关于基于 WebAssembly 的边缘计算研究

万益民，张峰，王鹏
亚信科技（中国）有限公司，江苏南京 210000

摘　要：随着移动通信技术的发展，在物联网技术的快速发展和云服务的推动下，边缘计算成为 5G 时代的必然选择，但是边缘计算发展也面临着很多挑战，在边缘计算中，针对固定互联网、移动通信网、消费物联网、工业互联网等不同网络接入和承载技术中，边缘计算的技术实现上存在一定差异，计算任务分到各种不同平台的边缘节点上，不仅使得程序开发者面临着巨大的困难，而且存在着安全问题，通过基于 WebAssembly 的技术使用各个平台的浏览器作为计算容器，降低了开发难度、提高了使用的安全性，大大提升可以用于进行边缘计算的设备范围和数量，从而充分调动客户拥有的算力。

关键词：边缘计算；WebAssembly；云计算；物联网

A Study of edge computing Based on WebAssembly

Yimin Wan, Feng Zhang, Peng Wang
AsiaInfo Technology (China) Co., ltd，Nanjing Jiangsu 100193, China

Abstract: With the development of mobile communication technology, with the rapid development of Internet of Things technology and the promotion of cloud services, edge computing has become an inevitable choice in the 5G era, but the development of edge computing also faces many challenges. For fixed network, mobile communication network, consumer Internet of things, industrial Internet and other different network access and bearer technologies, there are some differences in the implementation of edge computing technology. The computing tasks are distributed to the edge nodes of different platforms, which not only makes the program developers face Huge difficulties, and security problems, using WebAssembly-based technology to use the browser of each platform as a computing container, reducing the development difficulty, improving the security of use, greatly improving the range and number of devices that can be used for edge computing In order to fully mobilize the computing power owned by the customer.

Key words: Edge Computing; WebAssembly; Cloud Computing; Internet of Things

引言

随着 5G 时代的到来，5G 网络是面向物联网通信网格，而边缘计算是面向物联网的 IT 支撑，因此 5G 时代下边缘计算成为必然。但是面对纷繁的终端平台，如何架起边缘计算与云计算的桥梁是一个关键技术，为了解决这个桥梁在跨平台、安全性方面的问题，本文提出了通过基于 WebAssembly 的边缘计算解决方案。

1　边缘计算的通用架构

边缘计算是在靠近物或数据源头的网络边缘侧，融合网格、计算、存储、应用核心能力的分布式开放平台（架构），就近提供边缘智能服务，满足

行业数字化在敏捷联接、实时业务、数据优化、应用智能、安全与隐私保护等方面的关键需求[1]。可以作为联接物理和数字世界的桥梁，使能智能资产、智能网关、智能系统和智能服务。

那么边缘计算与云计算间如何进行交互呢？边缘计算现有方案一般分为以下几个步骤[2]：

□ • 需要根据不同的系统平台（Linux、Android、Windows 等）集成对应的边缘计算 SDK，改造用于边缘计算的设备；

□ • 边缘计算设备连接到云端时，边缘计算设备利用 SDK 把自己注册到算力库；

□ • 云端把计算任务分配到边缘计算设备，利用边缘计算设备的空闲算力进行计算；

□ • 边缘计算设备将计算结果反馈到云端，由云端进行计算结果汇总，得出计算结果。

2 WebAssembly 技术特点

WebAssembly 是一个可移植、体积小、加载快并且即兼容 Web 的一种新的字节码格式，主流浏览器都已经支持 WebAssembly。WebAssembly 字节码和底层机器码很相似可快速装载运行，因此性能相对于 JS 解释执行大大提升。WebAssembly 作为一份字节码标准，需要用高级编程语言编译出字节码放到 WebAssembly 虚拟机中才能运行，浏览器厂商是根据 WebAssembly 规范实现虚拟机。

对于网络平台而言，WebAssembly 具有巨大的意义——它提供了一条途径，以使得以各种语言编写的代码都可以以接近原生的速度在 Web 中运行。在这种情况下，以前无法以此方式运行的客户端软件都将可以运行在 Web 中。而且，这是通过 W3C WebAssembly Community Group 开发的一项网络标准，并得到了来自各大主要浏览器厂商的积极参与。

