

**企业实习中期报告**

学号： 2014220301019

姓名： 卢杰

专业方向： 嵌入式系统

企业名称： 英特尔亚太研发有限公司

实习岗位名称： 虚拟化开发工程师

企业指导教师： 王震宇

院内代管教师： 廖勇

**信息与软件工程学院**

目 录

[1. 企业实习的进展情况 1](#_Toc479152473)

[1.1. 实习工作完成情况 1](#_Toc479152474)

[1.2. 知识技能学习情况 1](#_Toc479152475)

[1.3. 职业素养学习培养 1](#_Toc479152476)

[1.4. 工程协作与交流情况 1](#_Toc479152477)

[2. 存在问题与解决方案 2](#_Toc479152478)

[2.1. 存在的主要问题 2](#_Toc479152479)

[2.2. 解决方案与可行性研究 2](#_Toc479152480)

[3. 前期任务完成度与后续实施计划 3](#_Toc479152481)

[参考文献 4](#_Toc479152482)

说明:

1.报告要求6000字以上。

2.本模板仅为基本参考，请各位同学根据个人情况进行目录结构扩展。

3.封面和目录单面打印，从正文第1页开始双面打印。

4.打印时删除本段说明。

## 1. 企业实习的进展情况

### 1.1. 实习工作完成情况

（详细介绍已完成的工作情况，包括对复杂软件工程问题的推理分析,针对复杂软件工程问题设计满足特定需求的总体设计和详细设计。对于已经完成工作，需对实施结果进行分析和解释，并通过信息综合得到合理有效的结论。）

* + 1. 推理分析

1. Container与VM比较

Vm的理念是在宿主的系统上，自己虚拟了一个硬件平台，然后运行一个不同的OS，每个vm需要很多资源，所以一台机器上跑不了几个vm。

Container的理念是依托于宿主机提供的内核，仅仅把一个不同的Linux发行版所需的特性打包成一个image，当运行一个ubuntu镜像，感觉就是在一个ubuntu操作环境里，但实际调用的系统接口都来自宿主机。因为共用宿主机内核，所以其所需资源很少，性能开销很小，一台机器可以起上千个Container。其比较如图1-1所示。

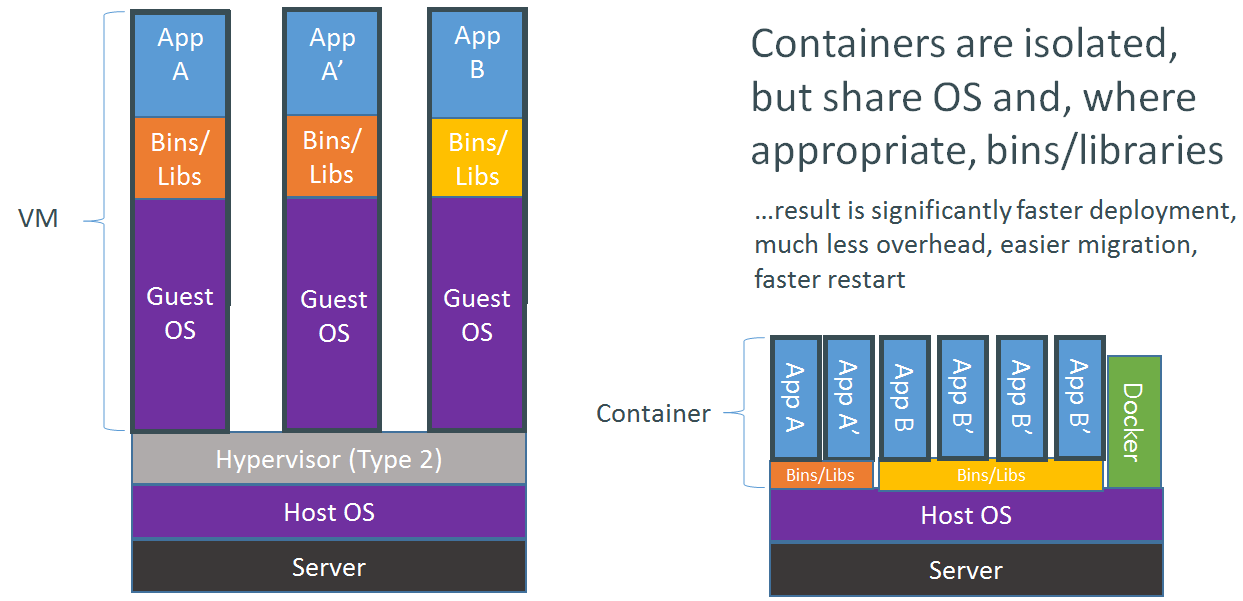


图1-1 Container与VM比较



图1-2 VM与Container关注点

(2) Container 组成

一个Container大体上由namespaces，resource control，filesystem组成，其中namespaces用于限制使用对象所能看见的空间，resource control用于限制使用对象能够使用的资源多少；filesystem的上层用于保存当前的修改，底层用于读取image。一个Container的组成如图1-3所示。

