计算机系统应用 ISSN 1003-3254, CODEN CSAOBN
Computer Systems & Applications [doi: 10.15888/j.cnki.csa.008699]

E-mail: csa@iscas.ac.cn
http://www.c-s-a.org.cn
Tel: +86-10-62661041

基于 RISC-V 平台 openEuler 系统的 COMO 构件技术移植与应用①

黄玉坤[1], 裴喜龙[2], 徐志宇[2], 王建民[3]

[1](江西财经大学 信息管理学院, 南昌 330032)
[2](同济大学 电子与信息工程学院, 上海 200092)
[3](中国科学院 软件研究所, 北京 100190)
通信作者: 黄玉坤, E-mail: huangyukun@jxufe.edu.cn

摘　要: 随着云计算、物联网技术的发展, 边缘计算模式开始兴起. RISC-V 开源指令集架构和 openEuler 开源操作系统正在逐渐形成一个开放、灵活、不断演进和架构包容的软件生态体系, 为边缘计算应用的构建提供了良好的软硬件创新平台. 然而, 在其之上的支持面向边缘计算的软件开发环境、开发框架和工具链等基础设施尚不够完善. COMO 是解决 C++软件资产复用的构件化技术, 将 COMO 技术与 RISC-V、openEuler 结合有助于 RISC-V、openEuler 生态在新的软件技术架构上发展. 本文提出了在基于 RISC-V 架构和 openEuler 操作系统的软硬件平台上进行 COMO 构件程序运行与开发环境移植的思路和方法, 并通过实验证明了 COMO 构件技术与 RISC-V 指令集架构和 openEuler 操作系统的兼容性和可行性; 通过一个简单的实例介绍 COMO 的 ServiceManager 框架在边缘计算中的应用, 在为面向云计算与物联网的边缘计算应用提供 XaaS 服务的构件化程序开发模式的方向上进行了有益的探索.

关键词: RISC-V; openEuler 操作系统; COMO 构件技术; XaaS 服务架构

引用格式: 黄玉坤,裴喜龙,徐志宇,王建民.基于 RISC-V 平台 openEuler 系统的 COMO 构件技术移植与应用.计算机系统应用. http://www.c-s-a.org.cn/1003-3254/8699.html

Transplantation and Application of COMO Component Technology Based on RISC-V Platform and openEuler System

HUANG Yu-Kun[1], PEI Xi-Long[2], XU Zhi-Yu[2], WANG Jian-Min[3]

[1](School of Information Management, Jiangxi University of Finance and Economics, Nanchang 330032, China)
[2](Institute of Electronic and Information Engineering, Tongji University, Shanghai 200092, China)
[3](Institute of Software, Chinese Academy of Sciences, Beijing 100190, China)

Abstract: With the development of cloud computing and Internet of Things (IoT) technology, the edge computing mode begins to emerge. On the basis of the open-source instruction set architecture RISC-V and openEuler operating system (OS), an open, flexible, evolving, and architecture-inclusive software ecosystem has been gradually formed, which provides a good innovative software and hardware platform for the construction of edge computing applications. However, the infrastructure such as software development environments, frameworks, and toolchains for edge computing is not yet complete. The component model (COMO) is a component technology that solves the reuse issue for C++ software and assets, and the combination of which with RISC-V and openEuler is conducive to the development of RISC-V and the openEuler ecosystem in a new software development architecture. Therefore, this study proposes the method of operating the COMO program and transplanting the development environment on the software and hardware platform based on RISC-V and openEuler OS, and an experiment proves the compatibility and feasibility of COMO with RISC-V

① 基金项目: 江西省 03 专项及 5G 项目 (20204ABC03A40)
收稿时间: 2021-12-26; 修改时间: 2022-01-24; 采用时间: 2022-02-15; csa 在线出版时间: 2022-06-17

and openEuler OS. In addition, a simple example is introduced for the application of COMO's ServiceManager framework in edge computing. Our work makes a useful exploration in the component-based development mode of providing XaaS services for edge computing applications oriented to cloud computing and IoT.

