM_IMAGE`:

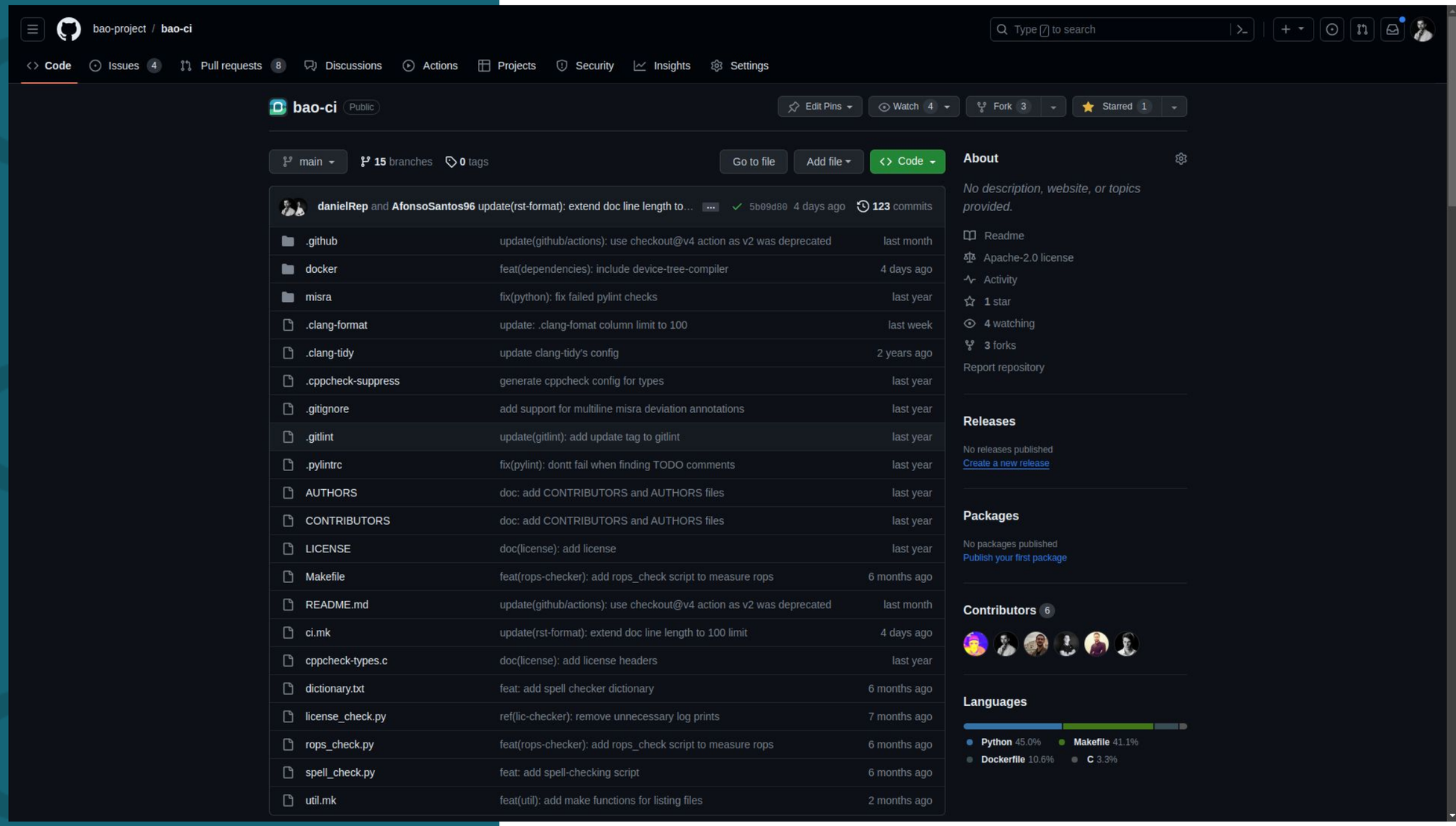
```c
VM_IMAGE(img_name, "/path/to/VM/binary.bin");
```

The `VM_IMAGE` macro has two parameters:

Roadmap

- **Documentation:**
  - User manual;
  - Troubleshooting;
  - Porting guide;
  - Internals

