

**SEAWAY Edge**

**应用移植参考文档**

中科海微（北京）科技有限公司

Seaway Technologies Co. Ltd

## **目录**

[目录 1](#_Toc1514737459)

[文档修订目录 1](#_Toc1921321875)

[文档版本 1](#_Toc2092636833)

[Version3.2 1](#_Toc1607565312)

[第一章 综述 4](#_Toc858435877)

[1.1引言 4](#_Toc936644193)

[1.2背景与目标 4](#_Toc1123819241)

[1.3关键步骤 4](#_Toc911308122)

[1.4 面临的挑战 4](#_Toc502236050)

[第二章 操作系统软件的迁移与移植 5](#_Toc1470559640)

[2.1软件移植到国产操作系统的迁移过程 5](#_Toc306576157)

[2.2 WINDOWS应用分析和移植方式 7](#_Toc812201546)

[2.3应用移植分类小结 8](#_Toc1265323290)

[2.4应用移植技术难点 8](#_Toc1905462436)

[2.5迁移中遇到的常见问题及解决 9](#_Toc1831017788)

[第三章 服务器上应用软件移植与国产化迁移 9](#_Toc475301406)

[3.1应用迁移基本过程 9](#_Toc1899047449)

[3.2应用迁移具体方法 11](#_Toc1388513629)

[第四章 MySQL 5.7.21 移植指南 14](#_Toc43770654)

[4.1简要介绍 14](#_Toc1213974504)

[4.2 案例环境 15](#_Toc27358581)

[4.3 案例环境 15](#_Toc254170409)

[4.4 性能结果评估 16](#_Toc497090180)

[4.5 安装数据库 mysql 18](#_Toc883268430)

[第五章 Apache 2.4.39 移植 20](#_Toc1685534946)

[5.1 环境要求 20](#_Toc1329049845)

[5.2配置编译环境 20](#_Toc1363332468)

[5.3参数配置 21](#_Toc2025759833)

[5.4验证 21](#_Toc739773693)

[第六章 Dubbo 2.6.8 移植 22](#_Toc1593625768)

[6.1 环境要求 22](#_Toc652237392)

[6.2配置编译环境 22](#_Toc1397313056)

[6.3编译Dubbo2.6.8 24](#_Toc1906852247)

[第七章 HAProxy 1.9.0 移植 25](#_Toc1567251148)

[7.1 镜像站 RPM 方式安装 25](#_Toc1903113981)

[7.2运行和验证 26](#_Toc1015240249)

[第八章 Lighttpd 1.4.53 移植 29](#_Toc1385289528)

[8.1 环境要求 29](#_Toc1690879969)

[8.2 配置编译环境 30](#_Toc968538232)

[8.3 配置参数 30](#_Toc296020964)

[8.4 服务器测 32](#_Toc1652215496)

[第九章 Iok 2.1.3 移植 32](#_Toc1822285562)

[9.1 兼容性检查 32](#_Toc1889150667)

[9.2 查看评估结果 33](#_Toc409539374)

[9.3 依赖包引入 33](#_Toc443170183)

[9.4 构建流程 34](#_Toc887977885)

[第十章 Nginx 1.14.2 移植 35](#_Toc1380450192)

[10.1 环境要求 35](#_Toc1960538403)

[10.2 配置编译环境 35](#_Toc1927343300)

[10.3 获取源码 36](#_Toc215511752)

[10.4 编译和安装 37](#_Toc1448587022)

[10.5 运行和验证 37](#_Toc379972715)

[第十一章 Tengine 2.2.2 移植 45](#_Toc1732538474)

[11.1 环境要求 45](#_Toc1043362845)

[11.2 安装Tengine 45](#_Toc1595358160)

[11.3 运行和验证 48](#_Toc1851262325)

# 文档修订目录

## 文档版本

## Version3.2

|  |  |  |  |
| --- | --- | --- | --- |
| 文档版本号 | 修订日期 | 修订内容 | 修订人 |
| V3.1 | 2023/04/29 | 初订 | 陈海鑫 |

**前言**

在使用本手册之前，请您认真阅读以下使用许可协议，只有在同意以下使用许可协议的情况下方能使用本手册中介绍的产品。

**版权声明**

中科海微（北京）科技有限公司版权所有，并保留对本文档及本声明的最终解释权和修改权。本文档中出现的任何文字叙述、文档格式、插图、照片、方法、过程等内容，除另有特别注明外，其著作权或其他相关权利均属于中科海微（北京）科技有限公司。未经中科海微（北京）科技有限公司书面同意，任何人不得以任何方式或形式对本手册内的任何部分进行复制、摘录、备份、修改、传播、翻译成其它语言、将其全部或部分用于商业用途**。**

**免责条款**

本文档依据现有信息制作，其内容如有更改，恕不另行通知。中科海微（北京）科技有限公司在编写该文档的时候已尽最大努力保证其内容准确可靠，但中科海微（北京）科技有限公司不对本文档中的遗漏、不准确、或错误导致的损失和损害承担责任。

**技术支持与信息反馈**

如果您在使用我们的产品时遇到问题,或者您认为我们的产品有某些功能缺陷,请访问我们的官网www.haiwei.tech联系我们的客服,我们将为您解决问题和反馈;或者需要技术支持指导以及有任何宝贵意见，也请您通过官网或者电话联系我们：

**应用移植参考文档简介**

本文档简要介绍应用移植几个参考示例，以指导用户进行应用移植。

* MySQL 5.6.21 移植
* Apache 2.4.39 移植
* Dubbo 2.7.5 移植
* HAProxy 1.9.0 移植
* Lighttpd 1.4.53 移植
* Iok 2.1.3 移植
* Memcached 1.5.12 移植
* TENGINE 2.2.2 移植

**应用移植指南**

本文档主要适用于应用移植的用户。用户需要具备以下经验和技能：

* 具备Linux操作系统得基础知识
* 了解Linux命令行的基本操作

# 综述

## 1.1引言

应用移植开发文档旨在提供关于将应用程序从一个平台或环境迁移到另一个平台或环境的全面指导。随着移动操作系统、嵌入式系统以及云计算技术的不断发展，应用移植已成为软件开发过程中的重要环节。

应用的迁移主要是系统软件层面的移植（指将某个环境中的事物移动到其他环境，并仍可正常发挥作用）与迁移（指将某个事物的运作方式移动到另一个事物上），而调优则包含国产平台上主机硬件层、操作系统层与软件本身的参数优化。

## 1.2背景与目标

随着不同操作系统、硬件平台和软件架构的兴起，软件开发者面临着将现有应用程序扩展到新平台的需求。应用移植能够确保软件在不同环境下保持功能完整性和性能稳定性，从而满足更广泛用户的需求。

应用移植的主要目标包括：

1.确保应用程序在新平台上正常运行，并保持与原始平台相同的功能和性能。

2.最小化移植过程中的代码修改量，降低开发成本和时间。

3.提高应用程序的可维护性和可扩展性，以便在未来进行更多平台的扩展。

## 1.3关键步骤

1.需求分析：明确应用程序在新平台上的功能需求、性能要求和用户体验要求。

2.平台评估：对新平台的硬件、操作系统、软件架构和API进行评估，确保其与应用程序的兼容性。

3.代码修改：根据新平台的特点和要求，对应用程序的代码进行修改和优化，以适应新环境。

4.测试与验证：在新平台上对修改后的应用程序进行详细的测试和验证，确保其满足所有需求。

5.发布与维护：将应用程序发布到新平台，并提供持续的维护和更新服务。

## 1.4 面临的挑战

1.跨平台兼容性：不同平台之间的硬件、操作系统和软件架构差异可能导致应用程序无法直接运行。因此，开发者需要深入了解新平台的特点和要求，并进行相应的代码修改和优化。

2.性能优化：在新平台上运行的应用程序可能面临性能下降的问题。开发者需要针对新平台的特性进行性能优化，以确保应用程序在新平台上保持与原始平台相同的性能水平。

3.安全性与稳定性：在移植过程中，应用程序可能面临安全漏洞和稳定性问题。开发者需要采取相应的安全措施和稳定性测试来确保应用程序的安全性和稳定性。

4.开发成本与时间：应用移植可能涉及大量的代码修改和优化工作，导致开发成本和时间增加。因此，开发者需要在确保应用程序质量和性能的前提下，尽可能降低开发成本和时间。

# 操作系统软件的迁移与移植

## 2.1软件移植到国产操作系统的迁移过程

###### 2.1.1 软件迁移过程概述

对用户而言，在进行国产化的过程中，底层硬件与操作系统的变更，会涉及上层应用的全面适配与兼容需求，因此进行兼容性评估和适配工作，需要经过以下几个步骤：

1. 软件移植系统信息收集

信息收集的目的是进行难点分析及工作量评估，而需要收集的信息主要包括如下内容：

Ø 硬件信息：整机型号（服务器/终端）、外设、配件等信息；

Ø 软件信息：虚拟化/云平台、操作系统、中间件、数据库、浏览器、控件、第三方软件（开源/商业/自研）、应用架构（B/S、C/S）等信息；

Ø 运行环境：开发语言、开发框架、底层组件、依赖库等信息。

2. 软件迁移分析

在移植前的信息收集完成之后，需要对这些信息进行分析，分析内容主要包括：软件技术栈、开发框架、开发语言、依赖基础组件、依赖库、以及软件代码。

3. 迁移支撑

在软件迁移过程中，操作系统厂商可提供迁移支撑服务，包括：

Ø 组件编译：提供应用底层所需组件支撑；

Ø 技术咨询：提供系统支撑组件包的技术咨询；

Ø 问题处理及协助：针对能定位到具体模块的问题，可提供Demo进行复现分析

Ø 开发文档：提供系统相关API文档和技术文档的输出

4. 性能调优

软件迁移完成，除满足功能需求外，还要在目标系统中满足性能指标要求，通常的性能调优过程如下：

Ø 性能指标测试

Ø 性能指标单项优化

Ø 软件系统业务综合指标优化

5. 测试与认证

软件迁移完成，在软件规模上线应用前，除功能测试、性能测试，同时还要进行压力测试、稳定性测试，保证业务压力测试指标要求。

###### 2.1.2 迁移前的分析与评估

1. 技术框架

请首先确认产品在原系统上使用的技术框架，在目标系统上是否存在。

Ø 跨平台技术框架

如果采用了跨平台技术框架，比如QT，那么移植相对容易一些。

但需要注意，如果是C/C++技术框架，其SDK是否提供支持目标系统的预编译二进制文件，如果没有，需要首先将技术框架必要的源码进行编译。

如果是纯java或lua之类的解释型语言框架，则无此必要。

如果是混合技术框架，比如java框架中调用了C的类库，需要特别注意类库问题，具体请查看下一小节。

Ø 非跨平台技术框架

如果您的应用使用了WPF这一类特定平台深度绑定的技术框架，而这种技术框架并不支持目标系统，那就需要考虑进行替代开发。

替代开发可以分开来看：

图形库：图形库无法直接迁移，需要重新替代开发，推荐使用qt或electron。

逻辑库：如果使用通用C/C++逻辑库，比如gnu stdc++，可以尝试直接对逻辑库进行绑定开发。

2. 开发辅助工具

需要确认在Windows上的开发辅助工具，在目标系统是否存在，或有替代工具。

###### 2.1.3 分析软件包依赖库

1. 技术框架

请首先确认产品在原系统上使用的技术框架，在目标系统上是否存在。

Ø 跨平台技术框架

如果采用了跨平台技术框架，比如QT，那么移植相对容易一些。

但需要注意，如果是C/C++技术框架，其SDK是否提供支持目标系统的预编译二进制文件，如果没有，需要首先将技术框架必要的源码进行编译。

如果是纯java或lua之类的解释型语言框架，则无此必要。

如果是混合技术框架，比如java框架中调用了C的类库，需要特别注意类库问题，具体请查看下一小节。

Ø 非跨平台技术框架

如果您的应用使用了WPF这一类特定平台深度绑定的技术框架，而这种技术框架并不支持目标系统，那就需要考虑进行替代开发。

替代开发可以分开来看：

图形库：图形库无法直接迁移，需要重新替代开发，推荐使用qt或electron。

逻辑库：如果使用通用C/C++逻辑库，比如gnu stdc++，可以尝试直接对逻辑库进行绑定开发。

2. 开发辅助工具

需要确认在Windows上的开发辅助工具，在目标系统是否存在，或有替代工具。

###### 2.1.4 分析待迁移软件的源码文件

如果是C/C++源码，首先确认是否使用平台特有API，比如win32 API，需要替换为gnu libc/libc++兼容的跨平台api库，例如Direct2D可以考虑替换为SDL。

