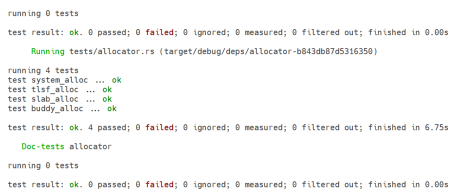
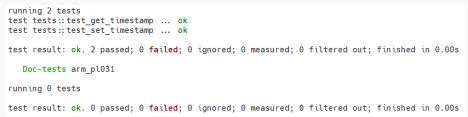
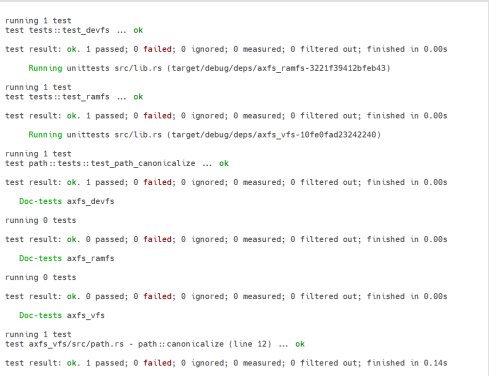
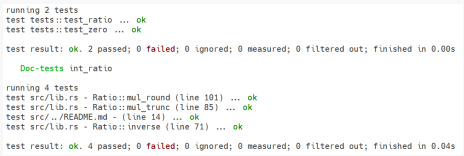
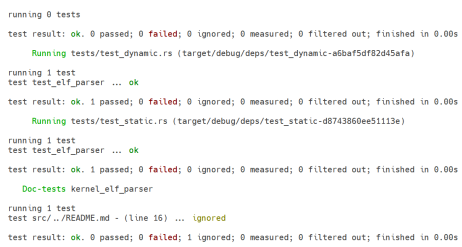
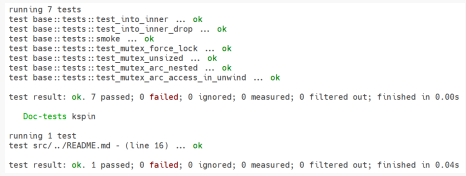
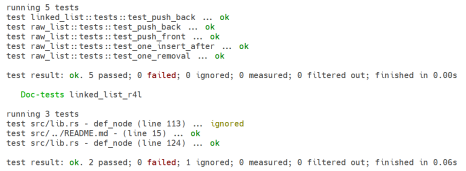
序号	指标项	复现情况
1.1	30个以上OS无关的功能组件	50个 ✅
1.2	20个以上与OS相关的功能组件	6个✅
1.3	可在物理设备上运行的unikernel架构的OS内核主干	已复现 ✅
1.4	可在物理设备上运行的monolithickernel架构的OS内核主干	已复现 ✅
1.5	国产处理器对OS编译环境的支持	已复现 ✅
2.1	在x86硬件平台上支持运行redis，相对Linux下，性能提升10%以上，避免Linux中的常见内存缺陷	已复现 ✅
2.2	在x86硬件平台上支持运行tokio，相对Linux下，性能提升10%以上，避免Linux中的常见内存缺陷	已复现 ✅

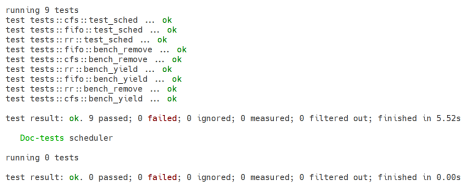
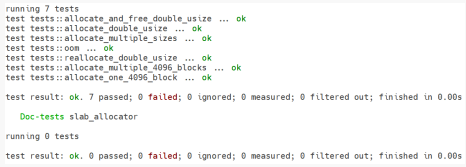
【1.1】 30个以上OS无关的功能组件 ✅

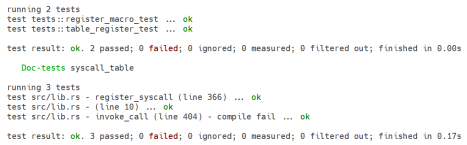
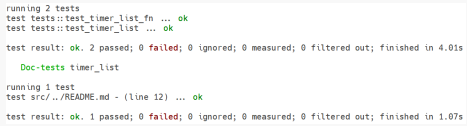
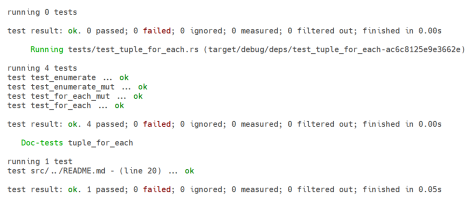
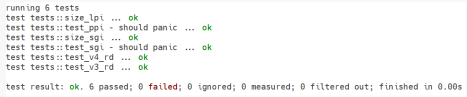
总数：50个

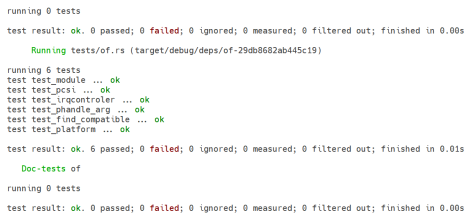
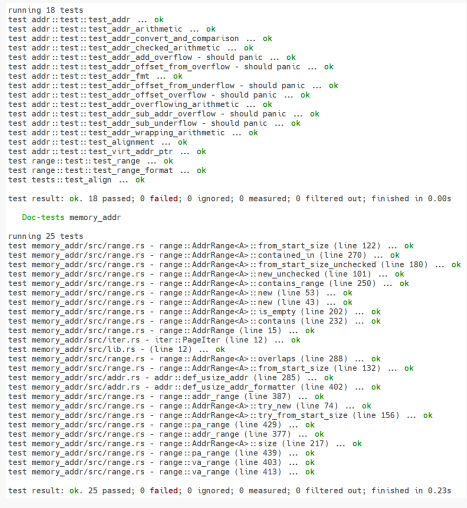
序号	已完成的组件	项目链接	文档	复现情况
1	allocator	https://github.com/arceos-org/allocator	https://os-checker.github.io/docs/kern-crates/allocator/allocator/	
2	arm_pl031	https://github.com/arceos-org/arm_pl031	https://os-checker.github.io/docs/kern-crates/arm_pl031/arm_pl031/	

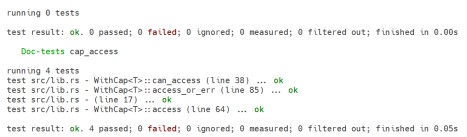
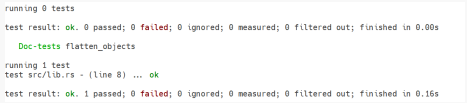
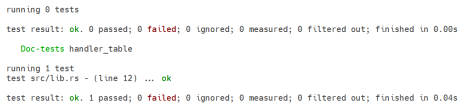
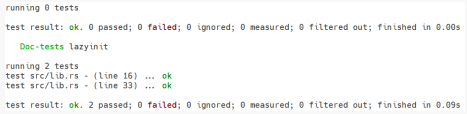
序号	已完成的组件	项目链接	文档	复现情况
3	axfs_crates	https://github.com/arceos-org/axfs_crates	https://arceos.org/axfs_crates/	
4	crate_interface	https://github.com/arceos-org/crate_interface	https://os-checker.github.io/docs/kern-crates/crate_interface/crate_interface/	
5	elf_parser_rs	https://github.com/Azure-stars/elf_parser_rs	https://os-checker.github.io/docs/kern-crates/elf_parser_rs/elf_parser_rs/	
6	int_ratio	https://github.com/arceos-org/int_ratio	https://os-checker.github.io/docs/kern-crates/int_ratio/int_ratio/	

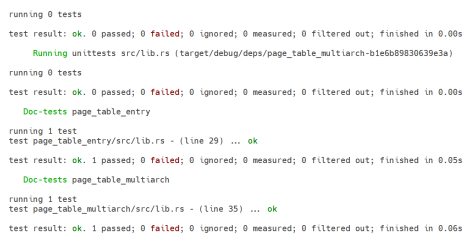
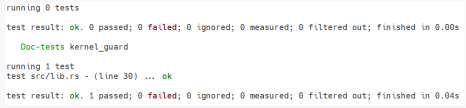
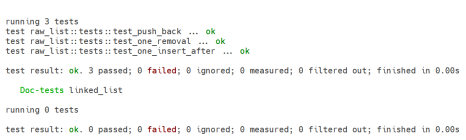
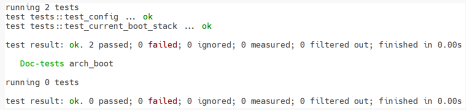
序号	已完成的组件	项目链接	文档	复现情况
7	kernel-elf-parser	https://github.com/Starry-OS/kernel-elf-parser	https://docs.rs/kernel-elf-parser	
8	kspin	https://github.com/arceos-org/kspin	https://os-checker.github.io/docs/kern-crates/kspin/kspin/	
9	linked_list_r4l	https://github.com/arceos-org/linked_list_r4l	https://os-checker.github.io/docs/kern-crates/linked_list_r4l-10/linked_list_r4l/	
10	memory_set	https://github.com/arceos-org/memory_set	https://os-checker.github.io/docs/kern-crates/axmm_crates/memory_set/	