以下主要阐述 WebAssembly 对于我们提升边缘计算架构的技术特性：

2.1 跨平台性

WebAssembly 字节码是一种抹平了不同 CPU 架构的机器码，WebAssembly 字节码不能直接在任何一种 CPU 架构上运行，但由于非常接近机器码，可以非常快的被翻译为对应架构的机器码，因此 WebAssembly 运行速度和机器码接近。

WebAssembly 有如下优点：

- 体积小：由于浏览器运行时只加载编译成的字节码，一样的逻辑比用字符串描述的 JS 文件体积要小很多；
- 加载快：由于文件体积小，再加上无需解释执行，WebAssembly 能更快的加载并实例化，减少运行前的等待时间；
- 兼容性问题少：WebAssembly 是非常底层的字节码规范，制订好后很少变动，就算以后发生变化,也只需在从高级语言编译成字节码过程中做兼容。可能出现兼容性问题的地方在于 JS 和 WebAssembly 桥接的 JS 接口。

每个高级语言都去实现源码到不同平台的机器码的转换工作是重复的，高级语言只需要生成底层虚拟机(LLVM)认识的中间语言(LLVM IR)，LLVM 能实现：

- LLVM IR 到不同 CPU 架构机器码的生成；
- 机器码编译时性能和大小优化。

除此之外 LLVM 还实现了 LLVM IR 到 WebAssembly 字节码的编译功能，也就是说只要高级语言能转换成 LLVM IR，就能被编译成 WebAssembly 字节码，目前能编译成 WebAssembly 字节码的高级语言有：

- AssemblyScript:语法和 TypeScript 一致，对前端来说学习成本低，为前端编写 WebAssembly 最佳选择；
- c\c++:官方推荐的方式；
- Rust:语法复杂、学习成本高，对前端来说可能会不适应；
- Kotlin:语法和 Java、JS 相似，语言学习成本低；
- Golang:语法简单学习成本低。但对

WebAssembly 的支持还处于未正式发布阶段。

通常负责把高级语言翻译到 LLVM IR 的部分叫做编译器前端，把 LLVM IR 编译成各架构 CPU 对应机器码的部分叫做编译器后端；现在越来越多的高级编程语言选择 LLVM 作为后端，高级语言只需专注于如何提供开发效率更高的语法同时保持翻译到 LLVM IR 的程序执行性能。

2.2 性能

WebAssembly 适合用于需要大量计算的场景，例如：

- 在浏览器中处理音视频，flv.js 用 WebAssembly 重写后性能会有很大提升；
- React 的 dom diff 中涉及到大量计算，用 WebAssembly 重写 React 核心模块能提升性能。Safari 浏览器使用的 JS 引擎 JavaScriptCore 也已经支持 WebAssembly，RN 应用性能也能提升；
- 突破大型 3D 网页游戏性能瓶颈，白鹭引擎已经开始探索用 WebAssembly。

2.3 风险

WebAssembly 标准虽然已经定稿并且得到主流浏览器的实现，但目前还存在以下问题：

- 浏览器兼容性不好，只有最新版本的浏览器支持，并且不同的浏览器对 JS WebAssembly 互调的 API 支持不一致；
- 生态工具不完善不成熟，目前还不能找到一门体验流畅的编写 WebAssembly 的语言，都还处于起步阶段；
- 学习资料太少，还需要更多的人去探索去踩坑。

3 边缘计算的改进架构

在边缘计算的通用架构中由于必须对边缘计算设备进行改造以集成边缘计算 SDK，这就带来了一些安全隐患，这是部分设备拥有者无法接受的。另外，改造带来的额外工作量也是影响了部分用户的改造决心。显然，这带来了系统整体算力的潜在损失。同时，SDK 的开发需要针对不同平台开发，给边缘计算平台的提供方也带来了额外的开发工作量。为此我们设计了以下改进架构。