其次注意是否使用了汇编代码，由于主流国产操作系统支持多架构，尽量避免使用汇编，建议最低级的语言为C。

分析源码文件，如果是C++源码部分，可以考虑按照类逐个分析，提取核心类，优先尝试进行迁移。

一款软件中，一般可以分类为图形渲染引擎、数据库、逻辑脚本库、网络协议库、文字读取和渲染，登录界面等。

Ø 图形渲染引擎：Windows上最知名和常用的directX，主流国产操作系统并不支持，考虑替换为OpenGL或vulkan。

Ø 数据库：一般用作本地数据存储和校验，可以考虑替换为sqlite。

Ø 逻辑脚本库：可以考虑替换为LUA相关的技术框架。

Ø 网络协议库：分为通信库和加密库，加密库一般用的是openssl，这是跨平台的；主流国产操作系统有很多通信库，TCP、UDP支持的都有。

Ø 登录界面：Windows上用wpf的比较多，可以考虑替换为qt或electron。

Ø 字体读取和渲染：可以考虑用freetype替换。

特别注意：如果实在不知道C/C++代码如何迁移，请全部代码用标准C/C++语法和跨平台的类库进行编写。推荐使用C/C++ 11标准的规范，不要使用C/C++ 17或更新。

###### 2.1.5 平台迁移代码量和工作量

建议按照功能模块，通过分类，进行评估。

每个模块的迁移量和工作量，依据迁移模块数量、开发人员数量、学习时间等维度综合评估。如果有跨平台可重复使用的模块，迁移量会减少。

一般情况下，如果模块难以迁移，工作量可以按照该模块重写一遍的历史工时进行参考计算。

###### 2.1.6 迁移步骤及具体方法

★ 首先按照功能，对产品进行模块分类。

★ 其次，每一个模块，判断其使用的技术是否跨平台，不是跨平台的部分，挑选替代品继续进行评估。

★ 再次，优先迁移核心模块，确保软件最基础部分能够成功迁移，然后可以将其他模块逐渐迁移完善。

★ 优先在AMD64架构下进行迁移，后续可以针对其他CPU架构编译后，再进行测试和调整。

★ 最后，测试稳定性和性能，然后进行打包。

## 2.2 WINDOWS应用分析和移植方式

Windows的系统开发大约已经有二十多年的时间，已经积累了大量的应用开发工具，以及在此基础上开发的应用程序。Windows的应用开发技术也经历了很多变迁，从单机程序发展到C/S架构、B/S架构等。下面就这些应用使用到的语言和技术来分析对应的移植方法。

1. 单机版应用

单机版应用程序相对比较简单，一些老的MIS系统等均为单机版应用。

对于仍在使用的DOS应用程序，在国产操作系统下可以使用dosbox等虚拟环境来运行。

2. C/S架构的Client端

Client端编程使用的语言和技术都很广泛，针对应用程序所使用的语言不同，移植方法也不同。如果应用程序是由C++编写并且带有图形界面，可以考虑使用跨平台的开发环境QtCreator、Eclipse等，其中根据功能调用或者使用的技术对应主流国产操作系统下的解决方法如下：

Ø 网络接口：支持Berkeley套接字编程，提供C编程接口；提供EPOLL传输框架，同样具备高并发处理能力。

Ø 图形接口：可以替换为gtk，qt，wxWidgets，JAVA等。

Ø 音频接口：可以通过内核提供的系统调用访问声卡驱动，也可通过编程接口如alsa、sox等实现。

Ø 数据库：在国产操作系统上使用国产达梦、金仓或者开源的mysql等，接口有unixODBC，SQL等。

Ø 并发多任务：在主流国产操作系统上同样支持进程/线程概念，符合POSIX标准，支持IPC机制。

Ø 注册表：在主流国产操作系统上每个用户可以使用gconf机制，共用的环境变量可以根据系统配置文件管理。

Ø MFC，ATL，WTL：可以使用跨平台开发环境，部分需要根据功能编写Linux的插件或corba组件。其中MFC应用可以相对容易地迁移为wxWidgets。

Ø DirectX：可以使用第三方软件SDL、OpenGL等。

Ø COM、ActiveX：根据功能不同可以对应到corba、widget、插件等。

有些开发语言的某些版本如果有Linux下相应的开发工具可以对其代码进行重新编译。比如：

Ø Delphi/C++ builder：kylix。

Borland发布到kylix 3之后没有推进新的版本，所以kylix的使用只能针对Delphi 6及之前的版本。

Ø VB：可以使用.net迁移向导VB.net，或者使用Gambas（与VB较为类似）重新编写。

在VB编写中，如果没有使用ActiveX、没有和数据库的连接，可移植程度较高。

Ø Powerbuilder：Sybase曾经推出过Linux版本，并不能保证在相应的国产操作系统版本上能安装，没有对应版本的只能重写代码。

Ø .net：利用MONO技术迁移到主流国产操作系统平台。

总体来说，这类程序的可移植性较低。

3. B/S架构的Browser端

主流国产操作系统支持国产360、奇安信浏览器，也支持开源Firefox浏览器，使用中可能遇到以下的问题：

Ø HTML页面：部分程序页面不符合W3C标准，需要进行测试和少量迁移。

Ø Java Applet：一般浏览器需要安装相对应的支持库。

Ø 页面内嵌多媒体：主流国产操作系统支持多种媒体格式，如果网页没有调用具体的控件，则网页不需要修改，否则就需要代码调整。

Ø HTC组件：在主流国产操作系统下可利用moz-behavior特性进行迁移。

Ø ActiveX控件：只能重新开发为对应的浏览器插件。

## 2.3应用移植分类小结

从上面的分析可以知道，不同的Windows应用移植方法，可移植的程度都不同。从移植的程度来说，分为以下几类：

1. 完全移植

在主流国产操作系统下均以本地代码运行，应用程序安全可靠并且运行效率最高。主要包括：

Ø 跨平台的脚本语言（如PHP、Python等）编写的应用程序，几乎无需修改源代码即可在目标系统下运行。

Ø 跨平台的编译型语言（如Java、C++等）编写的应用程序，几乎无需修改源代码，在目标系统下重新编译即可运行。

Ø VC、C#、.net等非跨平台开发语言编译的应用程序，在目标系统下需重新开发，成本较高。

2. 虚拟化

使用虚拟化技术是在国产操作系统上再运行一个Windows，无需移植，保证完美运行。虽然虚拟机是效率损失最大的，但是随着硬件虚拟化技术的出现使得效率损失减少，尤其在多核CPU出现后，这种效率损失几乎忽略不计。

不常使用的应用程序，可以应用这种方式，例如某些带ActiveX控件的网站。

在实际移植过程中，要考虑性能、稳定性和移植代价之间的平衡。根据应用、使用环境不同，选择相应的移植方式。

## 2.4应用移植技术难点

1. 硬件适配

硬件适配主要涉及板卡移植和外设移植，由于硬件设计细节由硬件开发商掌握，操作系统厂商仅能根据公开的技术资料开发驱动。一直以来，Windows操作系统都占据市场主导地位，硬件厂商通常自主为其提供驱动，所以在Windows上的硬件能力发挥稳定；而国产操作系统平台暂时不具备市场号召力，只能根据部分硬件资料，参考公开可获得的驱动程序源代码，进行驱动开发和移植工作，通常新驱动都无法完全发挥硬件的全部能力。如果没有相关硬件资料和驱动源代码，通常必须重新选型。

2. 运行支撑库

无论是Windows还是国产操作系统都提供一套运行库，提供应用软件调用，其中尤以3D图形库和多媒体库的需求比较突出。主流国产操作系统提供了一套与Linux兼容的基于XWindow系统的标准图形库，与Unix平台的XWindow接口基本一致。主流国产操作系统所提供的3D图形库主要是来自开源社区的Mesa3D，它是目前唯一免费和开源的OpenGL实现。使用广泛的Windows平台下Direct3D引擎目前暂无对应的适配国产软硬件平台的兼容版本。不过，可以利用开源社区Gallium3D项目。

多媒体库主要提供多媒体格式的编解码器，由于公开资料中相关资源较为丰富，国产操作系统对此的支持较好，但对于原先基于国外操作系统开发的软件，其调用接口将需要调整。

其他的运行库还包括线程库、数学库、字符串处理库等，POSIX都对其进行了较为详细的规范。

3. 字体字库

中文应用的字体字库问题通常会给应用迁移过程带来麻烦，因为主流国产操作系统所能支持的字体及其缺省采用的编码和字体，与国外平台均有所不同。

由于版权关系，Windows平台上的“宋体”、“楷体”、“黑体”、“仿宋”等常用字体文件只能采用字形相似的其他字体文件代替，目前因主流国产操作系统提供了方正和华文字库，基本实现了宋楷黑仿主要字体的显示。为解决字体名不同的问题，主流国产操作系统进行了字体映射配置，应用软件沿用Windows平台的字体名时，也能在主流国产操作系统上以相同字体显示。

但部分Windows平台上常用的字体，则没有相应的可获得授权的可选字体，如微软雅黑字体等，即使采用映射的方式选择其他字体，在字形上也会有所区别，对应用软件而言，就有可能在对齐、排版上带来一些问题。

Windows系统缺省编码通常是GB18030（中文系统），主流国产操作系统缺省编码基本统一为UTF8，在与中文编码相关的应用中，如果未采用标准的国际化处理规程，则可能在应用迁移中带来问题。主流国产操作系统提供了一整套国际化和本地化编程接口，应用迁移时应遵循该接口对字符串进行处理，以避免引入编码错乱问题。

## 2.5迁移中遇到的常见问题及解决

1. 什么是依赖，怎么安装依赖？

所谓的依赖，指的是操作系统在线仓库内的软件包，大部分是动态链接库和头文件，可以避免自行编译，节省开发和部署成本。

2. 怎么调试？

调试工具有很多，比如strace和gdb，strace详细记录二进制文件的运行过程，gdb可以排查二进制文件的运行断点。

建议开发过程中编译带debug信息的二进制文件，方便调试。

3. 开发中需要的xx软件没有，怎么办？

如果是软件的组成部分，仓库中并没有的情况，优先考虑自行编译，也可以联系软件作者提供协助；主流国产操作系统厂商可以分派工程师进行在线协助。

# 服务器上应用软件移植与国产化迁移

## 3.1应用迁移基本过程

一个应用从X86平台到其他平台典型的迁移过程包括如下三个步骤：

1、技术分析阶段：准备调试编译环境，软件栈分析。

2、编译迁移阶段：搭建编译调试环境，重写汇编代码，修改编译选项，代码编译。

3、功能验证：搭建功能测试环境，全量功能验证。

按照程序源代码到机器指令之间翻译方式的不同，高级语言通常可以分为两类：一类是编译翻译，一类是解释翻译，分别对应着编译型语言和解释型语言。

###### 3.1.1 编译型语言应用迁移

典型的如C、C++语言，都属于编译型语言。C/C++编译好的程序是机器指令，由操作系统加载到存储器（一般为内存）后由CPU直接执行。

基于编译型语言开发的应用程序，例如C/C++语言应用程序，其编译后得到可执行程序，可执行程序执行时依赖的指令是CPU架构相关的。因此，基于X86架构编译的C/C+ +语言应用程序，从X86平台迁移到国产平台时，必须使用源代码、经过重新翻译才能运行。