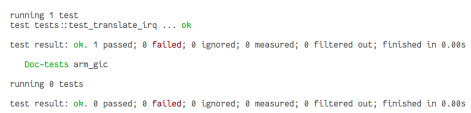
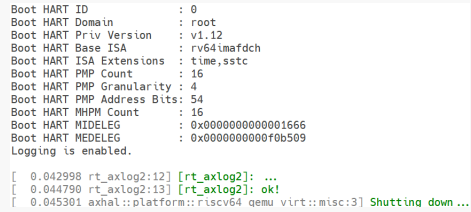
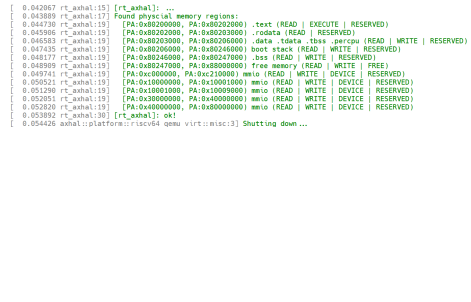
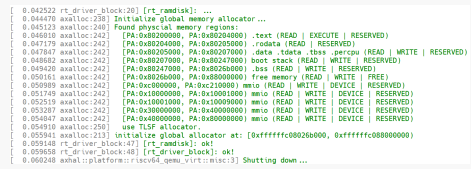
序号	已完成的组件	项目链接	文档	复现情况
11	pager	https://github.com/os-module/pager	https://os-checker.github.io/docs/kern-crates/pager/pager/	
12	rvfs	https://github.com/os-module/rvfs	https://os-checker.github.io/docs/kern-crates/rvfs/vfscore/	
13	scheduler	https://github.com/arceos-org/scheduler	https://os-checker.github.io/docs/kern-crates/scheduler/scheduler/	
14	slab_allocator	https://github.com/arceos-org/slab_allocator	https://arceos.org/slab_allocator/slab_allocator/index.html	

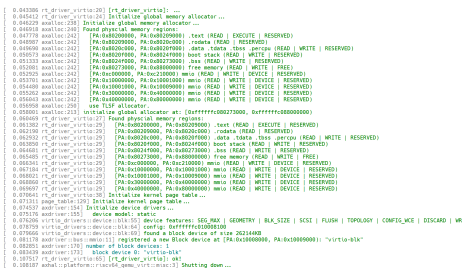
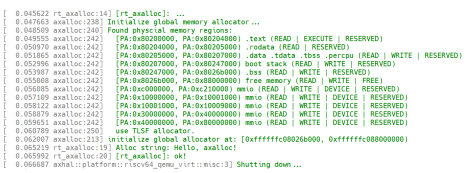
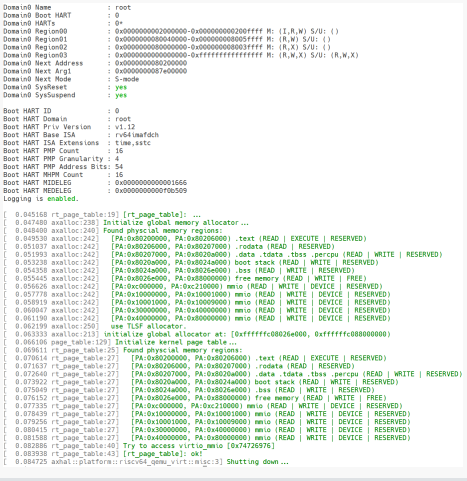
序号	已完成的组件	项目链接	文档	复现情况
15	syscall-table	https://github.com/os-module/syscall-table	https://os-checker.github.io/docs/kern-crates/syscall-table/syscall_table/	
16	timer_list	https://github.com/arceos-org/timer_list	https://os-checker.github.io/docs/kern-crates/timer_list/timer_list/	
17	tuple_for_each	https://github.com/arceos-org/tuple_for_each	https://os-checker.github.io/docs/kern-crates/tuple_for_each/tuple_for_each/	
18	arm-gic-driver	https://github.com/ZR233/arm-gic-driver	https://os-checker.github.io/docs/ZR233/arm-gic-driver/arm_gic_driver/	

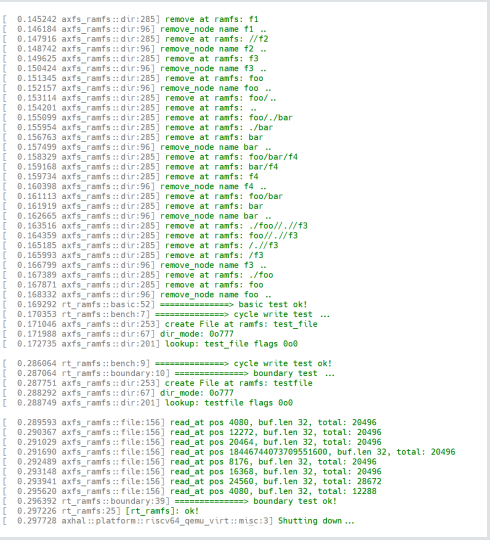
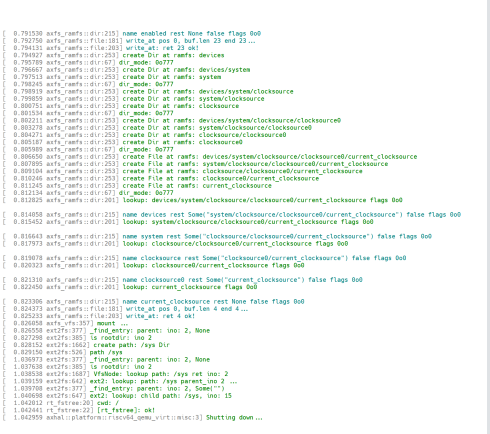
序号	已完成的组件	项目链接	文档	复现情况
19	percpu	https://github.com/arceos-org/percpu	https://os-checker.github.io/docs/kern-crates/percpu/percpu/	
20	fdt-parser	https://github.com/qclic/fdt-parser	https://os-checker.github.io/docs/qclic/fdt-parser/fdt_parser/	
21	of	https://github.com/Starry-OS/of	https://os-checker.github.io/docs/kern-crates/of/of/	
22	memory_addr	https://github.com/arceos-org/axmm_crates/tree/main/memory_addr	https://os-checker.github.io/docs/kern-crates/axmm_crates/memory_addr/	

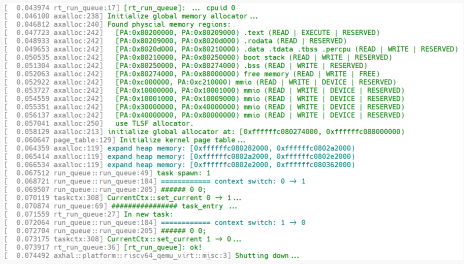
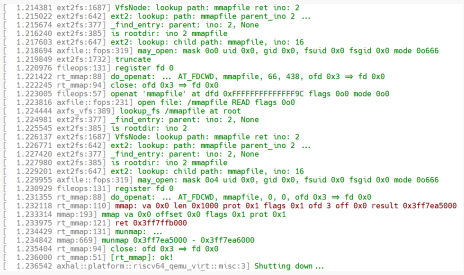
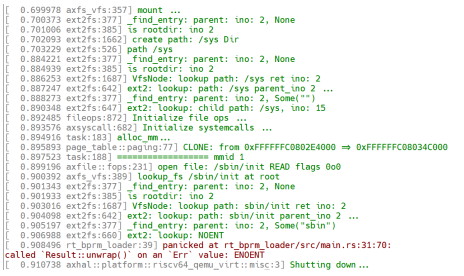
序号	已完成的组件	项目链接	文档	复现情况
23	cap_access	https://github.com/arceos-org/cap_access	https://os-checker.github.io/docs/kern-crates/cap_access/cap_access/	
24	flatten_objects	https://github.com/arceos-org/flatten_objects	https://docs.rs/flatten_objects/latest/flatten_objects/	 在arceos内运行测例
25	handler_table	https://github.com/arceos-org/handler_table	https://os-checker.github.io/docs/kern-crates/handler_table/handler_table/	
26	lazyinit	https://github.com/arceos-org/lazyinit	https://os-checker.github.io/docs/kern-crates/lazyinit/lazyinit/	

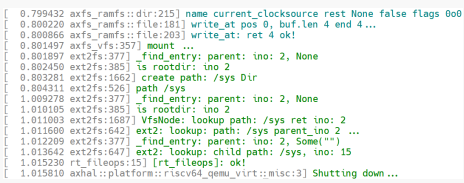
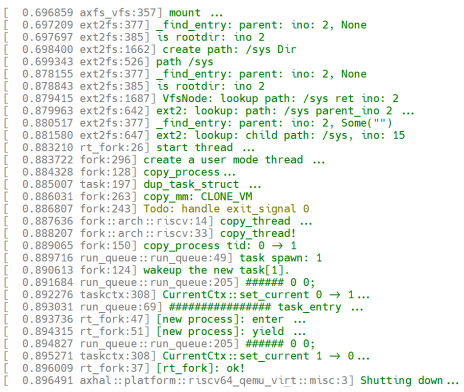
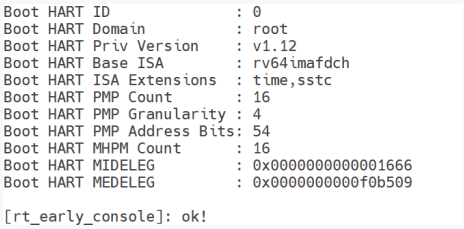
序号	已完成的组件	项目链接	文档	复现情况
27	page_table_multiarch	https://github.com/arceos-org/page_table_multiarch	https://docs.rs/page_table_multiarch/latest/page_table_multiarch/	 需要设一个环境变量，export RUSTFLAGS="--cfg doc" 然后cargo test
28	kernel_guard	https://github.com/arceos-org/kernel_guard	https://os-checker.github.io/docs/kern-crates/kernel_guard/kernel_guard/	
29	linked list	https://github.com/Starry-OS/linked_list	https://starry-os.github.io/linked_list/	
30	arch_boot	https://github.com/Starry-OS/arch_boot	https://starry-os.github.io/arch_boot/arch_boot/index.html	