Key words: RISC-V; openEuler operating system; COMO component technology; XaaS service architecture

近年来, 伴随着云计算、物联网、人工智能技术不断进步, 智能设备迅速发展, 数据无处不在, 最终实现万物互联. 移动终端和智能设备广泛普及, 异构计算开始兴起, 边缘计算应用爆发式增长[1,2]. 边缘计算是物联网中重要的一环, 它将算力下沉到边缘侧, 大大减少云端算力压力, 节约带宽, 且能快速响应端设备请求. 面向万物互联的边缘计算具有丰富的场景和应用[3,4], 如何应对不同场景下对算力的差异化需求, 对边缘计算平台软硬件体系的开放性、灵活性、可定制、可扩展性等方面提出了挑战.

在物联网碎片化环境中, RISC-V 架构具有广泛的发展前景. RISC-V 是一种开源的指令集架构 (instruction set architecture, ISA), RISC-V 指令集架构的开放性、模块化、高度可定制性的特点使得其成为体系结构和软件系统创新的理想实验平台[5,6]. RISC-V 的发展十分迅速, 涌现了大量采用 RISC-V 指令集架构的开源或商用处理器和 SoC, 如鲲鹏、PulSAR 等[7]; 除了硬件的微架构设计、逻辑设计外, 在软件生态方面也日渐成熟, 如 UCB 提供了 RISC-V 的开源编译器 GCC、LLVM[8], 开源仿真器 Spike、QEMU, 社区也实现了支持 RISC-V 架构的 openEuler[9] 开源操作系统和 FreeBSD、Debian 等操作系统的移植. 在一个系统级的方案中, 芯片是核心和基础, 但构筑在芯片之上的整个系统软硬件方案, 才是最终与应用直接接口的关键.

在边缘计算领域中, 采用 RISC-V 架构的操作系统无论是内核的架构还是应用组织的方式, 都在不断的发生着创新和变化. openEuler 是一个基于 RISC-V 指令集架构的开源 Linux 发行版本, 具有开放、灵活、不断演进和架构包容的软件生态体系. openEuler 的目标是从系统软件的角度, 打通不同算力, 让开发者可以在其之上进行技术创新, 使其适应多样性的计算场景, 支撑边缘侧应用的运行需求. 因此, 基于 RISC-V 指令集架构的 openEuler 操作系统是一个很好的构建边缘计算应用的创新平台.

然而, 边缘计算应用对底层软件工具链的要求也非常高, 包括对一些主流的算法库和程序开发框架的支持, 这对上层开发者的使用体验以及市场的接受度和认可度有直接的影响. 理想状态下, 应用开发者应该可以在无需了解芯片底层硬件的细节和指令集架构的情况下进行应用程序的开发, 这需要扩展操作系统功能、增强底层软件开发环境和工具链. 另外, 不同的应用场景和应用模式, 对面向边缘计算的软件开发框架等基础设施也提出了需求.

1 基于 RISC-V 指令集架构的边缘计算软硬件基础服务架构

在云计算和边缘计算的软件架构中, 广泛采用了 XaaS 模式, 即 Paas 平台即服务、IaaS 基础设施即服务、SaaS 软件即服务等模式. 在边缘计算中实现面向 XaaS 的软件架构能够将边缘端与云端进行融合. COMO (C++ component model) 技术[10,11] 是一个改进 C++构件技术的开源项目, 它可以将细粒度的计算 (机器指令或软件程序) 抽象成构件服务, 并提供 XaaS 服务架构, 有利于对软件系统在编译阶段和运行时的动态演进.

RISC-V 架构相比其他成熟的商业架构的最大的不同在于它是一个模块化的架构. ISA 在 CPU 软件和 CPU 硬件设计者之间, 提供了一个抽象层 (接口). RISC-V 架构不仅短小精悍, 而且其不同的部分还能以模块化的方式组织在一起, 从而试图通过一套统一的架构满足各种不同的应用. 基于虚拟原型 (virtual prototype) 扩展和配置 RISC-V 是基于 RISC-V 的计算机系统开发的重要手段[12,13].