C/C++移植流程过程如下。

1. 获取应用程序源代码。

2. 如果源代码涉及到X86汇编的代码需要使用国产架构汇编重写；汇编程序和C/C++源代码中内联汇编需要重写。原因国产架构指令集不兼容，跨平台执行汇编指令会触发指令异常。

3. 根据使用国产平台编译环境，使用相应的编译工具，编译应用程序。 在国产架构编译环境上，编译方法与X86服务器一致，可使用make、cmake、autoconfig等工具进行编译。对于多数开源软件，执行./configure; make; make install。

4. 如遇到开源项目库不支持国产架构出现编译失败（这种情况极少，一般发生在较旧的项目代码），解决方法包括：修改代码，寻求替代库等。

5. 编译时提示代码错误，可能需要根据平台差异修改部分代码，例如平台类型相关宏定义。

6. 应用程序部署：安装部署、设置系统启动脚本等操作方法与X86服务器一致。

###### 3.1.2 解释型语言应用迁移

基于解释型语言开发的应用程序，与CPU架构/指令集相关的差异由语言运行环境（即虚拟机或者解释器）进行屏蔽/解决，例如Java、Python、PHP等。Java/Python编译好的程序是平台无关的字节码，由虚拟机解释执行，虚拟机完成平台差异的屏蔽。

将这类应用程序移植到国产架构平台，通常需要两步：

大多数解释型语言开发的应用程序，无需修改即可在国产架构平台上运行。

Java应用程序jar包内，可能包含基于C/C++语言开发的so库文件，这类so库需要移植编译，移植so库文件后，重新打包jar包，移植编译so库遇到的问题可以参考编译型语言应用迁移的方法解决。

1. 安装语言执行环境。

对于Java是Java虚拟机，对于PHP是PHP语言解释器，对于其他解释型语言，例如 Python、Ruby、Go等，请安装各语言解释器。各语言虚拟机或解释器，需要国产架构版本，其中大多已被操作系统预置，请使用操作系统包管理器检查。如果系统未预置或者预置版本不符合要求，请使用源代码或包管理器进行安装，并参考本文档相关节内容进行移植。

以UOS使用包管理器安装JAVA虚拟机为例：

yum install java-1.8.0-openjdk

2. 执行应用程序。

将应用程序部署在国产架构服务器上，无需修改和重新编译，按照与X86一致的方式部署和运行应用程序。

###### 3.1.3 制定整体迁移策略

建议在做应用迁移前，先制定好整体迁移策略：

收集软件栈信息。

作为迁移计划和迁移策略制定的输入，软件栈信息包括当前使用的硬件信息，如芯片，服务器型号。软件信息包含，如OS，数据库、中间件、编译软件，自研软件，开源软件，商用软件等。

制定具体的迁移策略和计划。

根据收集好的软件栈情况，针对不同类型的软件，制定具体的迁移策略和实施计划。

## 3.2应用迁移具体方法

###### 3.2.1 编译工具

开源编译工具

rpm及deb格式的Linux应用软件二进制安装包，是与操作系统版本及CPU架构相关的，因此为X86服务器编译生成的二进制安装包，是无法在国产架构服务器上使用的，需要使用源代码，通过编译工具软件，重新编译为国产架构服务器版本的安装包。对于不同编译型语言开发的软件，应选择对应的编译器进行二进制编译。

编译器选择如下：

基于GO开发的软件，推荐使用Golang编译器1.9以上版本（1.8版本在部分场景下存在性能问题）。

基于Fortran开发的软件，推荐使用gFortran编译器。

基于C/C++开发的软件，推荐使用GCC。

由于不同版本的编译器对语言标准的支持存在差异，故需检查软件源码所使用的语言标准，选择支持该语言标准的版本。

###### 3.2.2 编译类型

本地编译

本地编译本地运行，即在直接在国产架构环境上利用gcc等工具，编译出来的可运行的程序，编译完成后直接测试运行。

1. ./configure 执行编译配置。

configure是一个shell脚本，用来检测你的安装平台的目标特征，它可以自动设定源程序以符合各种不同平台上Linux/Unix系统的特性，并且根据系统叁数及环境产生合适的Makefile编译配置文件或是C的头文件(header file)，让源程序可以正确地在这些不同的平台上被编译连接。

2. make 执行编译

它从Makefile中读取指令，然后编译。

3. make install执行安装

它也从Makefile中读取指令，安装到指定的位置。

###### 3.2.3 C/C++代码移植参考

1. -m64 编译选项

-m64是X86 64位应用编译选项，m64选项设置int为32 bits及long指针为64 bits，为 AMD的X86 64架构生成代码，在国产架构平台无法支持，编译时报错。

解决方法：

将ARM64平台对应的编译选项设置为“-mabi=lp64”

将LoongArch64平台对应的编译选项设置为“-mabi=lp64d”

2. configure配置与makefile构建

检查软件源代码是否包含Makefile文件。当软件源代码不包含Makefile文件时，用户可利用自动编译工具，例如Autoconf、Automake和Build工具构建Makefile文件。解压源代码后，进入代码目录检查代码结构，查看是否存在configure脚本，再执行 configure命令：

./configure

如果不存在configure脚本，可通过源码中的自动化脚本生成configure文件。

如果执行过程中存在以下问题，对应的解决方案如下：

configure文件不包括国产架构选项。

修改configure文件，增加适配国产架构编译选项。

无法识别编译类型。

执行configure脚本时，提示无法识别编译类型，如下所示：

configure: error: cannot guess build type: you must specify one

configure: error: ./configure failed for libltdl

需要在config.guess和config.sub里增加国产架构的支持。

3. -march与-mtune编译参数不兼容

Makefile中包含“-march”与-mtune编译参数，国产架构平台上编译提示参数不兼容。

解决方法：

兼容的调整后参数为：

ARM平台：

-march=armv8.1-a -tune=cortex-a72

参数设置指导请参考：https://gcc.gnu.org/onlinedocs/gcc/ARM-Options.html

Loongarch平台：

-march=loongarch64-tune=la464

参数设置指导请参考：https://gcc.gnu.org/onlinedocs/gcc/LoongArch-Options.html

4. 代码中汇编指令需要重写

在追求更高性能或者调用CPU架构相关的特殊指令时，C/C++代码中可以\_\_asm\_\_或asm关键字修饰，直接以汇编指令代码方式实现某些运算功能，这样的汇编代码我们称为“内联汇编”。

由于指令集的差异，X86和国产架构的汇编完全不一样，所以只要涉及X86内联汇编的地方都要移植修改，针对国产架构平台进行相应修改或重写。

内联汇编的写法有两种形式，“asm”和“\_\_asm\_\_”，其中asm关键字是GNU扩展，如果编译时使用-ansi或者-std选项，需要使用\_\_asm\_\_。 volatile和\_\_volatile\_\_是防止编译器优化，通常我们都添加了volatile描述。

asm [volatile] ( AssemblerTemplate

: OutputOperands

[ : InputOperands

[ : Clobbers ] ])

汇编/指令可参考资料：

1) X86指令集手册：https://www.intel.cn/content/www/cn/zh/architecture-and-technology/64-ia-32-architectures-software-developer-vol-2a-manual.html

2) ARM指令集手册：https://developer.ARM.com/docs/100076/0100

3) LoongArch指令集手册：https://www.loongson.cn/uploads/images/2023102309132647981.%E9%BE%99%E8%8A%AF%E6%9E%B6%E6%9E%84%E5%8F%82%E8%80%83%E6%89%8B%E5%86%8C%E5%8D%B7%E4%B8%80\_r1p10.pdf

4) 内联汇编规则：[https://gcc.gnu.org/onlinedocs/gcc/Using-Assembly-Language-with-C.html#Using-Assembly-Language-with-C](https://gcc.gnu.org/onlinedocs/gcc/Using-Assembly-Language-with-C.html" \l "Using-Assembly-Language-with-C)

###### 3.2.4 java代码移植参考

Java开源软件Maven仓库软件迁移

Maven是Apache下的一个纯Java开发的开源项目，基于项目对象模型（缩写：POM），可以对 Java项目进行构建、依赖管理。通过Maven工具和项目配置能自动化完成对依赖组件的下载和参与构建过程，其中Maven仓的正确配置和使用是构建迁移关键因素。

Maven仓库搜索顺序

1) 本地仓库搜索

本地仓库找到，直接返回

本地仓库没有找到，去远程仓库搜索

2) 远程仓库搜索

没有配置远程仓库，去中央仓库搜索

远程仓库找到，下载到本地仓库

远程仓库没有找到，搜索下一个远程仓，依次类推，如果所有远程仓都未找到，搜索中央仓

3) 中央仓库搜索

中央仓库找到，下载到本地仓库

中央仓库没有找到，前台打印错误信息

Maven软件构建迁移关键流程：将X86依赖文件替换成国产架构依赖文件，重新构建，直到不包含X86依赖。

###### 3.2.5 python代码移植参考

移植基本步骤：

步骤 1 准备Python运行环境：使用国产架构版操作系统自带的Python。

步骤 2 设置环境变量：设置Python运行时的环境变量

步骤 3 【可选】编译生成pyc文件（字节码）

步骤 4 【可选】移植外部依赖库

步骤 5 【可选】更新代码中新的外部依赖库调用代码

步骤 6 测试：执行Python源码程序

###### 3.2.6 go代码移植参考

1. Go编译系统迁移

Go编译系统迁移，首先下载最新的国产架构 golang。

下载地址为：https://golang.org/dl/，国内下载地址：https://studygolang.com/dl。

将下载的二进制包解压至安装目录，这里以/usr/local为例，

通常需要设置3个环境变量GOROOT（软件包安装在系统上的位置），GOPATH（工作目录的位置、需要根据实际代码目录设置）和PATH（设置Go的bin目录），在.profile或/etc/profile文件设置环境变量如下：

echo "export GOROOT=/usr/local/go" >> ~/.profile

echo "export PATH=$GOPATH/bin:$GOROOT/bin:$PATH" >> ~/.profile

source ~/.profile

2. Go汇编迁移

一般情况下，在国产架构平台上搭建好编译系统就能直接编译运行代码，但对于存在Go汇编的情况，即\*.s汇编文件，需额外进行迁移。

现象描述：

源码包含X86 Go汇编时，编译报错：missing function body。

Go中寄存器包括通用寄存器和伪寄存器，以下重点介绍伪寄存器。

伪寄存器

X86平台下Go存在四种伪寄存器FP、PC、SB、SP，实际是对内存位置的一个引用。

FP（frame pointer）：帧指针，保存参数和本地变量，用来标识函数参数、返回值。

SP（stack pointer）：栈指针，存放栈的偏移地址，它指向本地栈帧的顶部。

SB（static base）：静态基指针，用来表示全局的变量或者函数。

PC（Program counter）：程序指针，即处理器中常见的PC寄存器，存储需执行指令地址。

汇编代码迁移

Go编译器会输出一种抽象的汇编代码，这种汇编并不对应某种真实的硬件架构。Go汇编器使用这种伪汇编，为目标硬件生成具体的二进制文件。伪汇编的优点在于更易将Go移植到新的架构上，但即使是伪汇编，X86平台与国产架构平台还是存在一些差异。