序号	已完成的组件	项目链接	文档	复现情况
31	arm_gic	https://github.com/Starry-OS/arm_gic	https://starry-os.github.io/arm_gic/arm_gic/index.html	 直接在飞腾派上验证
32	axlog2	https://github.com/shilei-massclouds/axlog2	https://github.com/shilei-massclouds/axlog2/blob/main/README.md	需要在 `/root/.cargo/config` 中加入 [net] git-fetch-with-cli = true 
33	axhal	https://github.com/shilei-massclouds/axhal	https://github.com/shilei-massclouds/axhal/blob/main/README.md	
34	driver_block	https://github.com/shilei-massclouds/driver_block	https://github.com/shilei-massclouds/driver_block/blob/main/README.md	

序号	已完成的组件	项目链接	文档	复现情况
35	driver_virtio	https://github.com/shilei-massclouds/driver_virtio/blob/main/README.md	https://github.com/shilei-massclouds/driver_virtio/blob/main/README.md	
36	mutex	https://github.com/shilei-massclouds/mutex	https://github.com/shilei-massclouds/mutex/blob/main/README.md	
37	axalloc	https://github.com/shilei-massclouds/axalloc	https://github.com/shilei-massclouds/axalloc/blob/main/README.md	
38	page_table	https://github.com/shilei-massclouds/page_table	https://github.com/shilei-massclouds/page_table/blob/main/README.md	

序号	已完成的组件	项目链接	文档	复现情况
39	axfs_ramfs	https://github.com/shilei-massclouds/axfs_ramfs	https://github.com/shilei-massclouds/axfs_ramfs/blob/main/README.md	
40	mm	https://github.com/shilei-massclouds/mm/blob/main/README.md	https://github.com/shilei-massclouds/mm/blob/main/README.md	
41	fstree	https://github.com/shilei-massclouds/fstree	https://github.com/shilei-massclouds/fstree/blob/main/README.md	

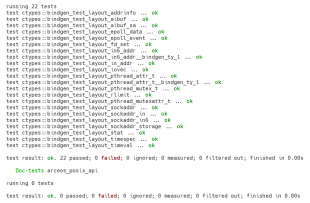
序号	已完成的组件	项目链接	文档	复现情况
42	run_queue	https://github.com/shilei-massclouds/run_queue/blob/main/README.md	https://github.com/shilei-massclouds/run_queue	
43	user_stack	https://github.com/shilei-massclouds/user_stack/blob/main/README.md	https://github.com/shilei-massclouds/user_stack	
44	mmap	https://github.com/shilei-massclouds/mmap/blob/main/README.md	https://github.com/shilei-massclouds/mmap	
45	bprm_loader	https://github.com/shilei-massclouds/bprm_loader/blob/main/README.md	https://github.com/shilei-massclouds/bprm_loader	

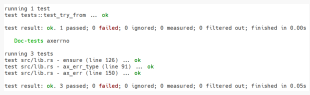
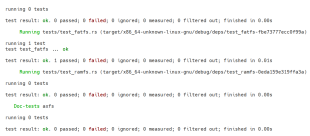
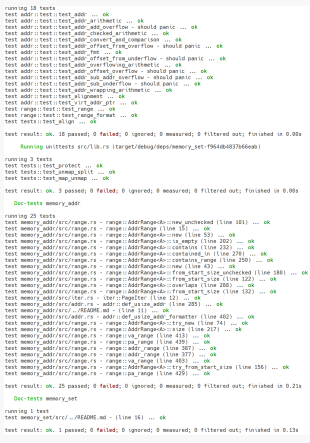
序号	已完成的组件	项目链接	文档	复现情况
46	fileops	https://github.com/shilei-massclouds/fileops	https://github.com/shilei-massclouds/fileops/blob/main/README.md	
47	fork	https://github.com/shilei-massclouds/fork	https://github.com/shilei-massclouds/fork/blob/main/README.md	
48	early_console	https://github.com/shilei-massclouds/early_console	https://github.com/shilei-massclouds/early_console/blob/main/README.md	
49	axmount	https://github.com/shilei-massclouds/axmount	https://github.com/shilei-massclouds/axmount/blob/main/README.md	

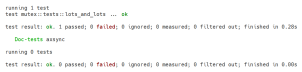
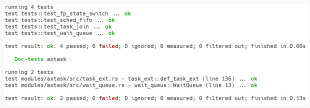
序号	已完成的组件	项目链接	文档	复现情况
50	axdtb	https://github.com/shilei-massclouds/axdtb	https://github.com/shilei-massclouds/axdtb/blob/main/README.md	

【1.2】 20个以上OS有关的功能组件 ✅

可复现组件数量：6个

序号	已完成的组件	相关OS	项目链接	文档链接	复现情况
1	arceos_posix_api	StarryOS/ArceOS	https://github.com/Starry-OS/arceos_posix_api	https://arceos.org/arceos/arceos_posix_api/index.html	

序号	已完成的组件	相关OS	项目链接	文档链接	复现情况
2	axerrno	ArceOS	https://github.com/arceos-org/axerrno	https://os-checker.github.io/docs/kern-crates/axerrno/axerrno/	
3	axfs	ArceOS	https://github.com/arceos-org/arceos/tree/main/modules/axfs	https://arceos.org/arceos/axfs/index.html	
4	axmm_crates	ArceOS	https://github.com/arceos-org/axmm_crates	https://arceos.org/axmm_crates/	

序号	已完成的组件	相关OS	项目链接	文档链接	复现情况
5	axsync	ArceOS	https://github.com/arceos-org/arceos/tree/main/modules/axsync	https://arceos.org/arceos/axsync/index.html	
6	axtask	ArceOS	https://github.com/arceos-org/arceos/tree/main/modules/axtask	https://arceos.org/arceos/axtask/index.html	

【1.3】 可在物理设备上运行的unikernel架构的OS内核主干 ✅

在飞腾派上运行 arceos unikernel

【操作文档】

仓库地址：https://github.com/tkf2019/arceos/tree/phytium-pi
文档：https://github.com/tkf2019/arceos/blob/phytium-pi/doc/platform_phytium_pi.md