COMO 技术可以在系统服务和软件服务抽象的接口层面保持构件的一致性. COMO 同样采用虚拟原型的设计方法, 把程序分为元数据和逻辑两层, 元数据则是程序的原型抽象, 这一特性使得 COMO 构件可以在支持 RISC-V 架构的系统中与底层硬件的接口层保持统一的模块化结构. 结合 RISC-V 架构及 openEuler 的模块化设计的技术特点, COMO 提供了一个面向 XaaS 模式的 ServiceManager 软件开发框架, 使得 COMO 可

以在云计算、边缘计算及其结合的应用场景提供一个统一的构件化程序运行环境和开发模型. COMO 在 ISC-V 指令集和 openEuler 操作系统上的系统架构如图 1 所示.

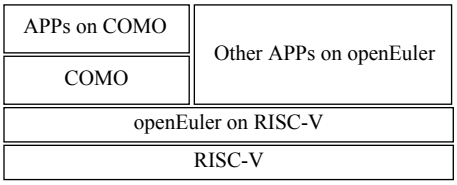

图 1 COMO 在 RISC-V 指令集和 openEuler 操作系统上的系统架构

本文的主要工作是在基于 RISC-V 架构和 openEuler 操作系统的软硬件平台上移植 COMO 构件程序运行与开发环境, 为面向云计算与物联网的边缘计算应用提供面向 XaaS 服务的构件化程序开发模式; 并通过一个简单的实例介绍 COMO 的 ServiceManager 框架在边缘计算中的应用. 基于 RISC-V 和 openEuler 系统之上的 QEMU 模拟器进行了 COMO 运行与开发环境移植的实验, 证明了 COMO 构件技术与 RISC-V 指令集架构和 openEuler 操作系统的兼容性和可行性. COMO 构件技术与 RISC-V 指令集系统和 openEuler 操作系统的结合, 为 openEuler 生态圈注入了新的特性和活力, 同时, 借助 RISC-V 硬件技术和 openEuler 生态圈的发展, COMO 构件技术能够获得更多机会得到推广和使用.

2 COMO 在 RISC-V 指令集架构和 openEuler 系统上的移植

2.1 基于 QEMU 搭建 RISC-V 上的 openEuler 系统环境

由于 COMO、openEuler、RISC-V 都处于研发的起步和发展阶段, 从无到有地研发一个计算环境比开发一个应用要难得多. 在一个新的体系架构的计算机上开发软件系统, 首先要在上面运行起较完整的开发环境, 本文采用 openEuler 的 RISC-V 镜像作为实验环境[14]. 目前支持“openEuler on RISC-V”的硬件平台有: (1) NutShell (果壳, UCAS) COOSCA 1.0; (2) SiFive HiFive Unleashed. 然而现在 RISC-V 还不普及, 在实际的硬件平台上进行移植工作之前, 利用 RISC-V 的 QEMU 进行 COMO 移植的仿真, 可以以较低的硬件成本和时间成本验证 COMO 移植的可行性和 COMO 构件在 openEuler on RISC-V 上运行的兼容性.

工作的第一步需要通过 QEMU 仿真出 RISC-V 硬件. 本文的实验环境是 X86 的 Ubuntu 18.04, 以及自己编译 RISC-V 版的 QEMU[15–17].

从官方资源库下载 openEuler RISC-V 移植版, 并通过 wget 下载互联网资源.

通过下列命令启动虚拟机.

```
$qemu-system-riscv64
-nographic -machine virt
-smp 8 -m 2G
-kernel fw_payload_oe.elf
-drive
file=oe-rv-rv64g-30G.qcow2, format=qcow2, id=hd0
-object rng-random, filename=/dev/urandom, id=rng0
-device virtio-rng-device, rng=rng0
-device virtio-blk-device, drive=hd0
-device virtio-net-device, netdev=usernet
-netdev user, id=usernet, hostfwd=tcp::12055-:22
-append 'root=/dev/vda1 rw console=ttyS0
systemd.default_timeout_start_sec=600 selinux=0
highres=off mem=4096M earlycon'
```

RISC-V 没有 X86 平台上的基本输入输出系统 BIOS, 所以 openEuler 操作系统的引导加载程序是基于 OpenSBI 项目的 OpenSBI RISC-V, 后者实现了 Supervisor 二进制接口[18]. 如果虚拟机正常工作, 则显示结果如图 2 所示. RISC-V 上 openEuler 成功启动后的界面如图 3 所示.