###### 3.2.7 其他代码移植

1. 开源类软件

其他小语种代码移植：

编译型：go

解释型：lua、ruby、php、perl、typescript、javascript

编译型：源码需要重新编译

解释型：源码不用编译，安装解释器即可

解释器搭建，程序运行环境准备方法:

1. 操作系统自带解释器软件，直接安装即可使用；

2. 从官网上下载支持A国产架构的解释器软件包，直接解压安装使用；

3. 从官网下载解释器源码，编译后使用。

2. 闭源类软件

2.1、 获取国产架构版本：

联系商用软件提供商，获取与国产架构兼容的软件版本，重新部署使用，迁移或置换商用license。

2.2、 无ARM或LoongArch版本：

原商用软件提供商无法提供国产架构版本的，可以替换为其他支持国产架构软件的替代方案。

# MySQL 5.7.21 移植指南

## 4.1简要介绍

本文主要用于指导在SeawayEdge操作系统上部署 mysql 数据库。MySQL 是一款安全、跨平台、高效的，并与 PHP、Java 等主流编程语言紧密结合的数据库系统。 本案例使用 x86\_64 架构虚拟机，通过评估工具 x2openEuler 评估 MySQL 5.7.21 软件移植到 openEuler 操作系统的兼容性，再实施数据搬迁。建议使用版本为 MySQL 5.7.21。说明: 本文档适用于 MySQL 5.7.21，其他版本的 MySQL 移植步骤也可参考本文档。

## 4.2 案例环境

OS 环境

| 软件 | 版本 | 备注 |
| --- | --- | --- |
| OS | Centos 7.6.1810 | 当前mysql集群服务器 |
| OS | SeawayEdge-v3.3 | 迁移目标服务器 |

软件包

| 软件 | 版本 |
| --- | --- |
| Mysql5 | 5.7.21 |
| Mysql5-common | 5.7.21 |
| Mysql5-embedded | 5.7.21 |
| Mysql5-embedded-devel | 5.7.21 |
| Mysql5-errmsg | 5.7.21 |
| Mysql5-libs | 5.7.21 |
| Mysql5-server | 5.7.21 |
| Mysql5-test | 5.7.21 |

## 4.3 案例环境

openEuler 社区提供了 x2openEuler 工具 ，针对已经编译好的二进制程序，进行主要完成软件包、接口级评估，明确应用软件是否需要移植适配，是否有依赖的软件包待引入；同时评估软件调用的接口原型在两个系统中是否有差异。注：已经编译好的二进制程序，难以保障全部兼容新 OS，严重时会引发才内存风险，往往这种问题很难通过验证的方式识别出来，迁移前针对软件兼容性评估尤为重要。

* 获取 mysql 的 RPM 包并解压到/opt/mysql 目录下

wget -P /opt <https://downloads.mysql.com/archives/get/p/23/file/mysql-5.7.21-1.el7.x86_64.rpm-bundle.tar>

cd /opt/mkdir mysqltar -xf mysql-5.7.21-1.el7.x86\_64.rpm-bundle.tar -C mysql

* 下载 x2openEuler 工具到/opt/mysql

下载指引：https://www.openeuler.org/zh/other/migration/

* 部署工具

cd /opt/mysqlrpm -ivh x2openEuler-2.0.0-1.x86\_64.rpm

su x2openEulerx2openEuler redis-db -init

x2openEuler init source\_centos7.6-openEuler20.03-LTS-SP1.tar.gz

* 扫描 mysql

x2openEuler scan /opt/mysql/注意要分析的移植文件需要有能够让x2openEuler用户可以读取的权限扫描完成后会在/opt/x2openEuler/output目录生成html格式的报告

## 4.4 性能结果评估

软件兼容性评估报告分三块内容展示软件兼容性，分别是依赖包兼容性、C/C++接口兼容性、java 接口兼容性，依赖包兼容性反映了软件包安装过程中的直接依赖，非 100%表明无法正确安装；接口兼容性反映的是单个软件运行过程中对其他软件包、动态库或系统接口的调用变化，非 100%表明在某个功能调用时可能会触发异常，未调用到时可能表现正常；部分结果建议人工复核，最终软件包使用建优先级建议 openEuler 已移植包>openEuler 上人工重编译包>centos 软件包。

* 报告分析

打开html报告，逐行分析，得出结论：在openEuler上直接使用centos的mysql包存在风险，风险如下：1个待确认接口表明mysql系列软件包会调用到libaio.so.1.0.1，其函数参数数量从4变为5，直接影响了功能，在某个功能调用时可能会触发异常；另外，报告显示需要确认3个依赖软件包，经过人工确认属于mysql系列包自闭环的依赖，故软件包安装无影响



* 分级结果分析

建议：由于函数调用风险，建议直接使用在openEuler官方编译移植过的mysql-5.7.21系列软件包https://repo.openeuler.org/openEuler-20.03-LTS-SP1/everything/x86\_64/Packages/mysql5-5.7.21-3.oe1.x86\_64.rpmhttps://repo.openeuler.org/openEuler-20.03-LTS-SP1/everything/x86\_64/Packages/mariadb-common-10.3.9-9.oe1.x86\_64.rpmhttps://repo.openeuler.org/openEuler-20.03-LTS-SP1/everything/x86\_64/Packages/mysql5-common-5.7.21-3.oe1.x86\_64.rpmhttps://repo.openeuler.org/openEuler-20.03-LTS-SP1/everything/x86\_64/Packages/mysql5-server-5.7.21-3.oe1.x86\_64.rpmhttps://repo.openeuler.org/openEuler-20.03-LTS-SP1/everything/x86\_64/Packages/mysql5-errmsg-5.7.21-3.oe1.x86\_64.rpmhttps://repo.openeuler.org/openEuler-20.03-LTS-SP1/everything/x86\_64/Packages/mecab-0.996-2.oe1.x86\_64.rpm

## 4.5 安装数据库 mysql

* 安装 mariadb 及 mysql 相关服务

rpm -ivh mysql5-5.7.21-3.oe1.x86\_64.rpm mariadb-common-10.3.9-9.oe1.x86\_64.rpm mysql5-common-5.7.21-3.oe1.x86\_64.rpm mysql5-server-5.7.21-3.oe1.x86\_64.rpm mecab-0.996-2.oe1.x86\_64.rpm mysql5-errmsg-5.7.21-3.oe1.x86\_64.rpm

* 启动 mysql

systemctl start mysqld

* mysql 状态查询

systemctl status mysqld状态为 running 则启动成功：root@vm-2p32g.2288hv5-2s44p-384g--b5-0 ~# systemctl status mysqld

● mysqld.service - MySQL 5.7 database server Loaded: loaded (/usr/lib/systemd/system/mysqld.service; disabled; vendor preset: disabled) Active: active (running) since Thu 2021-09-09 10:23:25 CST; 1 day 4h ago Process: 103715 ExecStartPre=/usr/libexec/mysql-check-socket (code=exited, status=0/SUCCESS) Process: 103738 ExecStartPre=/usr/libexec/mysql-prepare-db-dir mysqld.service (code=exited, sta> Process: 103773 ExecStart=/usr/libexec/mysqld --daemonize --basedir=/usr --pid-file=/run/mysqld> Process: 103803 ExecStartPost=/usr/libexec/mysql-check-upgrade (code=exited, status=0/SUCCESS) Main PID: 103775 (mysqld) Tasks: 37 Memory: 188.4M CGroup: /system.slice/mysqld.service​ └─103775 /usr/libexec/mysqld --daemonize --basedir=/usr --pid-file=/run/mysqld/mysqld.

* 登录并修改默认密码

mysql -uroot -pa. 默认没有密码，按回车即可登录。

root@vm-2p32g.2288hv5-2s44p-384g--b5-0 /# mysql -uroot -pEnter password:Welcome to the MySQL monitor. Commands end with ; or \g.Your MySQL connection id is 2Server version: 5.7.21 MySQL Community Server (GPL)Copyright (c) 2000, 2018, Oracle and/or its affiliates. All rights reserved.Oracle is a registered trademark of Oracle Corporation and/or itsaffiliates. Other names may be trademarks of their respectiveowners.Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.mysql>

b. 设置密码。alter user 'user'@'localhost' identified by 'passward';

mysql> alter user 'root'@'localhost' identified by '123456';Query OK, 0 rows affected (0.00 sec)mysql> flush privileges;Query OK, 0 rows affected (0.00 sec)mysql>

* 验证密码

退出后重新登录，查看密码是否修改成功。

root@vm-2p32g.2288hv5-2s44p-384g--b5-0 /# mysql -uroot -pEnter password:Welcome to the MySQL monitor. Commands end with ; or \g.Your MySQL connection id is 3Server version: 5.7.21 MySQL Community Server (GPL)Copyright (c) 2000, 2018, Oracle and/or its affiliates. All rights reserved.Oracle is a registered trademark of Oracle Corporation and/or itsaffiliates. Other names may be trademarks of their respectiveowners.Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.mysql>

# 第五章 Apache 2.4.39 移植

Apache HTTP Server（简称 Apache）是 Apache 软件基金会的一个开放源码的网页服务器，可以在大多数计算机操作系统中运行，由于其多平台和安全性被广泛使用，是最流行的 Web 服务器端软件之一。它快速、可靠并且可通过简单的 API 扩展，将 Perl/Python 等解释器编译到服务器中。开发语言：C一句话描述：Web 服务器

## 5.1 环境要求

| 软件 | 版本 |
| --- | --- |
| SeawayEdge | SeawayEdge-v3.3 |
| Kernel | 4.19 |

## 5.2配置编译环境

安装开发包yum update yum install gcc gcc-c++安装依赖库yum install apr-devel.aarch64 apr-util-devel.aarch64 pcre-devel.aarch64 -y获取源码下载地址： <http://archive.apache.org/dist/httpd/httpd-2.4.39.tar.gz>

* 配置安装

tar xzvf httpd-2.4.39.tar.gz

* 修改源码配置

cd httpd-2.4.39vi ./build/config.sub找到下面两行进行修改,添加aarch64：| x86 | xc16x | xstormy16 | xtensa \=>| x86 | aarch64 | xc16x | xstormy16 | xtensa \

* 编译安装

./configure --host=aarch64 --build=aarch64make -j4make install

## 5.3参数配置

* 修改 httpd.conf

vi /usr/local/apache2/conf/httpd.conf89行取消注释： LoadModule socache\_shmcb\_module modules/mod\_socache\_shmcb.so196行取消注释，修改为当前服务器IP #ServerName www.example.com:80 => ServerName local\_server\_ip:80461行取消注释 Include conf/extra/httpd-mpm.conf488行取消注释 Include conf/extra/httpd-default.conf

* 修改 httpd-default.conf

vi /usr/local/apache2/conf/extra/httpd-default.conf23行值修改为0 MaxKeepAliveRequests 0

## 5.4验证

apache启动命令: /usr/local/apache2/bin/httpd -f /usr/local/apache2/conf/httpd.conf -k startapache停止命令: /usr/local/apache2/bin/httpd -f /usr/local/apache2/conf/httpd.conf -k stop查看服务进程: ps -ef |grep httpd