【复现结果】

```
Phytium-Pi#go 0x90100000
## Starting application at 0x90100000 ...
aaaaaa
       d8888                            .d88888b.   .d8888b.
      d88888                           d88P" "Y88b d88P  Y88b
     d88P888                           888     888 Y88b.
    d88P 888 888d888  .d8888b  .d88b.  888     888  "Y888b.
   d88P  888 888P"   d88P"    d8P  Y8b 888     888     "Y88b.
  d88P   888 888     888      88888888 888     888       "888
 d8888888888 888     Y88b.    Y8b.     Y88b. .d88P Y88b  d88P
d88P     888 888      "Y8888P  "Y8888   "Y88888P"   "Y8888P"

arch = aarch64
platform = aarch64-phytium-pi
target = aarch64-unknown-none-softfloat
smp = 1
build_mode = release
log_level = trace

[115.000861 0 axruntime:130] Logging is enabled.
[115.006589 0 axruntime:131] Primary CPU 512 started, dtb = 0x1.
[115.013706 0 axruntime:133] Found physcial memory regions:
[115.020391 0 axruntime:135]   [PA:0x90100000, PA:0x90107000) .text (READ | EXECUTE | RESERVED)
[115.030198 0 axruntime:135]   [PA:0x90107000, PA:0x90109000) .rodata (READ | RESERVED)
[115.039312 0 axruntime:135]   [PA:0x90109000, PA:0x9010d000) .data .tdata .tbss .percpu (READ | WRITE | RESERVED)
[115.050770 0 axruntime:135]   [PA:0x9010d000, PA:0x9014d000) boot stack (READ | WRITE | RESERVED)
[115.060838 0 axruntime:135]   [PA:0x9014d000, PA:0x9014e000) .bss (READ | WRITE | RESERVED)
[115.070386 0 axruntime:135]   [PA:0x0, PA:0x1000) spintable (READ | WRITE | RESERVED)
[115.079414 0 axruntime:135]   [PA:0x9014e000, PA:0x9074e000) nocache memory (READ | WRITE | DEVICE)
[115.089656 0 axruntime:135]   [PA:0x9074e000, PA:0x100000000) free memory (READ | WRITE | FREE)
[115.099551 0 axruntime:135]   [PA:0x2800c000, PA:0x2800d000) mmio (READ | WRITE | DEVICE | RESERVED)
[115.109880 0 axruntime:135]   [PA:0x2800d000, PA:0x2800e000) mmio (READ | WRITE | DEVICE | RESERVED)
[115.120209 0 axruntime:135]   [PA:0x2800e000, PA:0x2800f000) mmio (READ | WRITE | DEVICE | RESERVED)
[115.130539 0 axruntime:135]   [PA:0x2800f000, PA:0x28010000) mmio (READ | WRITE | DEVICE | RESERVED)
[115.140868 0 axruntime:135]   [PA:0x30000000, PA:0x38000000) mmio (READ | WRITE | DEVICE | RESERVED)
[115.151197 0 axruntime:135]   [PA:0x40000000, PA:0x50000000) mmio (READ | WRITE | DEVICE | RESERVED)
[115.161526 0 axruntime:135]   [PA:0x28014000, PA:0x28016000) mmio (READ | WRITE | DEVICE | RESERVED)
[115.171856 0 axruntime:135]   [PA:0x28016000, PA:0x28018000) mmio (READ | WRITE | DEVICE | RESERVED)
[115.182185 0 axruntime:135]   [PA:0x28018000, PA:0x2801a000) mmio (READ | WRITE | DEVICE | RESERVED)
[115.192514 0 axruntime:135]   [PA:0x2801a000, PA:0x2801c000) mmio (READ | WRITE | DEVICE | RESERVED)
[115.202843 0 axruntime:135]   [PA:0x2801c000, PA:0x2801e000) mmio (READ | WRITE | DEVICE | RESERVED)
[115.213172 0 axruntime:135]   [PA:0x28034000, PA:0x28035000) mmio (READ | WRITE | DEVICE | RESERVED)
[115.223502 0 axruntime:135]   [PA:0x28035000, PA:0x28036000) mmio (READ | WRITE | DEVICE | RESERVED)
[115.233831 0 axruntime:135]   [PA:0x28036000, PA:0x28037000) mmio (READ | WRITE | DEVICE | RESERVED)
[115.244160 0 axruntime:135]   [PA:0x28037000, PA:0x28038000) mmio (READ | WRITE | DEVICE | RESERVED)
[115.254489 0 axruntime:135]   [PA:0x28038000, PA:0x28039000) mmio (READ | WRITE | DEVICE | RESERVED)
[115.264818 0 axruntime:135]   [PA:0x28039000, PA:0x2803a000) mmio (READ | WRITE | DEVICE | RESERVED)
[115.275148 0 axruntime:150] Initialize platform devices...
[115.281831 0 axruntime:186] Primary CPU 512 init OK.
Hello, world!
[115.289296 0 axruntime:199] main task exited: exit_code=0
[115.295893 0 axhal::platform::aarch64_phytium_pi::misc:22] Shutting down...
```

【注意事项】

1. 使用u盘启动，u盘格式化为fat32文件系统
2. 将加载内核文件的指令改为

```
fatload usb 0 0x90100000 helloworld_aarch64-phytium-pi.bin
```

【1.4】 可在物理设备上运行的monolithickernel架构的OS内核主干 ✅

在飞腾派上运行宏内核Starry

【操作文档】

https://github.com/Starry-OS/Starry/pull/41

【注意事项】

1. 在虚拟环境中安装pyserial和xmodem，pipx中没有xmodem，通过以下方式进入虚拟环境并安装

```
python3 -m venv myenv
source myenv/bin/activate
pip install pyserial xmodem
```

2. 代码需要调整，将根目录下的Cargo.toml里的 `tools/axlibc` 删除掉，否则编译失败（已反馈修改）
3. 编译过程中可能因本地环境缺少clang导致报错，可通过apt直接安装

```
error: failed to run custom build command for `taskctx v0.1.0 (https://github.com/Starry-OS/taskctx.git#d7d646ca)`

Caused by:
  process didn't exit successfully: `/code/Starry/target/release/build/taskctx-d3d326bddd789df7/build-script-build` (exit status: 101)
  --- stderr
  thread 'main' panicked at /root/.cargo/git/checkouts/taskctx-38ebba1102c121c2/d7d646c/build.rs:21:10:
  failed to execute clang: Os { code: 2, kind: NotFound, message: "No such file or directory" }
  note: run with `RUST_BACKTRACE=1` environment variable to display a backtrace
warning: build failed, waiting for other jobs to finish...
make: *** [scripts/make/build.mk:35: _cargo_build] Error 101
```

```
apt install clang
which clang
```

4. 需要在本机上编译，通过串口传递镜像，在飞腾派上运行

```
make A=apps/helloworld PLATFORM=aarch64-phytiumpi LOG=debug  chainboot
go 0x90100000
```

5. 如果飞腾派启动后直接进入了linux，需要再u-boot中先将自动引导关闭

```
setenv bootdelay -1
saveenv
```

6. 进入minicom之后无法输入

   https://blog.csdn.net/qq_38880380/article/details/78222695

【运行结果】

```
Phytium-Pi#go 0x90100000
## Starting application at 0x90100000 ...

       d8888                            .d88888b.   .d8888b.
      d88888                           d88P" "Y88b d88P  Y88b
     d88P888                           888     888 Y88b.
    d88P 888 888d888  .d8888b  .d88b.  888     888  "Y888b.
   d88P  888 888P"   d88P"    d8P  Y8b 888     888     "Y88b.
  d88P   888 888     888      88888888 888     888       "888
 d8888888888 888     Y88b.    Y8b.     Y88b. .d88P Y88b  d88P
d88P     888 888      "Y8888P  "Y8888   "Y88888P"   "Y8888P"

arch = aarch64
platform = aarch64-phytiumpi
target = aarch64-unknown-none-softfloat
smp = 1
build_mode = release
log_level = debug

[ 96.859078 0 axruntime:122] Logging is enabled.
[ 96.864806 0 axruntime:123] Primary CPU 512 started, dtb = 0x1.
[ 96.871924 0 axruntime:124] Platform name aarch64-phytiumpi.
[ 96.878781 0 axruntime:126] Found physcial memory regions:
[ 96.885465 0 axruntime:129]   [PA:0x90100000, PA:0x9010c000) .text (READ | EXECUTE | RESERVED)
[ 96.895273 0 axruntime:129]   [PA:0x9010c000, PA:0x9010f000) .rodata (READ | RESERVED)
[ 96.904387 0 axruntime:129]   [PA:0x9010f000, PA:0x90113000) .data .tdata .tbss .percpu (READ | WRITE | RESERVED)
[ 96.915845 0 axruntime:129]   [PA:0x90113000, PA:0x90153000) boot stack (READ | WRITE | RESERVED)
[ 96.925913 0 axruntime:129]   [PA:0x90153000, PA:0x90154000) .bss (READ | WRITE | RESERVED)
[ 96.935461 0 axruntime:129]   [PA:0x1000, PA:0x101000) fdt reserved (READ | RESERVED)
[ 96.944489 0 axruntime:129]   [PA:0x90154000, PA:0x100000000) free memory (READ | WRITE | FREE)
[ 96.954384 0 axruntime:129]   [PA:0x2800c000, PA:0x2800d000) mmio (READ | WRITE | DEVICE | RESERVED)
[ 96.964713 0 axruntime:129]   [PA:0x2800d000, PA:0x2800e000) mmio (READ | WRITE | DEVICE | RESERVED)
[ 96.975042 0 axruntime:129]   [PA:0x2800e000, PA:0x2800f000) mmio (READ | WRITE | DEVICE | RESERVED)
[ 96.985371 0 axruntime:129]   [PA:0x2800f000, PA:0x28010000) mmio (READ | WRITE | DEVICE | RESERVED)
[ 96.995701 0 axruntime:129]   [PA:0x30000000, PA:0x38000000) mmio (READ | WRITE | DEVICE | RESERVED)
[ 97.006030 0 axruntime:129]   [PA:0x40000000, PA:0x80000000) mmio (READ | WRITE | DEVICE | RESERVED)
[ 97.016359 0 axruntime:129]   [PA:0x1000000000, PA:0x3000000000) mmio (READ | WRITE | DEVICE | RESERVED)
[ 97.027035 0 axruntime:148] Initialize platform devices...
[ 97.033719 0 axruntime:187] Primary CPU 512 init OK.
Hello, world!
[ 97.041184 0 axruntime:197] main task exited: exit_code=0
[ 97.047780 0 axhal::platform::aarch64_phytiumpi::misc:3] Shutting down...
```

【1.5】 国产处理器对OS编译环境的支持 ✅

在飞腾派上编译arceos

【操作文档】

安装rust环境：https://chenlongos.com/Phytium-Car/setup2.html

编译：https://chenlongos.com/Phytium-Car/setup5.html

【编译】

编译出二进制文件

1. x86_64

```
root@phytiumpi:/arceos# make ARCH=x86_64 A=examples/helloworld
    Building App: helloworld, Arch: x86_64, Platform: x86_64-qemu-q35, App type: rust
cargo -C examples/helloworld build -Z unstable-options --target x86_64-unknown-none --target-dir /arceos/target --release
--features "axstd/log-level-warn"
warning: `/root/.cargo/config` is deprecated in favor of `config.toml`
note: if you need to support cargo 1.38 or earlier, you can symlink `config` to `config.toml`
warning: `/root/.cargo/config` is deprecated in favor of `config.toml`
note: if you need to support cargo 1.38 or earlier, you can symlink `config` to `config.toml`
    Finished `release` profile [optimized] target(s) in 0.28s
rust-objcopy --binary-architecture=x86_64 examples/helloworld/helloworld_x86_64-qemu-q35.elf --strip-all -O binary examples
/helloworld/helloworld_x86_64-qemu-q35.bin
```