- **CI:**
  - Code quality
  - Testing

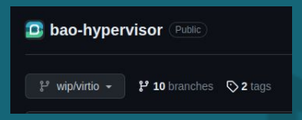


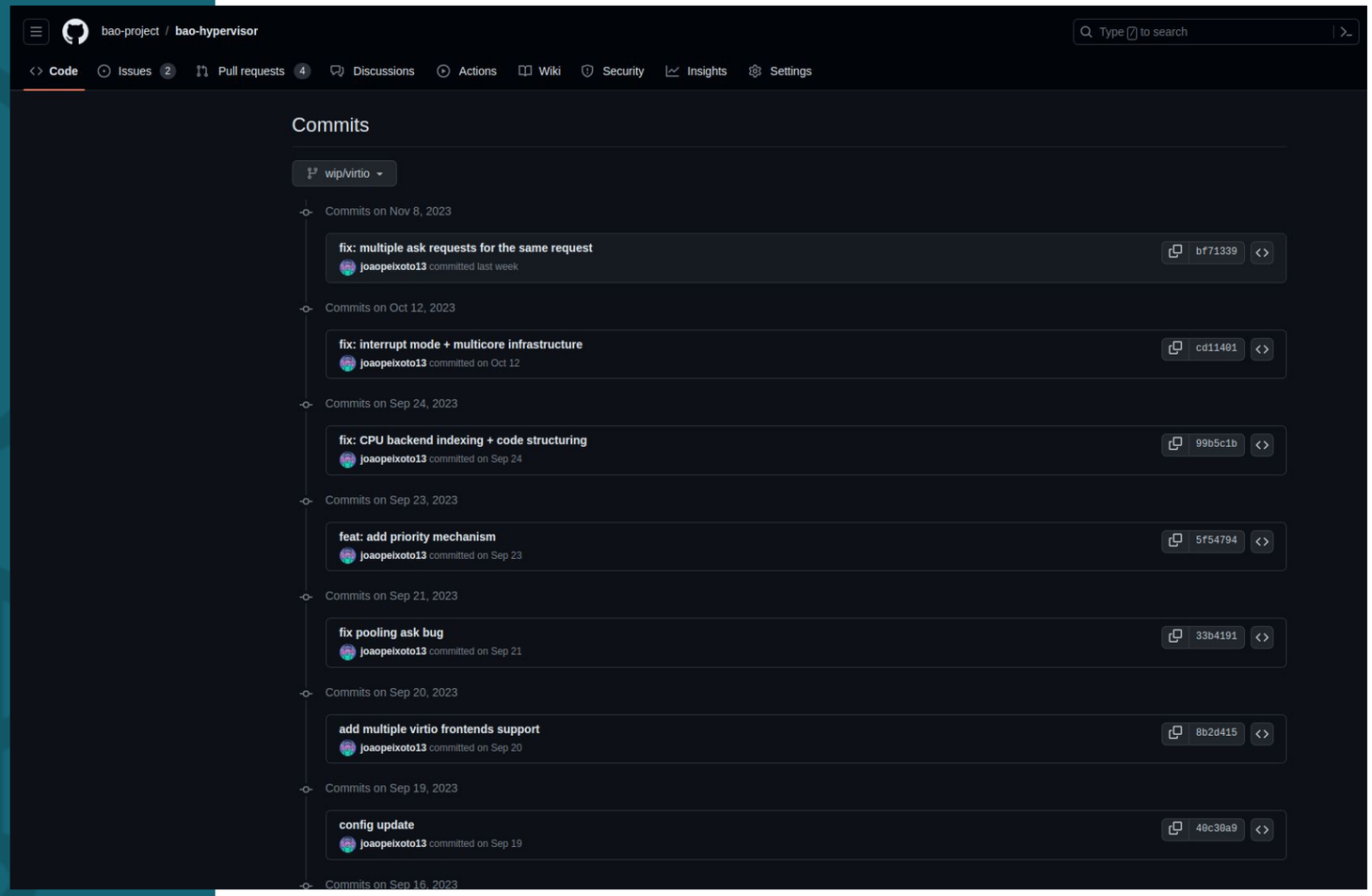
Roadmap

- **Documentation:**
  - User manual;
  - Troubleshooting;
  - Porting guide;
  - Internals

- **CI:**
  - Code quality
  - Testing

- Next in line features:
  - **Sharing:**
    - VirtIO (**support soon**)
    - CPU sharing
  - **MPU architectures:**
    - RISC-V sPMP
    - Infineon Aurix TC4
  - **UX:**
    - Configuration tools (guide, checks)
    - DTB generation
  - **Safety:**
    - MISRA compliance
    - Bandwidth regulation





THANK YOU!

Follow Bao for News and Updates!

Send your feedback (Github is our first choice!)

info@bao-project.org

Q&A