# 第六章 Dubbo 2.6.8 移植

Dubbo 是阿里巴巴公司开源的一个高性能优秀的服务框架，使得应用可通过高性能的 RPC（远程过程调用）实现服务的输出和输入功能，可以和 Spring 框架无缝集成。简单地说，Dubbo 是一个基于 Spring 的 RPC 框架，能够实现服务的远程调用、服务的治理。

## 6.1 环境要求

| 软件 | 版本 |
| --- | --- |
| SeawayEdge | SeawayEdge-v3.3 |
| Kernel | 4.19 |

## 6.2配置编译环境

* 配置DNS服务器

# cat /etc/resolv.confnameserver 114.114.114.114nameserver 8.8.8.8

* 安装依赖包

下载并安装依赖包yum install java-1.8.0\* tcl git gcc gcc-c++ make cmake libtool autoconf automake -y

查看 Java 版本[root@localhost ~]# java -versionopenjdk version "1.8.0\_272"OpenJDK Runtime Environment Bisheng (build 1.8.0\_272-b10)OpenJDK 64-Bit Server VM Bisheng (build 25.272-b10, mixed mode)

* 安装Maven

下载 Maven 安装包wget <https://archive.apache.org/dist/maven/maven-3/3.6.3/binaries/apache-maven-3.6.3-bin.tar.gz>

解压安装包到指定目录tar -zxvf apache-maven-3.6.3-bin.tar.gz -C /opt/

配置 Maven 环境变量

1. 在“/etc/profile” 文件末尾增加 Maven 路径echo "MAVEN\_HOME=/opt/apache-maven-3.6.3/" >> /etc/profileecho "export PATH=$MAVEN\_HOME/bin:$PATH" >> /etc/profile
2. 使修改的环境变量生效。

source /etc/profile

检查配置是否生效。

[root@localhost ~]# mvn -vApache Maven 3.6.3 (cecedd343002696d0abb50b32b541b8a6ba2883f)Maven home: /opt/apache-maven-3.6.3Java version: 1.8.0\_272, vendor: Bisheng, runtime: /usr/lib/jvm/java-1.8.0-openjdk-1.8.0.272.b10-7.oe1.aarch64/jreDefault locale: en\_US, platform encoding: UTF-8OS name: "linux", version: "4.19.90-2012.4.0.0053.oe1.aarch64", arch: "aarch64", family: "unix"

修改 Maven 配置文件中的本地仓、远程仓、代理等

配置文件路径：“/opt/apache-maven-3.6.3/conf/settings.xml”。配置网络代理，其中 host，port，username，password 需要根据当前环境修改：

<proxies> <proxy> <id>my-proxy</id> <active>true</active> <protocol>https</protocol> <host>代理服务器网址</host> <port>代理服务器端口</port> <username>用户名</username> <password>密码</password> <nonProxyHosts>local.net|some.host.com</nonProxyHosts> </proxy> <proxy> <id>my-proxy1</id> <active>true</active> <protocol>http</protocol> <host>代理服务器网址</host> <port>代理服务器端口</port> <username>用户名</username> <password>密码</password> <nonProxyHosts>local.net|some.host.com</nonProxyHosts> </proxy></proxies>

配置远程仓库：

<mirrors> <mirror> <id>huaweicloud</id> <mirrorOf>\*</mirrorOf> <url>https://mirrors.huaweicloud.com/repository/maven/</url> </mirror></mirrors>

## 6.3编译Dubbo2.6.8

* 获取源码

mkdir /home/Dubbo && cd /home/Dubbo && wget https://github.com/apache/dubbo/archive/dubbo-2.6.8.tar.gz&& tar -xvf dubbo-2.6.8.tar.gz

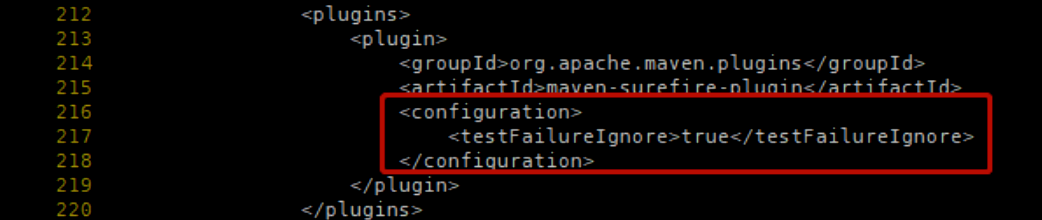
* 编译 dubbo-rpc-redis 模块

mvn install

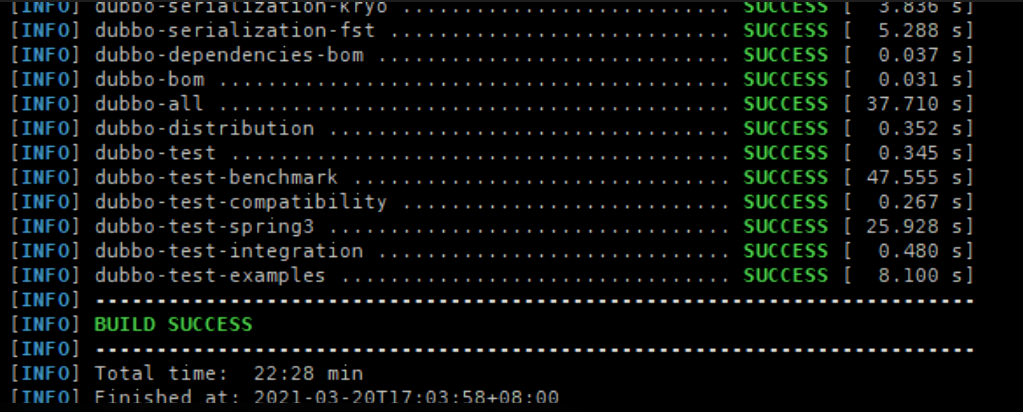
若窗口显示 BUILD SUCCESS，则 dubbo-rpc-redis 模块编译成功。

* 编译 Dubbo 2.6.8

/home/Dubbo/dubbo-dubbo-2.6.8/pom.xml 文件 552 行后增加如下内容：



若显示 BUILD SUCCESS，则 Dubbo 2.6.8 编译成功。



编译完成后的 dubbo-2.6.8.jar 包保存在“all/target”目录。

# 第七章 HAProxy 1.9.0 移植

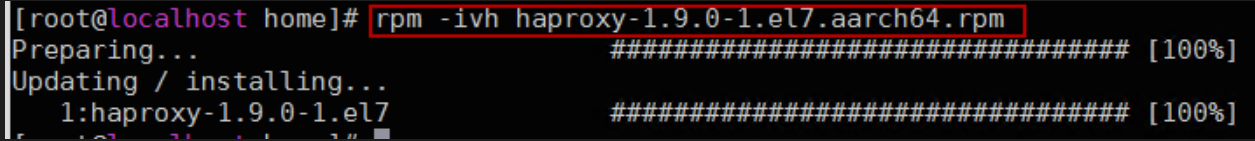
HAProxy 是一个使用 C 语言编写的自由及开放源代码软件，其提供高可用性、负载均衡，以及基于 TCP 和 HTTP 的应用程序代理，支持虚拟主机，它是免费、快速并且可靠的一种解决方案。 开发语言：C一句话描述：Web 负载均衡

## 7.1 镜像站 RPM 方式安装

若您的服务器可以访问网络，执行 wget https://mirrors.huaweicloud.com/kunpeng/yum/el/7/aarch64/Packages/web/haproxy-1.9.0-1.el7.aarch64.rpm 命令下载 RPM 包。否则，请访问 https://mirrors.huaweicloud.com/kunpeng/yum/el/7/aarch64/Packages/web/haproxy-1.9.0-1.el7.aarch64.rpm 下载 RPM 包并复制到服务器“/home”目录。说明： 镜像站中的 RPM 包都是通过开源代码编译打包而成，然后将其上传到镜像站。以本地下载 RPM 包并上传到服务器为例说明安装操作

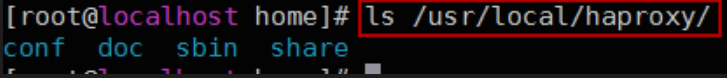
* 安装 HAProxy

rpm -ivh haproxy-1.9.0-1.el7.aarch64.rpm



* 查看安装目录

ls /usr/local/haproxy



## 7.2运行和验证

* 配置参数

1. 打开 option-http\_proxy.cfg 文件

vi /usr/local/haproxy/conf/option-http\_proxy.cfg

1. 修改 option-http\_proxy.cfg 为如下内容后，保存并退出文件

globalmaxconn 20000log 127.0.0.1 local0 infouid 0gid 0chroot /usr/local/haproxynbproc 4daemondefaultsmode httpretries 3timeout connect 10stimeout client 20stimeout server 30stimeout check 2sfrontend test-proxybind \*:80mode httplog globaldefault\_backend test-proxy-srvbackend test-proxy-srvbalance roundrobinoption http-server-closeoption httpchk GET /index.htmlhttp-check expect status 200server web1 IP1:PORT1 weight 3server web2 IP2:PORT2 weight 3

配置文件示例参数说明见下表。

| 参数 | 说明 |
| --- | --- |
| global | - |
| maxconn 20000 | 默认最大连接数。 |
| log 127.0.0.1 local0 info | 定义日志输出设备，info 表示日志级别。 |
| uid 0 | 运行 HAProxy 的用户 id。 |
| gid 0 | 运行 HAProxy 的用户组 id。 |
| chroot /usr/local/haproxy | chroot 运行路径。 |
| nbproc 4 | 设置进程数量。 |
| daemon | 以后台形式运行 HAProxy。 |
| defaults | - |
| mode http | 所处理的类别（7 层代理 http，4 层代理 tcp）。 |
| retries 3 | 设置连接后端服务器的失败重试次数，超过此值标记后端服务器为不可用。 |
| timeout connect 10s | HAProxy 与后端服务器建立连接的最长等待时间。 |
| timeout client 20s | 和客户端保持空闲连接的超时时间。 |
| timeout server 30s | 和服务端保持空闲连接的超时时间。 |
| timeout check 2s | 对服务端的检测超时时间。 |
| frontend test-proxy | - |
| bind \*:80 | 定义一个或几个监测的套接字，\*表示当前所有的 ipv4 地址。 |
| mode http | 所处理的类别（7 层代理 http，4 层代理 tcp）。 |
| log global | 继承 global 中 log 的定义。 |
| default\_backend test-proxy-srv | 指定默认的后端服务器池。 |
| backend test-proxy-srv | - |
| balance roundrobin | 指定负载均衡算法 roundrobin 是基于权重进行轮询的算法，适用于服务器性能均匀时。 |
| option http-server- | 当开启长连接时，应该开启此项。 |
| option httpchk GET /index.htmlhttp-check expect status 200 | 启用 HTTP 的服务状态检测（健康检查）。检查返回的状态码，接受不到 200 就不给后端 server 调度。 |
| server web1IP1:PORT1 weight 3 server web2 IP2:PORT2 weight 3 | 定义多个后端真实服务器。格式：server :[port] [param\*]\*\*说明：\*\*IP1:PORT1 和 IP2:PORT2 为后端服务器 IP 及端口号。weight 表示权重。 |

* 启动 HAProxy

taskset -c 0-3 /usr/local/haproxy/sbin/haproxy -f /usr/local/haproxy/conf/option-http\_proxy.cfg

* 验证 HAProxy

ps -ef | grep haproxy

可以看到 HAProxy 进程。打开浏览器在 url 处输入：http://IP:80，（IP 为 HAproxy 所在服务器 IP）可以看到后端服务器的页面，则说明 HAProxy 运行成功。刷新页面，显示页面在后端服务器间来回切换。说明：