2. aarch64

```
root@phytiumpi:/arceos# make ARCH=aarch64 A=examples/helloworld
    Building App: helloworld, Arch: aarch64, Platform: aarch64-qemu-virt, App type: rust
cargo -C examples/helloworld build -Z unstable-options --target aarch64-unknown-none-softfloat --target-dir /arceos/target
--release  --features "axstd/log-level-warn"
warning: `/root/.cargo/config` is deprecated in favor of `config.toml`
note: if you need to support cargo 1.38 or earlier, you can symlink `config` to `config.toml`
warning: `/root/.cargo/config` is deprecated in favor of `config.toml`
note: if you need to support cargo 1.38 or earlier, you can symlink `config` to `config.toml`
   Compiling axconfig v0.1.0 (/arceos/modules/axconfig)
   Compiling axhal v0.1.0 (/arceos/modules/axhal)
   Compiling axruntime v0.1.0 (/arceos/modules/axruntime)
   Compiling axtask v0.1.0 (/arceos/modules/axtask)
   Compiling axsync v0.1.0 (/arceos/modules/axsync)
   Compiling axfeat v0.1.0 (/arceos/api/axfeat)
   Compiling arceos_api v0.1.0 (/arceos/api/arceos_api)
   Compiling axstd v0.1.0 (/arceos/ulib/axstd)
   Compiling arceos-helloworld v0.1.0 (/arceos/examples/helloworld)
    Finished `release` profile [optimized] target(s) in 6.34s
rust-objcopy --binary-architecture=aarch64 examples/helloworld/helloworld_aarch64-qemu-virt.elf --strip-all -O binary examp
les/helloworld/helloworld_aarch64-qemu-virt.bin
```

3. riscv64

```
root@phytiumpi:/arceos# make ARCH=riscv64 A=examples/helloworld
    Building App: helloworld, Arch: riscv64, Platform: riscv64-qemu-virt, App type: rust
cargo -C examples/helloworld build -Z unstable-options --target riscv64gc-unknown-none-elf --target-dir /arceos/target --re
lease  --features "axstd/log-level-warn"
warning: `/root/.cargo/config` is deprecated in favor of `config.toml`
note: if you need to support cargo 1.38 or earlier, you can symlink `config` to `config.toml`
warning: `/root/.cargo/config` is deprecated in favor of `config.toml`
note: if you need to support cargo 1.38 or earlier, you can symlink `config` to `config.toml`
   Compiling axconfig v0.1.0 (/arceos/modules/axconfig)
   Compiling axhal v0.1.0 (/arceos/modules/axhal)
   Compiling axruntime v0.1.0 (/arceos/modules/axruntime)
   Compiling axtask v0.1.0 (/arceos/modules/axtask)
   Compiling axsync v0.1.0 (/arceos/modules/axsync)
   Compiling axfeat v0.1.0 (/arceos/api/axfeat)
   Compiling arceos_api v0.1.0 (/arceos/api/arceos_api)
   Compiling axstd v0.1.0 (/arceos/ulib/axstd)
   Compiling arceos-helloworld v0.1.0 (/arceos/examples/helloworld)
    Finished `release` profile [optimized] target(s) in 5.72s
rust-objcopy --binary-architecture=riscv64 examples/helloworld/helloworld_riscv64-qemu-virt.elf --strip-all -O binary examp
les/helloworld/helloworld_riscv64-qemu-virt.bin
```

【2.1】 在x86硬件平台上支持运行redis，相对Linux下，性能提升10%以上，避免Linux中的常见内存缺陷 ✅

【操作文档】https://github.com/arceos-org/arceos-apps/tree/main/c/redis

【代码仓库】https://github.com/ZR233/arceos/tree/pcie_dev

【注意事项】

在linux中查看网卡地址

```
lspci -vv
```

```
00:1f.6 Ethernet controller: Intel Corporation Device 0dc8 (rev 11)
        DeviceName: Onboard - Ethernet
        Subsystem: Intel Corporation Device 0000
        Control: I/O- Mem+ BusMaster+ SpecCycle- MemWINV- VGASnoop- ParErr- Stepping- SERR- FastB2B- DisINTx+
        Status: Cap+ 66MHz- UDF- FastB2B- ParErr- DEVSEL=fast >TAbort- <TAbort- <MAbort- >SERR- <PERR- INTx-
        Latency: 0
        Interrupt: pin D routed to IRQ 127
        Region 0: Memory at 80700000 (32-bit, non-prefetchable) [size=128K]
        Capabilities: [c8] Power Management version 3
                Flags: PMEClk- DSI+ D1- D2- AuxCurrent=0mA PME(D0+,D1-,D2-,D3hot+,D3cold+)
                Status: D0 NoSoftRst+ PME-Enable- DSel=0 DScale=1 PME-
        Capabilities: [d0] MSI: Enable+ Count=1/1 Maskable- 64bit+
                Address: 00000000fee00318  Data: 0000
        Kernel driver in use: e1000e
        Kernel modules: e1000e
```

并修改配置文件x86_64-pc-oslab.toml

```
mmio-regions = [
    ["0xfec0_0000", "0x1000"], # IO APIC
    ["0xfed0_0000", "0x1000"], # HPET
    ["0xfee0_0000", "0x1000"], # Local APIC
    ["0xc000_0000", "0x0100_0000"], # PCI config space
    ["0xfcd8_0000", "0x0008_0000"], # Ixgbe BAR0
    ["0x8070_0000", "0x2_0000"],    # e1000 driver
]
```

https://github.com/ZR233/arceos/commit/9110b6ec5dec364174ca5036ead8f5287dbb7881

【运行结果】

arceos编译redis指令

```
make A=c/redis LOG=error PLATFORM=x86_64-pc-oslab SMP=4 NET=y BLK=y
FEATURES=driver-e1000,driver-ixgbe,driver-ramdisk IP=10.3.10.81 GW=10.3.10.1
```

将编译好的文件放在x86工控机上，通过multiboot方式运行成功

```
Initialize IDT & GDT...
Got TSC frequency by CPUID: 2500 MHz

       d8888                            .d88888b.   .d8888b.
      d88888                           d88P" "Y88b d88P  Y88b
     d88P888                           888     888 Y88b.
    d88P 888 888d888  .d8888b  .d88b.  888     888  "Y888b.
   d88P  888 888P"   d88P"    d8P  Y8b 888     888     "Y88b.
  d88P   888 888     888      88888888 888     888       "888
 d8888888888 888     Y88b.    Y8b.     Y88b. .d88P Y88b  d88P
d88P     888 888      "Y8888P  "Y8888   "Y88888P"   "Y8888P"

arch = x86_64
platform = x86_64-pc-oslab
target = x86_64-unknown-none
smp = 4
build_mode = release
log_level = error

[  0.243917 fatfs::boot_sector:623] Invalid FAT type (expect Fat16 due to 16253 clusters
[  0.253982 fatfs::dir:139] Is a directory
[  0.259686 fatfs::dir:139] Is a directory
[  0.265410 fatfs::dir:139] Is a directory
[  0.271135 fatfs::dir:139] Is a directory
Run Redis with: ./redis-server --bind 0.0.0.0 --port 5555 --save "" --appendonly no --protected-mode no
2:C 01 Jan 1970 00:00:00.322 # oO0OoO0OoO0Oo Redis is starting oO0OoO0OoO0Oo
2:C 01 Jan 1970 00:00:00.329 # Redis version=7.0.12, bits=64, commit=00000000, modified=1, pid=2, just started
2:C 01 Jan 1970 00:00:00.340 # Configuration loaded
2:M 01 Jan 1970 00:00:00.345 * Increased maximum number of open files to 10032 (it was originally set to 1024).
2:M 01 Jan 1970 00:00:00.356 * monotonic clock: POSIX clock_gettime
2:M 01 Jan 1970 00:00:00.363 * Running mode=standalone, port=5555.
2:M 01 Jan 1970 00:00:00.369 # Server initialized
2:M 01 Jan 1970 00:00:00.374 # WARNING Memory overcommit must be enabled! Without it, a background save or replication may
fail under low memory condition. To fix this issue add 'vm.overcommit_memory = 1' to /etc/sysctl.conf and then reboot or ru
n the command 'sysctl vm.overcommit_memory=1' for this to take effect.
2:M 01 Jan 1970 00:00:00.404 * Ready to accept connections
```

【性能测试】

测试指令

```
redis-benchmark -h 10.3.10.81 -p 6666 -q -t
PING,SET,GET,INCR,LPUSH,RPUSH,LPOP,RPOP,SADD,HSET,SPOP,ZADD,ZPOPMIN
```