图 2 qemu-system-riscv64 起动界面

2.2 搭建 COMO 的 build 环境

COMO 的组成部分包括: (1) 工具部分, C++构件化需要的构件定义语言 (component definition language, CDL) 编译器 cdlc、构件编译环境; (2) 与 Java 基础类库对应的 Libcore 核心库. openEuler 在 RISC-V 上的基础类库支持不够完整, 所以这里把第 2 部分做了部分裁剪. 这样做的好处是能得到一个精简的 COMO, 使得精简 COMO 可以在计算能力更弱的环境中运行. 下面以 RISC-V 为开发机, 进行本地编译.

```
openEuler 20.03 (LTS)
Kernel 5.5.19 on an riscv64

openEuler-RISCV-rare login:
```

图 3 RISC-V 上 openEuler 启动成功

一个软件, 如果能在与目标机体系结构及操作系统等硬件、软件一致的环境中编译自己, 那比较容易实现开发、调试、测试的集成化. 可是 openEuler 在 RISC-V 上还很弱小, 这个环境的可安装包也不完整, 于是只能从编译 COMO 所需要的环境开始构建, 即以 RISC-V 为开发机, 进行本地编译, 除了 cmake, 其它 COMO 所依赖工具都是从源码开始构建的.

从源码开始安装 cmake, 遇到了 C/C++编译器版本上的问题. 通过验证, 这个问题可以在 openEuler 发行版上解决, 从源头治理一个生态往往成本是最小的, 解决软件包 (库) 的版本依赖是所有操作系统开发商都要面对的困难. 最后决定从 openEuler 的发行软件包中安装 cmake.

通过 yum 安装 cmake 等软件时, 遇到了“Cannot download repomd.xml”错误, 这是软件的安装源有问题引起的, 可以用非正式 (preview) 的源, 并在.repo 文件中加 sslverify=0 解决.

```
/etc/yum.repos.d/oe-rv.repo
[base]
name=base
baseurl=https://isrc.iscas.ac.cn/mirror/openeuler-sig-riscv/oe-RISCV-repo/
enabled=1
gpgcheck=0
sslverify=0
```

在 git clone COMO 源码仓库时, 遇到“SSL certificate problem”问题, 可通过下列命令解决.

git config --global http.sslVerify false

在“openEuler on RISC-V”环境中编译 COMO, 选 COMO 本地 Linux 编译环境: como_linux_riscv64.

由于 COMO 仓库中缺省的构件描述语言编译工具 cdlc 是面向 openEuler X86 平台的, 所以在编译 COMO 前, 首先要编译出 COMO 工具链, 所用的 COMO 环境是: Comotools.

在 Comotools 环境中编译完成 cdlc 后, 发布 cdlc 工具链:

cp ./out/host/como/tools/cdlc/cdlc ./tools/cdlc

编译 COMO 前, 需要安装并配置 dbus 开发包.

$yum install dbus-devel

$ cp ./lib64/dbus-1.0/include/dbus/dbus-arch-deps.h /usr/include/dbus-1.0/dbus

在搭建完成 openEuler on RISC-V 和 COMO 的 build 环境之后, 下一步的工作是 COMO 的代码移植.

2.3 COMO 代码移植

将 COMO 代码移植到 openEuler 平台上需要关注以下问题:

(1) ELF 格式的约定

openEuler 是一种 Linux 操作系统, 它的可执行文件及动态链接库等二进制文件是 ELF 格式的, COMO 的元数据存放在 ELF 格式文件的“.metadata”段中, 所以 COMO 在 openEuler 上没有发行文件格式上的问题.