* + （可选）停止 HAProxy 命令如下，业务运行中不需要执行该命令。

pkill haproxy

* + （可选）卸载 HAProxy，并查询。

rpm -qa | grep haproxyrpm -e --nodeps haproxy-1.9.0rpm -qa | grep haproxyrm -rf /usr/local/haproxy

# 第八章 Lighttpd 1.4.53 移植

Lighttpd 是开源 Web 服务器软件，其根本的目的是提供一个专门针对高性能网站，安全、快速、兼容性好并且灵活的 Web Server 环境。具有非常低的内存开销、CPU 占用率低、效能好以及丰富的模块等特点。Lighttpd 是众多 OpenSource 轻量级的 Web Server 中较为优秀的一个。支持 FastCGI，CGI，Auth，输出压缩(output compress)，URL 重写，Alias 等重要功能；而 Apache 之所以流行，很大程度也是因为功能丰富，在 Lighttpd 上很多功能都有相应的实现了，这点对于 Apache 的用户是非常重要的，因为迁移到 Lighttpd 就必须面对这些问题。开发语言：C一句话描述：Web 服务器

## 8.1 环境要求

| 软件 | 版本 |
| --- | --- |
| SeawayEdge | SeawayEdge-v3.3 |
| Kernel | 4.19 |

## 8.2 配置编译环境

安装依赖库yum -y install gcc gcc-c++ glib2-devel pcre-devel bzip2-devel zlib-devel gamin-devel

获取源码下载地址：https://download.lighttpd.net/lighttpd/releases-1.4.x/lighttpd-1.4.53.tar.gz

* 配置安装

cp lighttpd-1.4.53.tar.gz $HOME &amp;&amp; cd $HOMEtar xzvf lighttpd-1.4.53.tar.gz

* 编译安装

cd lighttpd-1.4.53./configure --prefix=/usr/local/lighttpd --with-fammake -j60 &amp;&amp; make install

## 8.3 配置参数

* 创建软件目录

cd /usr/local/lighttpd/mkdir log webpages cache config

* 拷贝配置文件目录

cp $HOME/lighttpd-1.4.53/doc/config/lighttpd.conf /usr/local/lighttpd/config/cp $HOME/lighttpd-1.4.53/doc/config/modules.conf /usr/local/lighttpd/config/cp $HOME/lighttpd-1.4.53/doc/config/conf.d /usr/local/lighttpd/config/ -r

* 修改 lighttpd.conf

vi /usr/local/lighttpd/config/lighttpd.conf

修改第 16-20 行为：

var.log\_root = &quot;/usr/local/lighttpd/log&quot;var.server\_root = &quot;/usr/local/lighttpd&quot;var.state\_dir = &quot;/usr/local/lighttpd&quot;var.home\_dir = &quot;/usr/local/lighttpd&quot;var.conf\_dir = &quot;/usr/local/lighttpd/config&quot;

修改第 61 行为：

var.cache\_dir = &quot;/usr/local/lighttpd/cache&quot;

第 93 行加注释：

#server.use-ipv6 = &quot;enable&quot;

修改第 104-105 行（该项为操作权限，不建议使用 root）为：

server.username = &quot;lighttpd1&quot;server.groupname = &quot;lighttpd&quot;

修改第 115 行（访问页面存放路径）为：

server.document-root = server\_root + &quot;webpages&quot;

修改第 246 行（访问页面存放路径）为：

server.stat-cache-engine = &quot;fam&quot;

修改第 182 行（访问页面存放路径）为：

server.max-worker = 4

* 创建用户组

groupadd lighttpduseradd -g lighttpd lighttpd1

* 修改权限

chown lighttpd1 /usr/local/lighttpd/log

* 添加测试页面

cd /usr/local/lighttpd/webpages vi index.html &lt;html&gt; &lt;head&gt; &lt;title&gt;lighttpd test&lt;/title&gt; &lt;/head&gt; &lt;body&gt; &lt;p&gt;this is a testing&lt;/p&gt; &lt;/body&gt; &lt;/html&gt;

## 8.4 服务器测

启动 lighttpd:

/usr/local/lighttpd/sbin/lighttpd -f /usr/local/lighttpd/config/lighttpd.conf

查看程序进程:

ps -ef |grep lighttpd

停止 apache:

pkill lighttpd

# 第九章 Iok 2.1.3 移植

Iok 是一款在屏幕上显示印度语言的键盘映射应用

## 9.1 兼容性检查

* 下载 iok-2.1.3 SRPM

wget <http://mirror.centos.org/centos/7/os/x86_64/Packages/iok-2.1.3-6.el7.x86_64.rpm>

* 下载 x2openEuler 工具

下载指引：<https://www.openeuler.org/zh/other/migration/>

* 部署工具

rpm -ivh x2openEuler-2.0.0-1.x86\_64.rpm

su x2openEulerx2openEuler redis-db -init

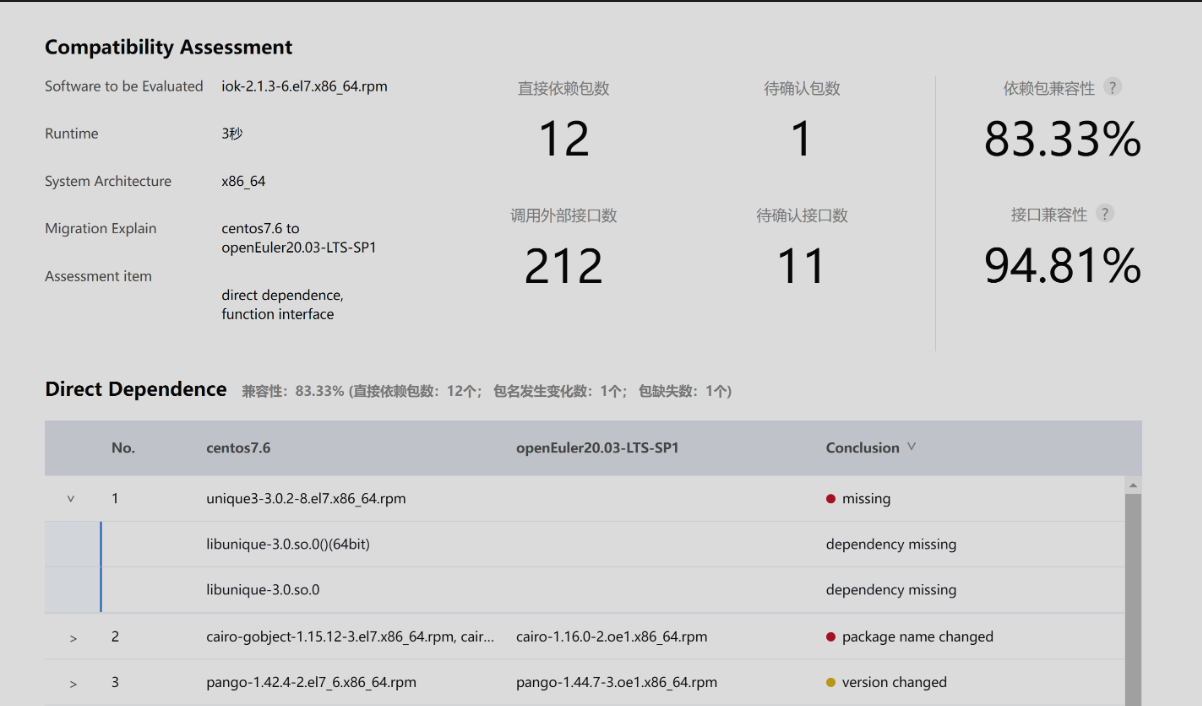
x2openEuler init source\_centos7.6-openEuler20.03-LTS-SP1.tar.gz

* 扫描软件

x2openEuler scan iok-2.1.3-6.el7.x86\_64.rpm注意要分析的移植文件需要有能够让x2openEuler用户可以读取的权限扫描完成后会在/opt/x2openEuler/output目录生成html格式的报告

## 9.2 查看评估结果

软件兼容性评估报告分三块内容展示软件兼容性，分别是依赖包兼容性、C/C++接口兼容性、java 接口兼容性，依赖包兼容性反映了软件包安装过程中的直接依赖，非 100%表明无法正确安装；接口兼容性反映的是单个软件运行过程中对其他软件包、动态库或系统接口的调用变化，非 100%表明在某个功能调用时可能会触发异常，未调用到时可能表现正常；部分结果建议人工复核，最终软件包使用建优先级建议 openEuler 已移植包>openEuler 上人工重编译包>centos 软件包。



结果：根据依赖报告可知，iok 移植到 openEuler 20.03 LTS SP1 需要解决 unique3 依赖问题

## 9.3 依赖包引入

* 在 openEuler/oec-application 仓库中发起 issue

仓库地址：<https://gitee.com/openeuler/oec-application>



* 持续追踪 issue 至缺失的依赖包被引入 openEuler 20.03 LTS SP1 的 YUM repo 中

## 9.4 构建流程

当缺失的依赖包被引入后可进行

获取 iok 的 Centos 7.6.1810 SRPM 包在 openEuler 20.03 LTS SP1 上构建二进制包

###构建二进制包

yum install -y rpm-build

提供 rpmbuild 命令

从网络安装 SRPM 包

rpm -i <https://vault.centos.org/7.6.1810/os/Source/SPackages/iok-2.1.3-6.el7.src.rpm>

安装依赖

yum-builddep -y ~/rpmbuild/SPECS/iok.spec

构建二进制包

rpmbuild -bb ~/rpmbuild/SPECS/iok.spec

二进制包安装

rpm -i ~/rpmbuild/RPMS/x86\_64/\*.rpm

查看二进制文件

which iok

# 第十章 Nginx 1.14.2 移植

Nginx 是一款轻量级的 Web 服务器/反向代理服务器及电子邮件（IMAP/POP3）代理服务器，其特点是占有内存少，并发能力强，支持 FastCGI、SSL、Virtual Host、URL Rewrite、gzip 等功能，并且支持很多第三方的模块扩展。开发语言：C一句话描述：Web 服务器/反向代理服务器及电子邮件（IMAP/POP3）代理服务器

## 10.1 环境要求

操作系统要求如表 2 所示。 表 2 操作系统要求、

| 项目 | 版本 | 版本查看命令 |
| --- | --- | --- |
| SeawayEdge | V3.3 | cat /etc/SeawayEdge-release |
| Kernel | 4.19.90 | uname -r |

## 10.2 配置编译环境

* 配置yum源

说明： 如果组网环境处于外网受限情况下，服务器 yum 命令无法通过外界获取依赖包时，可参考本节内容进行本地源配置。将操作系统镜像文件 openEuler-20.03-LTS-everything-aarch64-dvd.iso 文件拷贝到每台服务器的“/root”目录下。镜像文件挂载。 a. 将“/root”目录下的 openEuler 操作系统对应 iso 文件挂载到“/mnt”目录下。 mount /root/openEuler-20.03-LTS-SP1-everything-aarch64-dvd.iso /mnt 说明： 该操作单次生效，重启后失效。若需要配置开机启动自动挂载镜像（可选），可参考下面步骤。打开 fstab 文件。 vi /etc/fstab编辑 fstab 文件，在文件末尾添加如下信息： /root/openEuler-20.03-LTS-SP1-everything-aarch64-dvd.iso /mnt iso9660 loop 0 0保存并退出 fstab 文件。添加本地源文件。 a. 进入“/etc/yum.repos.d”目录。 cd /etc/yum.repos.d 说明： 此时，建议将此目录下的\*.repo 文件移到任意其他备份目录下。 b. 创建 local.repo 文件。打开 local.repo 文件。 vi local.repo编辑 local.repo 文件，在 local.repo 文件中添加如下内容：