分别测试同时发送给1，5，10，20个客户端

对比`PING,SET,GET,INCR,LPUSH,RPUSH,LPOP,RPOP,SADD,HSET,SPOP,ZADD,ZPOPMIN`每秒钟的请求数量

【Linux】

服务端 `IP：10.3.10.81  Port：6666`

客户端 `IP：10.3.10.62  Port：6666`

```
root@josen:~# redis-benchmark -h 10.3.10.81 -p 6666 -c 1 -q -t PING,SET,GET,INCR,LPUSH,RPUSH,LPOP,RPOP,SADD,HSET,ZPOPMIN
PING_INLINE: 7865.35 requests per second, p50=0.127 msec
PING_MBULK: 7873.40 requests per second, p50=0.127 msec
SET: 7861.64 requests per second, p50=0.127 msec
GET: 7870.92 requests per second, p50=0.127 msec
INCR: 7852.99 requests per second, p50=0.127 msec
LPUSH: 7861.64 requests per second, p50=0.127 msec
RPUSH: 7849.91 requests per second, p50=0.127 msec
LPOP: 7848.68 requests per second, p50=0.127 msec
RPOP: 7845.60 requests per second, p50=0.127 msec
SADD: 7860.40 requests per second, p50=0.127 msec
HSET: 7830.85 requests per second, p50=0.127 msec
ZPOPMIN: 7846.83 requests per second, p50=0.127 msec

root@josen:~# redis-benchmark -h 10.3.10.81 -p 6666 -c 5 -q -t PING,SET,GET,INCR,LPUSH,RPUSH,LPOP,RPOP,SADD,HSET,ZPOPMIN
PING_INLINE: 39904.23 requests per second, p50=0.127 msec
PING_MBULK: 40241.45 requests per second, p50=0.127 msec
SET: 40209.09 requests per second, p50=0.127 msec
GET: 39856.52 requests per second, p50=0.127 msec
INCR: 39936.10 requests per second, p50=0.127 msec
LPUSH: 40387.72 requests per second, p50=0.127 msec
RPUSH: 40371.42 requests per second, p50=0.127 msec
LPOP: 40436.71 requests per second, p50=0.127 msec
RPOP: 40048.06 requests per second, p50=0.127 msec
SADD: 40032.03 requests per second, p50=0.127 msec
HSET: 40016.00 requests per second, p50=0.127 msec
ZPOPMIN: 40032.03 requests per second, p50=0.127 msec

root@josen:~# redis-benchmark -h 10.3.10.81 -p 6666 -c 10 -q -t PING,SET,GET,INCR,LPUSH,RPUSH,LPOP,RPOP,SADD,HSET,ZPOPMIN
PING_INLINE: 72254.34 requests per second, p50=0.135 msec
PING_MBULK: 75700.23 requests per second, p50=0.119 msec
SET: 77399.38 requests per second, p50=0.119 msec
GET: 77220.08 requests per second, p50=0.119 msec
INCR: 74404.77 requests per second, p50=0.127 msec
LPUSH: 77579.52 requests per second, p50=0.119 msec
RPUSH: 79808.46 requests per second, p50=0.119 msec
LPOP: 74239.05 requests per second, p50=0.127 msec
RPOP: 72306.58 requests per second, p50=0.135 msec
SADD: 73046.02 requests per second, p50=0.127 msec
HSET: 77459.34 requests per second, p50=0.119 msec
ZPOPMIN: 79554.50 requests per second, p50=0.119 msec

root@josen:~# redis-benchmark -h 10.3.10.81 -p 6666 -c 20 -q -t PING,SET,GET,INCR,LPUSH,RPUSH,LPOP,RPOP,SADD,HSET,ZPOPMIN
PING_INLINE: 132100.39 requests per second, p50=0.135 msec
PING_MBULK: 131233.59 requests per second, p50=0.135 msec
SET: 130378.09 requests per second, p50=0.135 msec
GET: 131061.59 requests per second, p50=0.135 msec
INCR: 131578.95 requests per second, p50=0.135 msec
LPUSH: 136425.66 requests per second, p50=0.127 msec
RPUSH: 136798.91 requests per second, p50=0.127 msec
LPOP: 137174.22 requests per second, p50=0.127 msec
RPOP: 138312.59 requests per second, p50=0.127 msec
SADD: 130548.30 requests per second, p50=0.135 msec
HSET: 132626.00 requests per second, p50=0.135 msec
ZPOPMIN: 132450.33 requests per second, p50=0.135 msec
```

【Arceos】

服务端 `IP：10.3.10.81  Port：5555`

客户端 `IP：10.3.10.62  Port：5555`

```
root@josen:~# redis-benchmark -h 10.3.10.81 -p 5555 -c 1 -q -t PING,SET,GET,INCR,LPUSH,RPUSH,LPOP,RPOP,SADD,HSET,ZPOPMIN
PING_INLINE: 13835.09 requests per second, p50=0.071 msec
PING_MBULK: 14033.12 requests per second, p50=0.071 msec
SET: 13906.27 requests per second, p50=0.071 msec
GET: 13931.46 requests per second, p50=0.071 msec
INCR: 13798.81 requests per second, p50=0.071 msec
LPUSH: 13730.61 requests per second, p50=0.079 msec
RPUSH: 13879.25 requests per second, p50=0.071 msec
LPOP: 13964.53 requests per second, p50=0.071 msec
RPOP: 13986.01 requests per second, p50=0.071 msec
SADD: 13700.51 requests per second, p50=0.079 msec
HSET: 13740.04 requests per second, p50=0.079 msec
ZPOPMIN: 13982.10 requests per second, p50=0.071 msec

root@josen:~# redis-benchmark -h 10.3.10.81 -p 5555 -c 5 -q -t PING,SET,GET,INCR,LPUSH,RPUSH,LPOP,RPOP,SADD,HSET,ZPOPMIN
PING_INLINE: 62932.66 requests per second, p50=0.071 msec
PING_MBULK: 63291.14 requests per second, p50=0.071 msec
SET: 62460.96 requests per second, p50=0.071 msec
GET: 63011.97 requests per second, p50=0.071 msec
INCR: 62500.00 requests per second, p50=0.071 msec
LPUSH: 60240.96 requests per second, p50=0.079 msec
RPUSH: 62853.55 requests per second, p50=0.071 msec
LPOP: 62932.66 requests per second, p50=0.071 msec
RPOP: 62893.08 requests per second, p50=0.071 msec
SADD: 62578.22 requests per second, p50=0.071 msec
HSET: 61957.87 requests per second, p50=0.079 msec
ZPOPMIN: 62774.64 requests per second, p50=0.071 msec

root@josen:~# redis-benchmark -h 10.3.10.81 -p 5555 -c 10 -q -t PING,SET,GET,INCR,LPUSH,RPUSH,LPOP,RPOP,SADD,HSET,ZPOPMIN
PING_INLINE: 105820.11 requests per second, p50=0.087 msec
PING_MBULK: 105263.16 requests per second, p50=0.087 msec
SET: 104712.05 requests per second, p50=0.087 msec
GET: 105263.16 requests per second, p50=0.087 msec
INCR: 105263.16 requests per second, p50=0.087 msec
LPUSH: 100000.00 requests per second, p50=0.095 msec
RPUSH: 104712.05 requests per second, p50=0.087 msec
LPOP: 104602.52 requests per second, p50=0.087 msec
RPOP: 104931.80 requests per second, p50=0.087 msec
SADD: 105042.02 requests per second, p50=0.087 msec
HSET: 103734.44 requests per second, p50=0.087 msec
ZPOPMIN: 105374.08 requests per second, p50=0.087 msec

root@josen:~# redis-benchmark -h 10.3.10.81 -p 5555 -c 20 -q -t PING,SET,GET,INCR,LPUSH,RPUSH,LPOP,RPOP,SADD,HSET,ZPOPMIN
PING_INLINE: 160000.00 requests per second, p50=0.103 msec
PING_MBULK: 165562.92 requests per second, p50=0.103 msec
SET: 161550.89 requests per second, p50=0.103 msec
GET: 162601.62 requests per second, p50=0.103 msec
INCR: 157480.31 requests per second, p50=0.103 msec
LPUSH: 157977.88 requests per second, p50=0.111 msec
RPUSH: 164744.64 requests per second, p50=0.103 msec
LPOP: 176678.45 requests per second, p50=0.095 msec
RPOP: 171821.30 requests per second, p50=0.095 msec
SADD: 171526.58 requests per second, p50=0.103 msec
HSET: 162074.56 requests per second, p50=0.103 msec
ZPOPMIN: 167785.23 requests per second, p50=0.103 msec
```

【结论】对比redis运行在arceos和linux系统中的请求相应速度，在客户端数量为1，5，10，20时，性能提升均大于10%

性能指标	客户端数量			客户端数量			客户端数量			客户端数量		
	1			5			10			20		
	linux	arceos	提升	linux	arceos	提升	linux	arceos	提升	linux	arceos	提升
PING_INLIN	7865.35	13835.09	76.20%	39904.2	62932.66	57.79%	72254.3	105820.11	46.67%	132100	160000	21.12%
PING_MBULK	7873.4	14033.12	78.58%	40241.5	63291.14	57.33%	75700.2	105263.16	39.88%	131234	165563	26.07%
SET	7861.64	13906.27	77.70%	40209.1	62460.96	55.89%	77399.4	104712.05	35.25%	130378	161551	24.03%
GET	7870.92	13931.46	77.69%	39856.5	63011.97	58.08%	77220.1	105263.16	36.58%	131062	162602	24.14%
INCR	7852.99	13798.81	75.98%	39936.1	62500.00	56.71%	74404.8	105263.16	41.83%	131579	157480	19.63%
LPUSH	7861.64	13730.61	74.27%	40387.7	60240.96	49.28%	77579.5	100000.00	28.94%	136426	157978	15.74%
RPUSH	7849.91	13879.25	76.61%	40371.4	62853.55	55.66%	79808.5	104712.05	35.24%	136799	164745	20.46%
LPOP	7848.68	13964.53	77.95%	40436.7	62932.66	55.66%	74239.1	104602.52	40.81%	137174	176678	28.88%
RPOP	7845.6	13986.01	78.49%	40048.1	62893.08	57.18%	72306.6	104931.8	45.69%	138313	171821	24.04%
SADD	7860.4	13700.51	74.68%	40032	62578.22	56.98%	73046	105042.02	43.55%	130548	171527	31.45%
HSET	7830.85	13740.04	75.80%	40016	61957.87	54.40%	77459.3	103734.44	33.03%	132626	162075	22.23%
ZPOPMIN	7846.83	13982.10	78.66%	40032	62774.64	56.83%	79554.5	105374.08	32.56%	132450	167785	26.69%