COMO 采用与 Java 类似的元数据与代码编译结果放在一起的发行方式, 构件存储为 ELF 格式文件. 传统的基于 PE/ELF 可执行文件格式的线性符号表导出为标志的软件运行时资源共享机制, 在计算能力已经大大发展的时代, 显得臃肿. COMO 的类、接口、方法等定义信息是通过元数据表达的, 不产生导出符号, 所以其发行文件的导出符号表很简洁.

(2) RISC-V EABI 参数对齐

各种约定的一致性是评价一个操作系统成熟度的重要指标. COMO 代码中, 反射机制构造调用栈的一段汇编代码是与平台相关的, 它要求把从 RPC 等渠道得到的 COMO 方法调用参数, 遵守 RISC-V ABI 规范形成调用栈. ABI 定义了调用 C/C++程序传参数方法.

嵌入式系统上的 RISC-V 的 ABI 标准是 EABI, Linux 上的 RISC-V 的 ABI 标准是 UABI[19,20]. COMO 采用了 RISC-V EABI 参数对齐规则: 如果参数可以完

全放在寄存器内, 则放在寄存器内, 否则把寄存器内摆放不下的参数放在栈里. 无论参数是在寄存器里, 还是在栈上, 所有参数传递时要数据对齐. XLEN 这个参数是指数据总线宽度, 可能的值是 32 或 64.

1) 对于 XLEN=32 的系统, 数据总线宽度 8 B

2) 对于 XLEN=64 的系统, 数据总线宽度 16 B

EABI 函数调用参数所用寄存器, 如表 1 所示. 浮点数参数也通过整数寄存器 (x10–x11、x12–x17) 或者栈传递.

表 1 EABI 函数调用参数所用寄存器

寄存器	ABI名字	描述信息
x10–x11	a0–1	参数/返回值
x12–x17	a2–7	参数
f10–f17	fa0–fa7	参数
x2	sp	栈指针
x8	s0/fp	帧指针
x1	ra	函数调用返回指针

(3) COMO 在 RISC-V 上的移植

COMO 是采用 C++ 11 (–std=gnu++11) 标准实现的, 但有少部分用汇编写的程序是与计算机体系结构相关的, 以下代码需要用 RISC-V 汇编重写:

reflection/CMetaConstructor.cpp

reflection/CMetaMethod.cpp

reflection/invoke_riscv64.s

rpc/CProxy.cpp

另外还需要创建一个 RISC-V 的 CMake 编译脚本文件:

build/como_linux_riscv64.cmake

3 COMO 技术的 ServiceManager 框架在边缘计算中的应用实验

为了验证 COMO 构件技术在 RISC-V 指令集架构和 openEuler 操作系统上的兼容性、可行性和实用性, 搭建基于 RISC-V 和 openEuler 上的 QEMU 模拟器的实验环境并实现 COMO 技术的移植, 利用 COMO 的协同计算能力, 组成一个边缘计算集群, 实现多台 RISC-V 节点协同计算, 实验软硬件架构如图 4 所示.

COMO 构件技术能够方便实现程序框架模型, 但它本身不强调对某一框架的支持. ServiceManager 是 COMO 面向服务的开发框架之一. ServiceManager 提供了 Server 的 Name 和 Handle 之间对应关系的查询能力, 它主要包含的功能: (1) 注册, 当一个 Server 创建后, 应该将这个 Server 的 Name 和 Handle 对应关系记录到 ServiceManager 中. (2) 查询, 其他应用可以根据 Server 的 Name 查询到对应的 Service Handle.

为了验证“openEuler on RISC-V”的网络通信能力, 基于 IP 通信, 实现了一个 COMO ServiceManager 版本的应用程序, 该程序位于 COMO 源码中的 samples/democomponent 目录, 由 Client 和 Component 两部分组成. Component 实现了一个 COMO 构件, Client 实现了如何对这个构件进行调用. 类似于 OSGi, ServiceManager 框架实现了一个优雅、完整和动态的组件模型, 构件无需重新引导就可以被远程安装、启动、升级和卸载. 本实验中, 客户端和服务管理程序都运行在 Master 结点, 服务提供者运行在 Workers 结点, 通过 FindService 找到 COMO 构件的服务, 不需要知道它的提供者在哪个结点运行, 就可以调用它的服务.