[local]name=local.repobaseurl=file:///mntenabled=1gpgcheck=0

yum clean allyum makecacheyum list

* 安装依赖包

yum -y install gcc gcc-c++ make libtool zlib zlib-devel pcre pcre-devel pcre2-devel perl-devel perl-ExtUtils-Embed openssl openssl-devel

## 10.3 获取源码

本文使用源码编译安装，因此需要获取到 Nginx。下载 Nginx 源码  cd /homewget https://nginx.org/download/nginx-1.14.2.tar.gz --no-check-certificate说明： 也可以通过本地浏览器下载源码之后上传到服务器"/home"目录下。 源码地址：https://nginx.org/download/nginx-1.14.2.tar.gz注意： 若及其需要配置代理才可以访问外网，请参考下面操作配置网络代理。打开 profile 文件 vi /etc/profile添加如下代码后，保存并退出文件。 其中，代理服务用户名、代理服务器密码、代理服务器 IP 和代理服务端口需要根据当前环境配置

export http\_proxy="http://代理服务器名:代理服务器密码@代理服务器IP：代理服务器端口"export http\_proxy=$http\_proxyexport no\_proxy=127.0.0.1,.huawei.com,localhost,local,.local

使用代理效。 source /etc/profile查看环境变量中的代理信息。 env验证代理是否配置成功。 curl www.baidu.com 可以正常解析百度即为配置成功。

## 10.4 编译和安装

解压 Nginx 安装包。 tar -xvf nginx-1.14.2.tar.gz进入"nginx-1.14.2"目录。 cd /home/nginx-1.14.2/配置 Nginx。 ./configure --prefix=/usr/local/nginx --with-http\_ssl\_module 说明：--prefix=PATH:用来制定 Nginx 的安装目录，默认安装目录为"/usr/local/nginx"。不需要配置 with-http\_stub\_status\_module 模块，该统计模块会影响 Nginx 性能。编译并安装 Nginx make -j96 &amp;&amp; make -j96 install 说明： -j96: 充分利用 CPU 多核优势，加快编译安装速度。 CPU 的核数可以通过 lscpu 查看。 5.查看安装目录。 ls /usr/local/nginx

## 10.5 运行和验证

* 生成证书

进入"/usr/local/nginx"目录，在该目录下生成密钥 key。 cd /usr/local/nginxopenssl genrsa -des3 -out server\_2048.key 2048 会有两次要求输入密码，输入同一个即可，此时会生成 server\_2048.key 文件。

[root@localhost nginx]# openssl genrsa -des3 -out server\_2048.key 2048Generating RSA private key, 2048 bit long modulus (2 primes)..................................................................................+++++................+++++e is 65537 (0x010001)Enter pass phrase for server\_2048.key:Verifying - Enter pass phrase for server\_2048.key:

说明： 可通过如下命令实现免密码使用此文件： openssl rsa -in server\_2048.key -out -server\_2048.key

[root@localhost nginx]# openssl rsa -in server\_2048.key -out -server\_2048.keyEnter pass phrase for server\_2048.keywriting RSA key

创建服务器证书的申请文件。openssl req -new -key server\_2048.key -out server\_2048.csr

[root@localhost nginx]# openssl req -new -key server\_2048.key -out server\_2048.csrYou are about to be asked to enter information that will be incorporatedinto your certificate request.What you are about to enter is what is called a Distinguished Name or a DN.There are quite a few fields but you can leave some blankFor some fields there will be a default value,If you enter '.', the field will be left blank.Country Name (2 letter code) [AU]:CNState or Province Name (full name) [Some-State]:Locality Name (eg, city) []:Organization Name (eg, company) [Internet Widgits Pty Ltd]:Organizational Unit Name (eg, section) []:Common Name (e.g. server FQDN or YOUR name) []:Email Address []:Please enter the following 'extra' attributesto be sent with your certificate requestA challenge password []:An optional company name []:

输入 1 中设置的密码，其中 Country Name 选项输入 CN，其他选项可以不填。 3. 重写密钥 key。 openssl rsa -in server\_2048.key -out server\_2048.key

[root@localhost nginx]# openssl rsa -in server\_2048.key -out server\_2048.keywriting RSA key

生成证书。 openssl x509 -req -days 365 -in server\_2048.csr -signkey server\_2048.key -out server\_2048.crt

[root@localhost nginx]# openssl x509 -req -days 365 -in server\_2048.csr -signkey server\_2048.key -out server\_2048.crtSignature oksubject=C = CN, ST = Some-State, O = Internet Widgits Pty LtdGetting Private key

输入 1 中设置的密码。若已经设置免密码使用该文件，则无需输入密码。

* 配置功能

配置 Nginx 的 HTTPS 功能

打开 nginx.conf 配置文件。 vi /usr/local/nginx/conf/nginx.conf修改 nginx.conf 配置文件以下三处配置后，保存并退出（Esc+ :wq）。定义 Nginx 运行的用户权限 user 为 root。修改 listen 监测端口，可以使用默认端口，本文修改为 20000。指定 ssl\_certificate 和 ssl\_certificate\_key 文件。

原文默认内容：

#user nobody;... # HTTPS server # #server { # listen 443 ssl; # server\_name localhost; # ssl\_certificate cert.pem; # ssl\_certificate\_key cert.key; # ssl\_session\_cache shared:SSL:1m; # ssl\_session\_timeout 5m; # ssl\_ciphers HIGH:!aNULL:!MD5; # ssl\_prefer\_server\_ciphers on; # location / { # root html; # index index.html index.htm; # } #}

修改后内容：user root; ... HTTPS server server { listen 20000 ssl; server\_name localhost; ssl\_certificate /usr/local/nginx/server\_2048.crt; ssl\_certificate\_key /usr/local/nginx/server\_2048.key; ssl\_session\_cache shared:SSL:1m; ssl\_session\_timeout 5m; ssl\_ciphers HIGH:!aNULL:!MD5; ssl\_prefer\_server\_ciphers on; location / { root html; index index.html index.htm; } }

配置 Nginx 的 HTTP 功能打开 nginx.conf 配置文件。 vi /usr/local/nginx/conf/nginx.conf修改 nginx.conf 配置文件以下三处配置后，保存并退出（Esc+ :wq）。定义 Nginx 运行的用户权限 user 为 root。修改 listen 监测端口，可以使用默认端口，本文修改为 10000。

原文件默认内容：user root;...http { include mime.types; default\_type application/octet-stream; #log\_format main '$remote\_addr - $remote\_user [$time\_local] "$request" ' # '$status $body\_bytes\_sent "$http\_referer" ' # '"$http\_user\_agent" "$http\_x\_forwarded\_for"'; #access\_log logs/access.log main; sendfile on; #tcp\_nopush on; #keepalive\_timeout 0; keepalive\_timeout 65; #gzip on server { listen 80; server\_name localhost; #charset koi8-r; #access\_log logs/host.access.log main; location / { root html; index index.html index.htm; } }}

修改后内容：user root;...http { include mime.types; default\_type application/octet-stream; #log\_format main '$remote\_addr - $remote\_user [$time\_local] "$request" ' # '$status $body\_bytes\_sent "$http\_referer" ' # '"$http\_user\_agent" "$http\_x\_forwarded\_for"'; #access\_log logs/access.log main; sendfile on; #tcp\_nopush on; #keepalive\_timeout 0; keepalive\_timeout 65; #gzip on server { listen 10000; server\_name localhost; #charset koi8-r; #access\_log logs/host.access.log main; location / { root html; index index.html index.htm; } }}

运行 Nginx启动 Nginx（两种方式）。通过 Service 服务启动（使用该方法需要先将 Nginx 加入 Service 服务再执行启动命令）。修改“/etc/init.d/nginx”文件。 a. 删除原文件 nginx。 rm -rf /etc/init.d/nginx b. 新建 nginx 文件。 vi /etc/init.d/nginx c. 添加如下内容后，保存并退

#!/bin/bash # chkconfig: 2345 10 90 # description: nginx case "$1" in 'start') /usr/local/nginx/sbin/nginx -c /usr/local/nginx/conf/nginx.conf echo "$0\_start"; ;; 'stop') /usr/local/nginx/sbin/nginx -s quit echo "$0\_stop"; ;; esac

2. 修改“/etc/init.d/nginx”文件权限。

chmod 777 /etc/init.d/nginx 3. 将 Nginx 加入 chkconfig 管理列表。 chkconfig --add /etc/init.d/nginx 4. 设置 Nginx 开机自动启动。 chkconfig nginx on 5. 启动 Nginx。 service nginx start - 通过脚本命令启动。 /usr/local/nginx/sbin/nginx -c /usr/local/nginx/conf/nginx.conf 2. 查看 Nginx 的进程。 ps -ef | grep nginx

[root@localhost nginx]# ps -ef | grep nginxroot 9463 1 0 18:22 ? 00:00:00 nginx: master process /usr/local/nginx/sbin/nginx -c /usr/local/nginx/conf/nginx.confroot 9464 9463 0 18:22 ? 00:00:00 nginx: worker processroot 9466 1352 0 18:23 ttyAMA0 00:00:00 grep --color=auto nginx

说明： 关闭 Nginx 命令如下（3 种方式，可选）。业务运行中不要执行该命令。通过 Service 服务关闭。 service nginx stop通过脚本命令关闭。 /usr/local/nginx/sbin/nginx -s quit使用结束进程命令。pkill nginx

[root@localhost nginx]# pkill nginx[root@localhost nginx]# ps -ef | grep nginxroot 9469 1352 0 18:27 ttyAMA0 00:00:00 grep --color=auto nginx

验证 Nginx

查看 Nginx 的监测端口（10000 是 HTTP 监测端口，20000 是 HTTPS 监测端口）。 netstat -anp | grep 10000netstat -anp | grep 20000netstat -anpt

[root@localhost nginx]# netstat -anp | grep 10000tcp 0 0 0.0.0.0:10000 0.0.0.0:\* LISTEN 9535/nginx: master[root@localhost nginx]# netstat -anp | grep 20000tcp 0 0 0.0.0.0:20000 0.0.0.0:\* LISTEN 9535/nginx: master[root@localhost nginx]# netstat -anptActive Internet connections (servers and established)Proto Recv-Q Send-Q Local Address Foreign Address State PID/Program nametcp 0 0 0.0.0.0:10000 0.0.0.0:\* LISTEN 9535/nginx: mastertcp 0 0 0.0.0.0:22 0.0.0.0:\* LISTEN 775/sshd: /usr/sbin

查看 Nginx 的 HTML 文件所在目录。 ll -h /usr/local/nginx/html/

[root@localhost nginx]# ll -h /usr/local/nginx/html/total 8.0K-rw-r--r--. 1 root root 537 Mar 20 16:46 50x.html-rw-r--r--. 1 root root 612 Mar 20 16:46 index.html

验证 HTTPS 功能。 通过 curl 本地访问 Nginx 的 HTML 页面。 curl -k https://127.0.0.1:20000/index.html

[root@localhost nginx]# curl -k https://127.0.0.1:20000/index.htmlWelcome to nginx! body { width: 35em; margin: 0 auto; font-family: Tahoma, Verdana, Arial, sans-serif; }<h1>Welcome to nginx!</h1>If you see this page, the nginx web server is successfully installed andworking. Further configuration is required.For online ation and support please refer to<a href="#" class="white">nginx.org</a>.Commercial support is available at<a href="#" class="white">nginx.com</a>.<em>Thank you for using nginx.</em>