【2.2】 在x86硬件平台上支持运行tokio，相对Linux下，性能提升10%以上，避免Linux中的常见内存缺陷 ✅

【操作文档】：https://docs.qq.com/doc/DYWJKQWl4RWxmZUtQ?is_no_hook_redirect=1

【代码仓库】：https://github.com/Gallium70/arceos/tree/dora-wip

【注意事项】

1. monoio-benchmark暂未迁移到arceos中，先不使用原仓库的arceos，拉取下面仓库进行编译服务端

   https://github.com/Gallium70/arceos/tree/dora-wip

2. 编译客户端时，客户端的 main 中可能需要加上 `#![feature(impl_trait_in_assoc_type)]`，并将 Cargo.toml 中 monoio 的依赖版本改为 0.2。

【qemu运行】

```
make A=apps/std/monoio-benchmark/tokio-server SMP=1 NET=y NET_DEV=user LOG=info
STD=y run
```

服务端

```
       d8888                            .d88888b.   .d8888b.
      d88888                           d88P" "Y88b d88P  Y88b
     d88P888                           888     888 Y88b.
    d88P 888 888d888  .d8888b  .d88b.  888     888  "Y888b.
   d88P  888 888P"   d88P"    d8P  Y8b 888     888     "Y88b.
  d88P   888 888     888      88888888 888     888       "888
 d8888888888 888     Y88b.    Y8b.     Y88b. .d88P Y88b  d88P
d88P     888 888      "Y8888P  "Y8888   "Y88888P"   "Y8888P"

arch = x86_64
platform = x86_64-qemu-q35
target = x86_64-unknown-arceos
smp = 1
build_mode = release
log_level = info

[1730965490.005016 0 axruntime:129] Logging is enabled.
[1730965490.005576 0 axruntime:130] Primary CPU 0 started, dtb = 0x0.
[1730965490.006274 0 axruntime:132] Found physcial memory regions:
[1730965490.006954 0 axruntime:134]   [PA:0x200000, PA:0x2a8000) .text (READ | EXECUTE | RESERVED)
[1730965490.007910 0 axruntime:134]   [PA:0x2a8000, PA:0x2b6000) .rodata (READ | RESERVED)
[1730965490.008764 0 axruntime:134]   [PA:0x2b6000, PA:0x2c2000) .data .tdata .tbss .percpu (READ | WRITE | RESERVED)
[1730965490.009838 0 axruntime:134]   [PA:0x2c2000, PA:0x302000) boot stack (READ | WRITE | RESERVED)
[1730965490.010831 0 axruntime:134]   [PA:0x302000, PA:0x328000) .bss (READ | WRITE | RESERVED)
[1730965490.011744 0 axruntime:134]   [PA:0x1000, PA:0x9f000) low memory (READ | WRITE | RESERVED)
[1730965490.012638 0 axruntime:134]   [PA:0x328000, PA:0x8000000) free memory (READ | WRITE | FREE)
[1730965490.013571 0 axruntime:134]   [PA:0xb0000000, PA:0xc0000000) mmio (READ | WRITE | DEVICE | RESERVED)
[1730965490.014609 0 axruntime:134]   [PA:0xfe000000, PA:0xfec00000) mmio (READ | WRITE | DEVICE | RESERVED)
[1730965490.015506 0 axruntime:134]   [PA:0xfec00000, PA:0xfec01000) mmio (READ | WRITE | DEVICE | RESERVED)
[1730965490.016086 0 axruntime:134]   [PA:0xfed00000, PA:0xfed01000) mmio (READ | WRITE | DEVICE | RESERVED)
[1730965490.016670 0 axruntime:134]   [PA:0xfee00000, PA:0xfee01000) mmio (READ | WRITE | DEVICE | RESERVED)
[1730965490.017253 0 axruntime:240] Initialize global memory allocator...
[1730965490.017676 0 axruntime:241]   use TLSF allocator.
[1730965490.018060 0 axruntime:148] Initialize kernel page table...
[1730965490.018465 0 axruntime:152] Initialize platform devices...
[1730965490.018846 0 axhal::platform::x86_pc::apic:87] Initialize Local APIC...
[1730965490.019276 0 axhal::platform::x86_pc::apic:102] Using x2APIC.
[1730965490.019699 0 axhal::platform::x86_pc::apic:116] Initialize IO APIC...
[1730965490.020123 0 axtask::api:66] Initialize scheduling...
[1730965490.020493 0 axtask::api:72]   use FIFO scheduler.
[1730965490.020837 0 axdriver:152] Initialize device drivers...
[1730965490.021201 0 axdriver:153]   device model: static
[1730965490.026562 0 virtio_drivers::device::net::dev_raw:31] Device features CTRL_GUEST_OFFLOADS | MAC | MRG_RXBUF | STATUS
[1730965490.028362 0 axdriver::bus::pci:104] registered a new Net device at 00:02.0: "virtio-net"
[1730965490.152997 0 axnet:44] Initialize network subsystem...
[1730965490.153544 0 axnet:47]   use NIC 0: "virtio-net"
[1730965490.154382 0 axnet::smoltcp_impl:326] created net interface "eth0":
[1730965490.155047 0 axnet::smoltcp_impl:327]   ether:    52-54-00-12-34-56
[1730965490.155725 0 axnet::smoltcp_impl:328]   ip:       10.0.2.15/24
[1730965490.156374 0 axnet::smoltcp_impl:329]   gateway:  10.0.2.2
[1730965490.156984 0 axruntime:188] Primary CPU 0 init OK.
[1730965490.157550 0:2 axruntime:222] Start std::runtime_entry ...
Running ping pong server with Tokio.
Packet size: 1024
Listen 0.0.0.0:5555
CPU count: 1
```

客户端

```
cargo build --release
./target/release/client --target 127.0.0.1:5555 --conns-per-core 150
```

```
root@josen:/code/Acceptance_check/2.2/monoio-benchmark# ./target/release/client --target 127.0.0.1:5555 --conns-per-core 150
Running ping pong client.
Packet size: 1024
Connection count per core: 150; Global connection count: 150
QPS limit per core: 0; Global QPS limit: 0
Target: 127.0.0.1:5555
CPU slot: 0
1.000: NAdd: 156738; NSum: 156738; NAverage: 156722.312, LatencyAverage: 206.191 us
2.000: NAdd: 164209; NSum: 320947; NAverage: 160457.234, LatencyAverage: 252.058 us
3.000: NAdd: 159862; NSum: 480809; NAverage: 160253.297, LatencyAverage: 285.646 us
4.000: NAdd: 147456; NSum: 628265; NAverage: 157050.297, LatencyAverage: 304.432 us
5.001: NAdd: 156104; NSum: 784369; NAverage: 156857.875, LatencyAverage: 327.296 us
6.001: NAdd: 163048; NSum: 947417; NAverage: 157886.672, LatencyAverage: 331.045 us
7.001: NAdd: 158865; NSum: 1106282; NAverage: 158024.016, LatencyAverage: 334.953 us
8.001: NAdd: 155939; NSum: 1262221; NAverage: 157761.000, LatencyAverage: 354.932 us
9.001: NAdd: 168159; NSum: 1430380; NAverage: 158914.391, LatencyAverage: 355.087 us
10.001: NAdd: 167706; NSum: 1598086; NAverage: 159791.844, LatencyAverage: 355.314 us
11.001: NAdd: 167310; NSum: 1765396; NAverage: 160473.750, LatencyAverage: 355.580 us
12.001: NAdd: 184946; NSum: 1950342; NAverage: 162511.562, LatencyAverage: 355.648 us
13.001: NAdd: 168126; NSum: 2118468; NAverage: 162942.125, LatencyAverage: 355.706 us
14.001: NAdd: 167193; NSum: 2285661; NAverage: 163244.547, LatencyAverage: 355.898 us
15.002: NAdd: 167769; NSum: 2453430; NAverage: 163545.031, LatencyAverage: 367.345 us
16.002: NAdd: 168336; NSum: 2621766; NAverage: 163843.406, LatencyAverage: 367.377 us
17.002: NAdd: 168836; NSum: 2790602; NAverage: 164136.078, LatencyAverage: 367.334 us
```