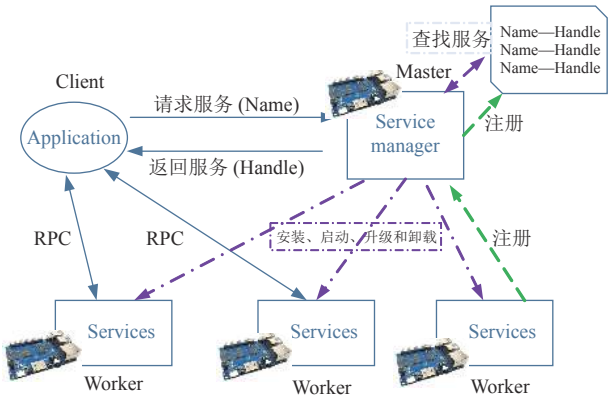

图 4 COMO 技术的 ServiceManager 框架实现集群式 RISC-V 协同计算

运行起来的 COMO 示例程序效果如图 5 所示, 这个示例演示了一个宿主程序通过反射机制调用 COMO 构件编写的服务组件的过程.

4 结论与展望

随着云计算、物联网等技术的兴起和广泛应用, 边缘计算作为云计算的补充和延伸, 可充分利用边缘算力. 但是, 边缘计算因为其条件受限的软硬件部署限制, 对服务器在环境适配性、程序运行环境的部署和程序运行的效率上有更严苛要求. 面对海量边缘应用场景, 硬件指令集架构 ISA、软件基础平台的操作系统和其上的软件开发框架、程序编译、运行的工具链等基础设施尤为重要.

本文完成了基于 RISC-V 和 openEuler 上的 QEMU 模拟器的实验环境搭建和 COMO 运行与开发环境的移植, 证明了 COMO 构件技术与 RISC-V 指令集架构和 openEuler 操作系统的兼容性和可行性. 实验应用实

例表明, 在一个以 RISC-V 指令集架构和 openEuler 操作系统为软硬件平台的集群上可以成功运行 COMO 构件技术和其所支持的 ServiceManager 框架编写的面向服务的应用程序, 为云计算和物联网为主要应用场景的边缘计算提供了一个简单、有效和可行的面向 XaaS 计算模式, 以及一个与 C++语言无缝衔接的构件化程序设计开发模式.

由于现在 RISC-V 的发展还不够成熟, 不能完全靠 RISC-V 环境把 COMO 完整编译出来, 因此提供交叉编译支持就很必要. 本文下一步的工作方向是, 尝试使用 Clang/LLVM 编译器, 把 COMO 裁成多个比较小的模块集合. 但这个尝试目前还处于交叉编译环境定义阶段. 随着交叉编译环境的完善, 逐步发现 openEuler 上 Clang/LLVM 应该做的工作. Clang/LLVM 是现在使用较多的编译器[21], 所以这个尝试是很有意义的.

希望 COMO 能在这个新的工具链建设中发挥一定作用, 并在与 openEuler 与 RISC-V 社区共同发展的过程中壮大自己.

```
[root@openEuler-RISCV-rare como.linux.riscv64.rls]#
==== Call CFoo() ====
==== Call CFoo::Foo, data is 9 ====
==== component name: FooBarDemo ====

==== component class number: 2 ====
==== [0] class name: CFoo, namespace: como::demo
==== [1] class name: CFooBar, namespace: como::de

==== component interface number: 3 ====
==== [0] interface name: IFoo, namespace: como::de
==== [1] interface name: IBar, namespace: como::der
==== [2] interface name: ICFooBarClassObject, name

==== Call CFoo() ====
==== Call ~CFoo() ====
==== Call ~CFoo() ====
```