验证 HTTP 功能。 通过 curl 本地访问 Nginx 的 HTML 页面。 curl http://127.0.0.1:10000/index.html

[root@localhost nginx]# curl http://127.0.0.1:10000/index.htmlWelcome to nginx! body { width: 35em; margin: 0 auto; font-family: Tahoma, Verdana, Arial, sans-serif; }<h1>Welcome to nginx!</h1>If you see this page, the nginx web server is successfully installed andworking. Further configuration is required.For online ation and support please refer to<a href="#" class="white">nginx.org</a>.Commercial support is available at<a href="#" class="white">nginx.com</a>.<em>Thank you for using nginx.</em>

卸载 Nginx编译安装只是生成对应的文件，不涉及卸载，直接删除对应的安装目录即可。 rm -rf /usr/local/nginx

# 第十一章 Tengine 2.2.2 移植

Tengine是由淘宝网发起的Web服务器项目。它在Nginx的基础上，针对大访问量网站的需求，添加了很多高级功能和特性。它的目的是打造一个高效、安全的Web平台。开发语言：C一句话描述：轻量级Web服务器

## 11.1 环境要求

| 软件 | 版本 |
| --- | --- |
| SeawayEdge | SeawayEdge-v3.3 |
| Kernel | 4.19 |

## 11.2 安装Tengine

* 配置dns解析文件

[root@localhost ~]# echo "nameserver 114.114.114.114" >> /etc/resolv.conf

* 安装依赖包

[root@localhost ~]# yum install gcc gcc-c++ make libtool zlib zlib-devel pcre pcre-devel perl-devel perl-ExtUtils-Embed wget vim -y

* 镜像站RPM方式安装tengine

说明： 镜像站中的RPM包都是通过开源代码编译打包而成，然后将其上传到镜像站。

* 获取Tengine 2.2.2的RPM包。

[root@localhost ~]# cd /home[root@localhost home]# wget https://mirrors.huaweicloud.com/kunpeng/yum/el/7/aarch64/Packages/web/tengine-2.2.2-1.el7\_4.ngx.aarch64.rpm

* 兼容性评估

下载x2openEuler工具下载指引：<https://www.openeuler.org/zh/other/migration/>

部署工具rpm -ivh x2openEuler-2.0.0-1.x86\_64.rpm注意：安装rpm时需要使用root用户，且目前需要网络（用于下载安装依赖） 注意：根据提示安装依赖包如bzip2-devel等

su x2openEulerx2openEuler redis-db -init

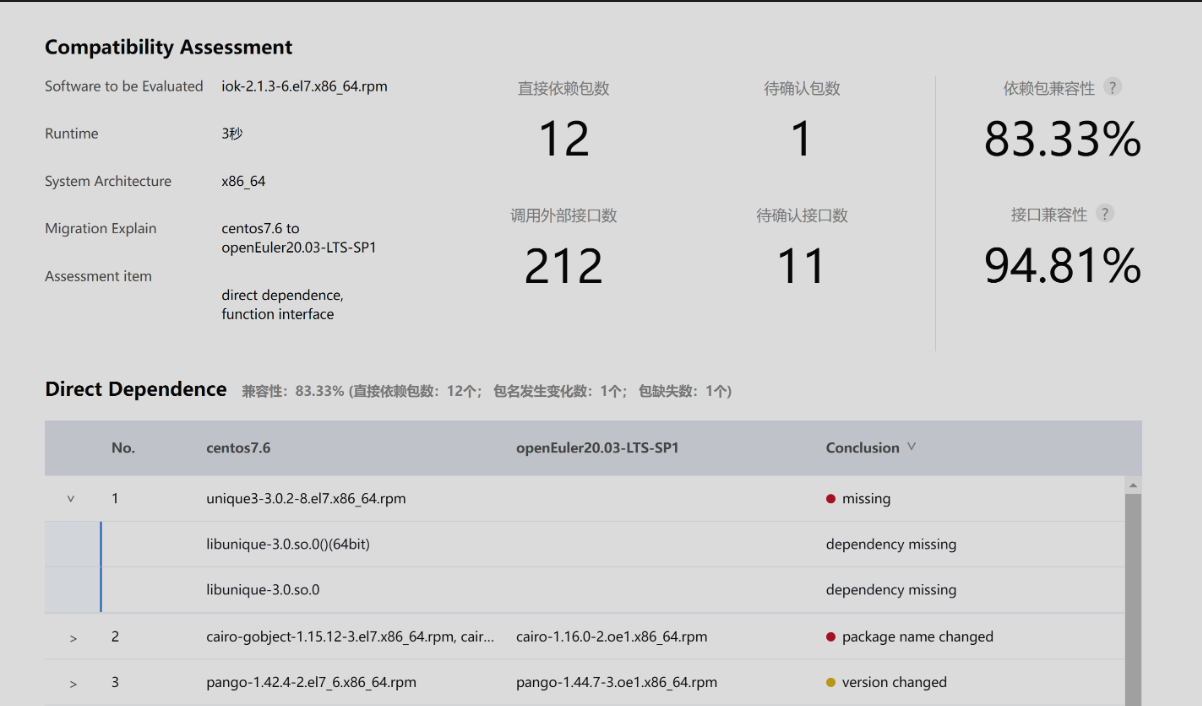
依次录入redis数据库的ip:127.0.0.1 端口：6379 数据库索引号（0-16）：0 密码（工具会对密码加密处理）：如果redis密码没有设置或者为空时，直接回车即可x2openEuler init source\_centos7.6-openEuler20.03-LTS-SP1.tar.gz

备注：x2openEuler使用rpm安装完成后会在/opt/x2openEuler目录下带有source\_centos7.6-openEuler20.03-LTS-SP1.tar.gz这个默认资源包 需要支持centos8.2到openEuler20.03-LTS-SP1的评估，则需获取对应的静态资源包导入，如对应的资源包为source\_centos8.2-openEuler20.03-LTS-SP1.tar.gz，导入此包命令：x2openEuler init source\_centos8.2-openEuler20.03-LTS-SP1.tar.gz，请示情况选择对应的资源包

扫描软件x2openEuler scan tengine-2.2.2-1.el7\_4.ngx.aarch64.rpm注意要分析的移植文件需要有能够让x2openEuler用户可以读取的权限扫描完成后会在/opt/x2openEuler/output目录生成html格式的报告

* 查看苹果结果

软件兼容性评估报告分三块内容展示软件兼容性，分别是依赖包兼容性、C/C++接口兼容性、java接口兼容性，依赖包兼容性反映了软件包安装过程中的直接依赖，非100%表明无法正确安装；接口兼容性反映的是单个软件运行过程中对其他软件包、动态库或系统接口的调用变化，非100%表明在某个功能调用时可能会触发异常，未调用到时可能表现正常；部分结果建议人工复核，最终软件包使用建优先级建议 openEuler已移植包>openEuler上人工重编译包>centos软件包。



结果：通过报告可知外部接口兼容性100%，依赖包兼容性人工复核后通过，经评估tengine2.2.2软件包在openEuler 20.03 LTS SP1系统上兼容，可安装此软件包至openEuler 20.03 LTS SP1系统进行验证。

* 安装Tengine

[root@localhost home]# rpm -ivh tengine-2.2.2-1.el7\_4.ngx.aarch64.rpmVerifying... ################################# [100%]Preparing... ################################# [100%]Updating / installing... 1:tengine-1:2.2.2-1.el7\_4.ngx ################################# [100%]

* 查看安装目录

[root@localhost home]# cd /usr/local/tengine-nginx/[root@localhost tengine-nginx]# lsconf html include logs modules sbin

。

## 11.3 运行和验证

* 配置HTTPS功能

生成证书

[root@localhost tengine-nginx]# openssl genrsa -des3 -out server\_2048.key 2048Generating RSA private key, 2048 bit long modulus (2 primes)...................................+++++..................+++++e is 65537 (0x010001)Enter pass phrase for server\_2048.key:Verifying - Enter pass phrase for server\_2048.key:[root@localhost tengine-nginx]# openssl rsa -in server\_2048.key -out server\_2048.keyEnter pass phrase for server\_2048.key:writing RSA key[root@localhost tengine-nginx]# openssl req -new -key server\_2048.key -out server\_2048.csrYou are about to be asked to enter information that will be incorporatedinto your certificate request.What you are about to enter is what is called a Distinguished Name or a DN.There are quite a few fields but you can leave some blankFor some fields there will be a default value,If you enter '.', the field will be left blank.-----Country Name (2 letter code) [AU]:CNState or Province Name (full name) [Some-State]:Locality Name (eg, city) []:Organization Name (eg, company) [Internet Widgits Pty Ltd]:Organizational Unit Name (eg, section) []:Common Name (e.g. server FQDN or YOUR name) []:Email Address []:Please enter the following 'extra' attributesto be sent with your certificate requestA challenge password []:An optional company name []:[root@localhost tengine-nginx]# openssl rsa -in server\_2048.key -out server\_2048.keywriting RSA key[root@localhost tengine-nginx]# openssl x509 -req -days 365 -in server\_2048.csr -signkey server\_2048.key -out server\_2048.crtSignature oksubject=C = CN, ST = Some-State, O = Internet Widgits Pty LtdGetting Private key[root@localhost tengine-nginx]# lsconf html include logs modules sbin server\_2048.crt server\_2048.csr server\_2048.key

* 配置Tengine

vim /usr/local/tengine-nginx/conf/nginx.conf

修改如下内容 # HTTPS server # #server { # listen 443 ssl; # server\_name localhost; # <== 修改这里： 证书位置 ==> # ssl\_certificate /usr/local/tengine-nginx/server\_2048.crt; # ssl\_certificate\_key /usr/local/tengine-nginx/server\_2048.key; # ssl\_session\_cache shared:SSL:1m; # ssl\_session\_timeout 5m;

* 运行Tengine

[root@localhost tengine-nginx]# /usr/local/tengine-nginx/sbin/nginx -c /usr/local/tengine-nginx/conf/nginx.conf[root@localhost tengine-nginx]# ps -ef | grep nginxroot 5710 1 0 17:25 ? 00:00:00 nginx: master process /usr/local/tengine-nginx/sbin/nginx -c /usr/local/tengine-nginx/conf/nginx.confnobody 5711 5710 0 17:25 ? 00:00:00 nginx: worker processroot 5713 1407 0 17:25 pts/0 00:00:00 grep --color=auto nginx

报http upstream check\_shm\_size is too small错误

[root@localhost tengine-nginx]# /usr/local/tengine-nginx/sbin/nginx -c /usr/local/tengine-nginx/conf/nginx.confnginx: [crit] ngx\_slab\_alloc() failed: no memorynginx: [emerg] http upstream check\_shm\_size is too small, you should specify a larger size.[root@localhost tengine-nginx]#[root@localhost tengine-nginx]# sed -i "/http {/a\check\_shm\_size 50m;" /usr/local/tengine-nginx/conf/nginx.conf[root@localhost tengine-nginx]# /usr/local/tengine-nginx/sbin/nginx -c /usr/local/tengine-nginx/conf/nginx.conf[root@localhost tengine-nginx]# ps -ef | grep nginxroot 5710 1 0 17:25 ? 00:00:00 nginx: master process /usr/local/tengine-nginx/sbin/nginx -c /usr/local/tengine-nginx/conf/nginx.confnobody 5711 5710 0 17:25 ? 00:00:00 nginx: worker processroot 5713 1407 0 17:25 pts/0 00:00:00 grep --color=auto nginx