【物理机运行】

1. 拉取代码

```
git clone https://github.com/Gallium70/arceos.git
cd arceos
git checkout dora-wip
# 这一步会拉取 rust 源码，可能耗时较长
git submodule update --init --recursive
```

2. 编辑 arceos/platforms/x86_64-pc-oslab.toml 文件内容,将 mmio-regions 中 PCI config space 和 Ixgbe BAR0 对应的起始地址替换为从 /proc/iomem 查询到的地址；

```
cat /proc/iomem
```

pci-ecam-base 同样替换为 PCI config space 的起始地址。

```
# Architecture identifier.
arch = "x86_64"
# Platform identifier.
platform = "x86_64-pc-oslab"
# Platform family.
family = "x86-pc"

# Base address of the whole physical memory.
phys-memory-base = "0"
# Size of the whole physical memory.
phys-memory-size = "0x8000_0000"    # 2G
# Base physical address of the kernel image.
kernel-base-paddr = "0x20_0000"
# Base virtual address of the kernel image.
kernel-base-vaddr = "0xffff_ff80_0020_0000"
# Linear mapping offset, for quick conversions between physical and virtual
# addresses.
phys-virt-offset = "0xffff_ff80_0000_0000"
# MMIO regions with format (`base_paddr`, `size`).
mmio-regions = [
    ["0xfec0_0000", "0x1000"],      # IO APIC
    ["0xfed0_0000", "0x1000"],      # HPET
    ["0xfee0_0000", "0x1000"],      # Local APIC
```

```
    ["0xc000_0000", "0x0100_0000"], # PCI config space
    ["0x8068_0000", "0x10_0000"],   #intel 82599
]
# VirtIO MMIO regions with format (`base_paddr`, `size`).
virtio-mmio-regions = []
# Base physical address of the PCIe ECAM space (should read from ACPI 'MCFG'
table).
pci-ecam-base = "0xc000_0000"
# End PCI bus number.
pci-bus-end = "0x05"
# PCI device memory ranges (not used on x86).
pci-ranges = [
    ["0x3ef_f0000", "0x1_0000"],            # PIO space
    ["0x1000_0000", "0x2eff_0000"],         # 32-bit MMIO space
    ["0x80_0000_0000", "0x80_0000_0000"],   # 64-but MMIO space
]

# Timer interrupt frequencyin Hz.
timer-frequency = "4_000_000_000"   # 4.0GHz
```

3. 编译内核文件

```
make A=apps/std/monoio-benchmark/tokio-server PLATFORM=x86_64-pc-oslab STD=y
SMP=1 NET=y FEATURES=driver-ixgbe,bus-pci,net IP=10.2.2.2 GW=10.2.2.1
LOG=info
```

将编译的镜像复制到物理机的 /boot 路径下

```
cp arceos/apps/std/monoio-benchmark/tokio-server/tokio-server_x86_64-pc-
oslab.bin /boot
```

4. 客户端机器安装万兆网卡，与arceos 物理机直连，配置静态 ip 与arceos 处于同一子网（如10.3.10.62/24）,ubuntu可以编辑 `/etc/netplan/<file_name>` 配置网卡ip

服务端ip配置：

```
network:
  version: 2
  ethernets:
    eno1:
      dhcp4: true
    enp1s0:
      dhcp4: false
      addresses:
        - 10.3.10.81/24
```

客户端ip配置：

```
network:
    ethernets:
        enp3s0:
            dhcp4: true
        enp1s0:
            dhcp4: false
            addresses:
              - 10.3.10.62/24
    version: 2
```

5. 配置grub启动信息，在/boot/grub/grub.cfg中添加对应的multiboot的选项

```
menuentry "ArceOS Tokio" {
    set root='hd0,gpt2'
    multiboot /tokio-server_x86_64-pc-oslab.bin
    boot
}
```

   重启物理机，在 grub 菜单中选择 ArceOS Tokio启动，在串口连接的机器中显示tokio-server的执行信息

6. 拉取测试脚本并运行客户端进行测试（注意修改测试脚本中的ip和每核连接数范围）

```
git clone https://github.com/Gallium70/monoio-benchmark.git
cd monoio-benchmark
./script/run-client.sh <结果名称> <客户端核数>
```

【注意事项】

1. 工控机PCI的82599基地址显示为 `0x80a0_0000`，但在arceos中实际加载的地址为 `0x8068_0000`，需要在配置文件的 `mmio-regions` 中将intel_82599的基地址改为 `0x8068_0000`
2. linux测试的服务端程序位于 `monoio-benchmark/target/debug/tokio-server`
3. `arceos/modules/axnet/src/smoltcp_impl/mod.rs` 这个里面的 `LISTEN_QUEUE_SIZE` 需要改大一点，不然连接数高的时候会掉线，可以改为4096

【测试结果】

```
Initialize IDT & GDT...
Got TSC frequency by CPUID: 2500 MHz

       d8888                            .d88888b.   .d8888b.
      d88888                           d88P" "Y88b d88P  Y88b
     d88P888                           888     888 Y88b.
    d88P 888 888d888  .d8888b  .d88b.  888     888  "Y888b.
   d88P  888 888P"   d88P"    d8P  Y8b 888     888     "Y88b.
  d88P   888 888     888      88888888 888     888       "888
 d8888888888 888     Y88b.    Y8b.     Y88b. .d88P Y88b  d88P
d88P     888 888      "Y8888P  "Y8888   "Y88888P"   "Y8888P"

arch = x86_64
platform = x86_64-pc-oslab
target = x86_64-unknown-arceos
smp = 1
build_mode = release
log_level = off

Running ping pong server with Tokio.
Packet size: 1024
Listen 0.0.0.0:5555
CPU count: 1
```

【Arceos】

客户端每核连接数量（个）	有效吞吐量(比特/秒)	平均延迟(微秒)	备注
10	253406.953	157.313	
20	315276.219	253.121	
30	308952.000	387.764	
40	346951.594	460.446	
50	336820.719	592.993	
60	323998.281	739.803	
70	312270.625	895.652	
80	299635.219	1066.787	
90	288305.656	1247.396	

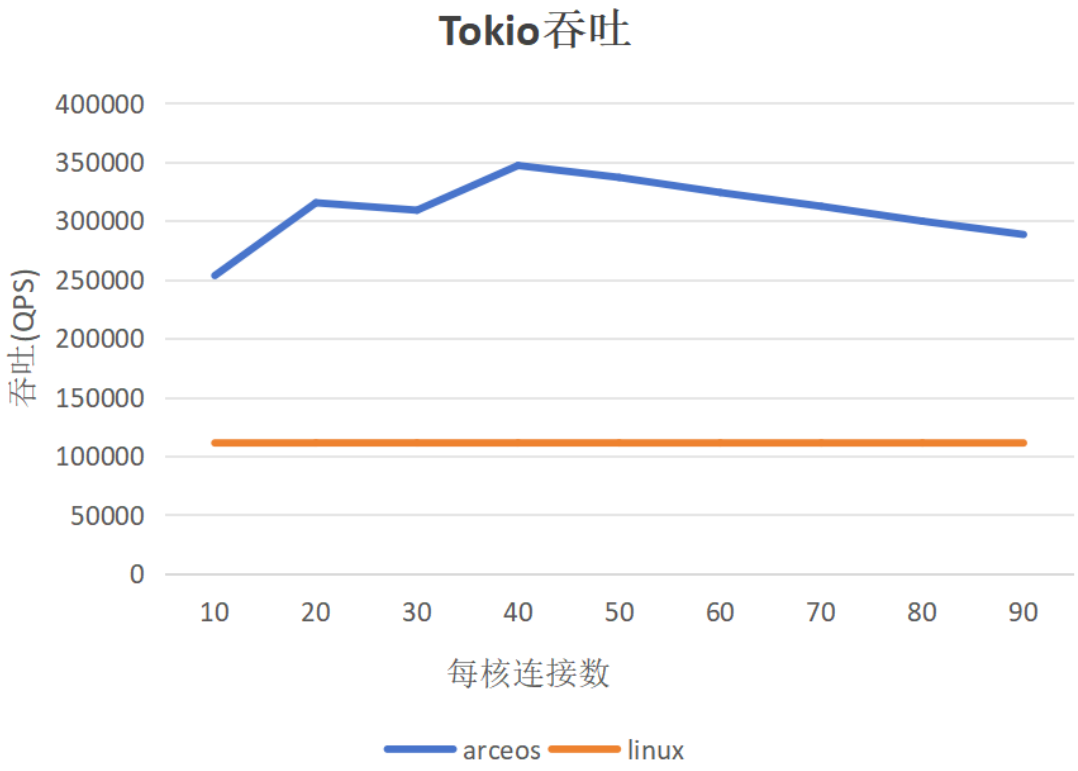
【Linux】

客户端每核连接数量（个）	有效吞吐量(比特/秒)	平均延迟(微秒)	备注
10	111138.195	359.23	
20	111176.641	718.814	
30	111160.18	1078.718	
40	111164.133	1438.307	
50	111131.344	1798.16	
60	111140.016	2157.913	
70	111134.516	2515.994	
80	111126.062	2877.47	
90	111112.914	3237.254	

【测试结论】

在x86硬件平台上支持运行tokio，相对Linux下，性能提升10%以上。

1. tokio分别在arceos和linux运行时，吞吐量的对比。



2. tokio分别在arceos和linux运行时，延迟的对比。