图 5 COMO 示例的运行结果

参考文献

1 Tufail A, Namoun A, Alrehaili A, *et al*. A survey on 5G enabled multi-access edge computing for smart cities: Issues and future prospects. IJCSNS International Journal of Computer Science and Network Security, 2021, 21(6): 107–118. [doi: 10.22937/IJCSNS.2021.21.6.15]

2 Xia XY, Chen FF, He Q, *et al*. Cost-effective app data distribution in edge computing. IEEE Transactions on Parallel and Distributed Systems, 2021, 32(1): 31–44. [doi: 10.1109/TPDS.2020.3010521]

3 Dai HJ, Zeng XY, Yu ZL, *et al*. A scheduling algorithm for autonomous driving tasks on mobile edge computing servers. Journal of Systems Architecture, 2019, 94: 14–23. [doi: 10.1016/j.sysarc.2019.02.004]

4 Yousefpour A, Fung C, Nguyen T, *et al*. All one needs to know about fog computing and related edge computing paradigms: A complete survey. Journal of Systems Architecture, 2019, 98: 289–330. [doi: 10.1016/j.sysarc.2019.02.009]

5 何小庆. RISC-V 处理器嵌入式开发概述. 单片机与嵌入式系统应用, 2020, 20(11): 1–6.

6 苏鹏, 景乃锋. 基于 RISC-V 的异构系统任务管理机制设计与研究. 微电子学与计算机, 2020, 37(9): 6–10.

7 周鹏. openEuler4RISC-V: 构建 openEuler 面向 RISC-V 的操作系统. https://www.bilibili.com/video/av413971430/. (2020-07-22).

8 Chris Lattner. RISC-V LLVM. https://github.com/sifive/riscv-llvm. (2021-08-06).

9 Bestony. 除了操作系统, openEuler 还可以是什么. https://xw.qq.com/cmsid/20201225A0H4IV00. (2020-01-25).

10 Pei XL. COMO Mirror warehouse and software source. https://gitee.com/tjopenlab/como. (2020-12-20).

11 黄玉坤, 陈榕, 裴喜龙, 等. 基于跨语言对象迁移策略的复合本地对象模型. 计算机研究与发展, 2015, 52(1): 141–155. [doi: 10.7544/issn1000-1239.2015.20131166]

12 Bradbury A. RISC-V Toolchain Conventions. https://github.com/riscv/riscv-toolchain-conventions. (2021-06-07).

13 Herdt V, Grobe D, Le HM, *et al*. Extensible and configurable RISC-V based virtual prototype. 2018 Forum on specification & Design Languages (FDL). Garching: IEEE, 2018. 5–16.

14 openEuler RISC-V. https://gitee.com/openeuler/RISC-V. (2021-02-30).

15 RISC-V Foundation. Running 64- and 32-bit RISC-V Linux on QEMU. https://risc-v-getting-started-guide.readthedocs.io/en/latest/linux-qemu.html. (2019-20-20).

16 汪辰. 在 QEMU 上运行 RISC-V 64 位版本的 Linux. https://zhuanlan.zhihu.com/p/258394849. (2020-09-23).

17 Roa Logic. RV12 RISC-V 32/64-bit CPU core datasheet (v1.3). https://roalogic.github.io/RV12/docs/RoaLogic_RV12_RISCV_Datasheet.pdf. (2018-02-01).

18 OpenSBI. RISC-V. https://www.oschina.net/p/opensbi?hmsr=aladdin1e1. (2019-07-23).

19 Jessica C. RISC-V ELF specification. https://github.com/riscv-non-isa/riscv-elf-psabi-doc/blob/master/riscv-elf.adoc. (2021-08-11).

20 Krste A. Proposal for a RISC-V embedded ABI (EABI). https://github.com/riscv/riscv-eabi-spec/blob/master/EABI.adoc. (2009-05-25).

21 王鹏, 陈影, 邢明杰. 基于 LLVM 的 RISC-V 自定义扩展指令支持方法. 计算机系统应用, 2021, 30(11): 20–26. [doi: 10.15888/j.cnki.csa.008347]

(校对责编: 孙君艳)