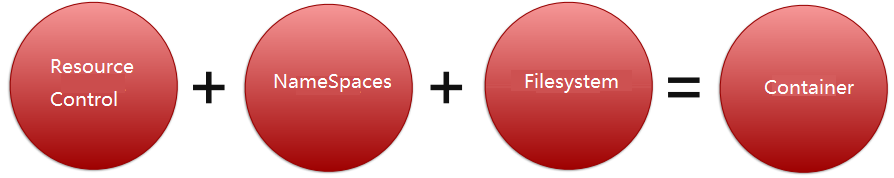


图1-3 Container基本组成要素

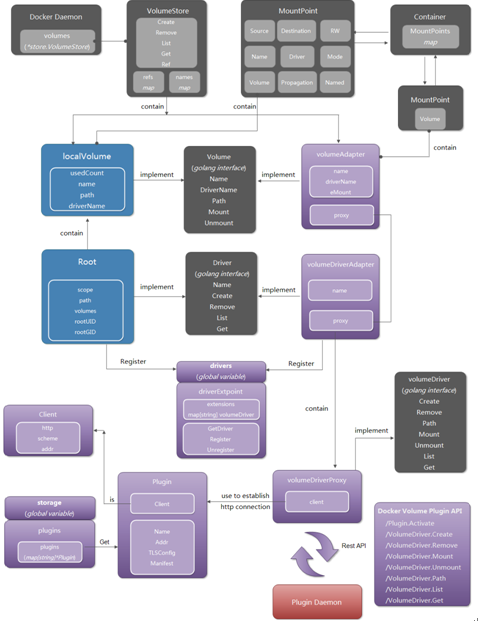
1. Container cgroup

cgroup 是 Linux 内核的一个特性，此特性可以帮助用户对一组进程进行资源使用的限制、统计以及隔离。 Docker 领域也不例外， Docker 利用对 cgroup 的支持，完成对 Docker 容器进程组的资源限制、统计以及隔离。与 cgroup 内容相关的 Cgroup 定义，位于 ./docker/libcontainer/cgroups/cgroups.go，如下:

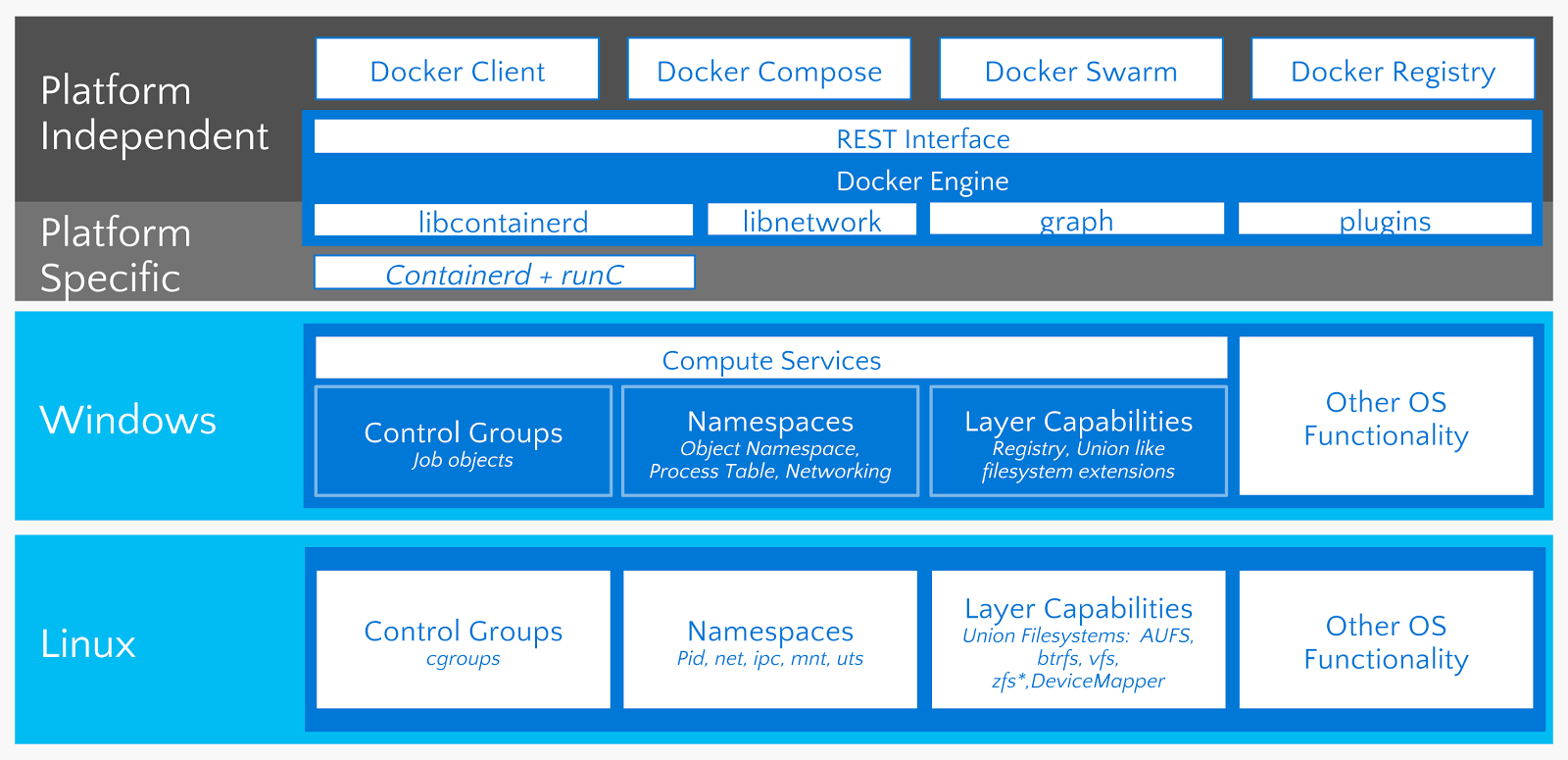
|  |
| --- |
| type Cgroup struct {  Name string  Parent string  AllowAllDevices bool  // name of parent cgroup or slice  // If this is true allow access to any  kind of device within the container. If false,  allow access only to devices explicitly listed  in the allowed devices list.  AllowedDevices []\*devices.DeviceMemory 工 nt64  MemoryReservat 工。 n int64  MemorySwap int64  CpuShares int64  CpuQuota int64  CpuPeriod int64  CpusetCpus string  第 14 章 libcontainer 介绍 e争 225  // Memory limit (in bytes)  // Memory reservation or soft\_limit (in bytes)  // Total memory usage (memory + swap); set  、 -1' to d 工 sable swap  // CPU shares (re1ative weight vs. other  conta 工 ners)  // CPU hardcap limit (in usecs). Allowed cpu  time in a given period.  // CPU period to be used for hardcapping (in  usecs). 0 to use system default.  // CPU to use  Freezer FreezerState // set the freeze value for the process  Sl 工 ce string // Parent slice to use for systemd |

从结构体 Cgroup 的定义可见， Docker 对于 cgroup 的支持，主要有以下 5 方面:设备( device)、内存 (Memo可)、 CPU 、 Freezer 以及 systemd。在容器设备方面， Docker 支持让用户选择容器可以使用的设备。在内存方面，支持为容器的运行设定用量限额。在 CPU 方面，支持容器进程之间拥有相对的运行时间片。 Freezer 则可以使容器挂起，节省 CPU 资源，Docker 命令中 pause 与 unpause 命令即采用了 cgroup 的 Freezer 子系统。需要注意的是，容器进程组挂起，并不意味着进程已经终止，从 Linux 内核的角度来看，被挂起的进程拥有完整的 task struct ，占用相应的内存，但是 Freezer 子系统保证容器中的进程不会被 CPU 调度到。 Slice 属性则属于 systemd 方面的配置。

1. volume plugin



* + 1. 总体设计



Libcontainer提供原生go去用namespaces，cgroup，capabilities和filesystem access control去创建一个container。RunC在运行时是一个轻量级通用容器，提供一个命令行界面，根据OCI规范运行容器。Containerd对于容器来说是作为一个daemon，管理容器在宿主机系统下的整个生命周期的活动，比如image 传输和存储，容器创建和挂起等。Docker是一个用于build，ship和run container的一个开源产品。其流程如下图1-4所示。