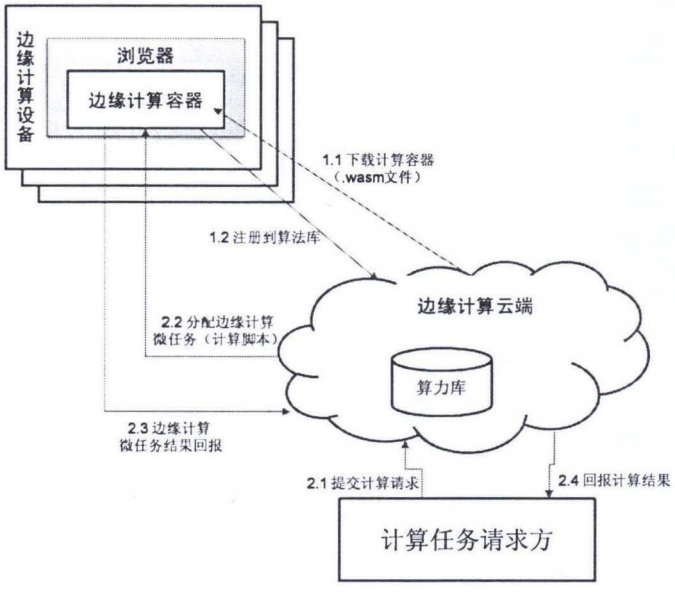
3.1 整体步骤

整体原理如下图所示：

整个过程分为准备阶段和计算阶段，其中准备阶段步骤如下：

1. 边缘计算平台将边缘计算容器编译成浏览器支持的 WebAssembly 格式；

2. 用户用浏览器连接到平台提供的网站，对应



网页自动下载边缘计算容器。在计算容器不需要升级的情况下，本动作只需要执行一次。为提升下载性能，可以使用 WebSocket 技术进行下载；

3. 边缘计算设备启动计算容器（可以用定时器技术）；

4. 边缘计算设备把本设备注册到云端算力库，开始接收边缘计算微任务。（为提升传送性能，要求设备到云端用 WebSocket 技术建立长连接）。

3.2 计算阶段步骤

其中计算阶段步骤如下：

1. 边缘计算平台的云端接收计算任务后，将计算任务分解为边缘计算微任务；为避免影响边缘计算设备原有职能的执行，要求每个微任务的执行时间不能超过 200 毫秒，但可以一次要求边缘设备进行同一微任务的多次计算；

2. 边缘计算平台根据当前算力库信息，向对应设备分配边缘计算微任务，为保证计算任务不受个别设备影响，任务分配采用冗余分配方式。任务的分配采用 WebSocket 技术传输，并采用效率较高的压缩算法在不影响算力的基础上减少网络传输量；

3. 边缘计算设备接收到计算任务，在计算容器中进行计算。单次微任务计算不能超过 200ms，计算容器有相关机制打断超时的计算并返回计算失败；

4. 边缘计算设备完成微任务后返回，由云端进行结果汇总并最终完成。

4 结语

利用目前大部分平台（Linux、Android、Windows 等）都具备的浏览器技术作为计算容器，从而避免了设备进行改造的工作量。由于直接基于浏览器技术，其安全性也由浏览器背书，使得客户对计算的安全性更加放心。最后，由于基于浏览器的 WebAssembly 技术，所以也不需要为每个平台开发单独的 SDK。大大提升可以用于进行边缘计算的设备范围和数量，从而充分调动客户拥有的算力。

参考文献：

[1] 施巍松, 刘芳, 孙辉, 裴庆祺.边缘计算[M].北京：科学出版社，2018.1-1.

[2] 蒋濛 (Mung Chiang), 巴拉思·巴拉萨布拉曼尼安 (Bharath Balasubramanian), 弗拉维奥·博诺米 (Flavio Bonomi). 雾计算:技术、架构及应用[M].北京：机械工业出版社，2018.1-1

[作者简介]

万益民（1971-），江苏南京，本科，南京亚信软件有限公司，资深工程师，应用与数据规划架构

张峰（1972-），福建福州，本科，南京亚信软件有限公司，资深工程师，应用与数据规划架构

王鹏（1976-），北京，硕士，亚信科技（中国）有限公司，中台产品研发与交付中心总经理，产品规划与研发