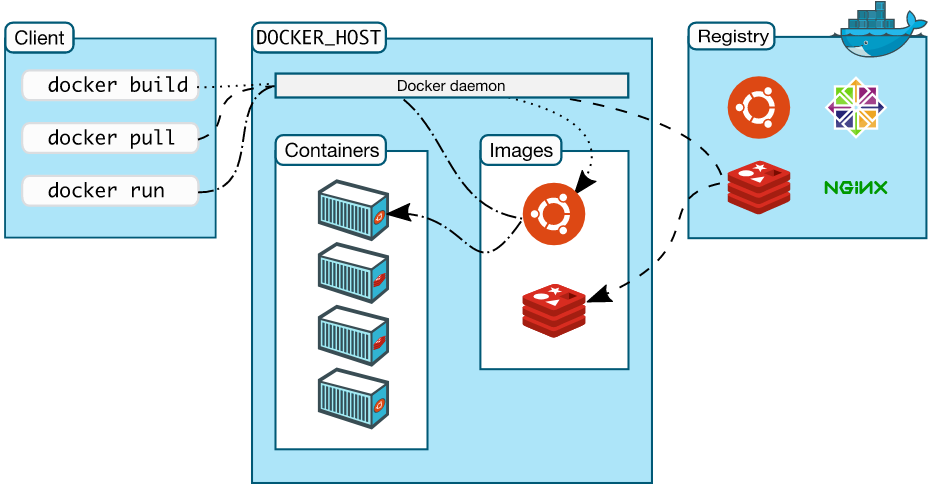
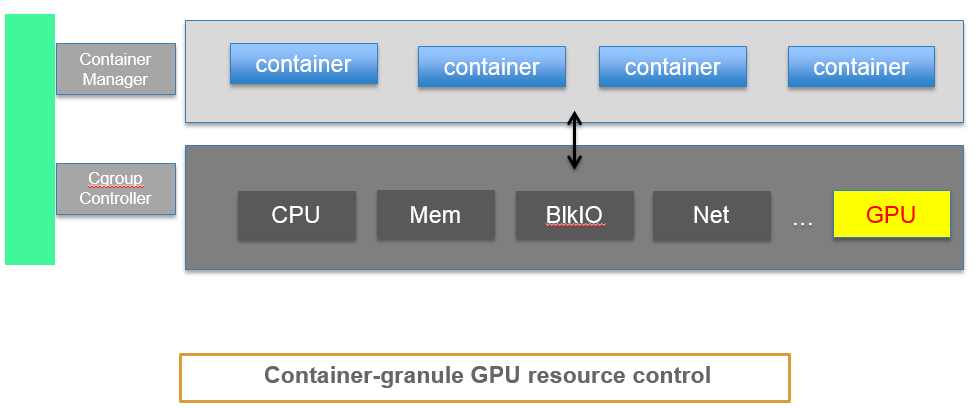


图1-4 client/server架构

* + 1. 详细设计



* + 1. 已完成工作

1. 在kernel cgroup里添加GPU资源控制，包括GPU priority和最大memory控制；
2. 在docker server端添加GPU资源控制；
3. 在RunC里添加GPU资源控制；
4. 在docker client端的CLI添加GPU资源控制；
5. 编写关于用Docker在linux container里使能GPU的wiki；
6. 成功在container里面运行yami的测试样例。
   * 1. 分析与解释
     2. 结论

### 1.2. 知识技能学习情况

（根据课题应用场景，选择开发环境、工具与技术标准；说明工具的学习和使用情况，文献和书籍的查阅、分析、总结和收获；阐述针对个人或职业发展的需求，进行的其它学习等。）

1.2.1 开发环境说明

本组项目的开发是brwndwire/skylake server上进行的，CPU为i5-5300U @2.3GHz（Max Speed 2.9GHz），4Core，16G Mem，GPU 900MHz，Turbo Ratio 57；软件环境为自己编译的ubuntu 4.12.0-rc3+版本；Docker选用的docker-ce 17.03版本；验证使用的mediaSDK和yami，驱动libva API 版本为0.39.0。

1.2.2 工具学习使用

在个人使用中最多使用的是MobaXterm Personal Edition工具，用于ssh连接远程服务器进行相关的操作。其经常使用功能如下：

（1）windows下支持多标签的终端：

通过MobaXterm进行远程终端链接，你可以创建 SSH, Telnet, Rlogin, RDP, VNC, XDMCP, FTP, SFTP or 串口等链接，如图1-6所示。你的每次链接都会自动保存并且出现在左侧链接窗口中。

全能型开源远程终端：MobaXterm

图1-6 可支持的终端链接

对于本机，MobaXterm让你可以在windows下运行Unix命令，如:ls, cd, grep, awk, tail, cut, sed, wget, rsync, telnet, ssh, rlogin, rsh…等等Unix基本命令。而且MobaXterm有很多免费的插件可以实现你更多的需求。

（2）图形化的SFTP 浏览器

当你链接一个ssh终端的时候，左侧窗口就会出现一个图形化的SFTP 浏览器，它可以让你通过安全的SFTP 链接拖放文件来管理远程服务器。（不喜欢可以关闭）

### 

图1-7 操作界面

（3）分屏显示及多任务执行

MobaXterm支持多分屏显示，方便管理多台服务器，并且你可以仅输入一次，让一条命令同时在这些不同的服务器终端执行。

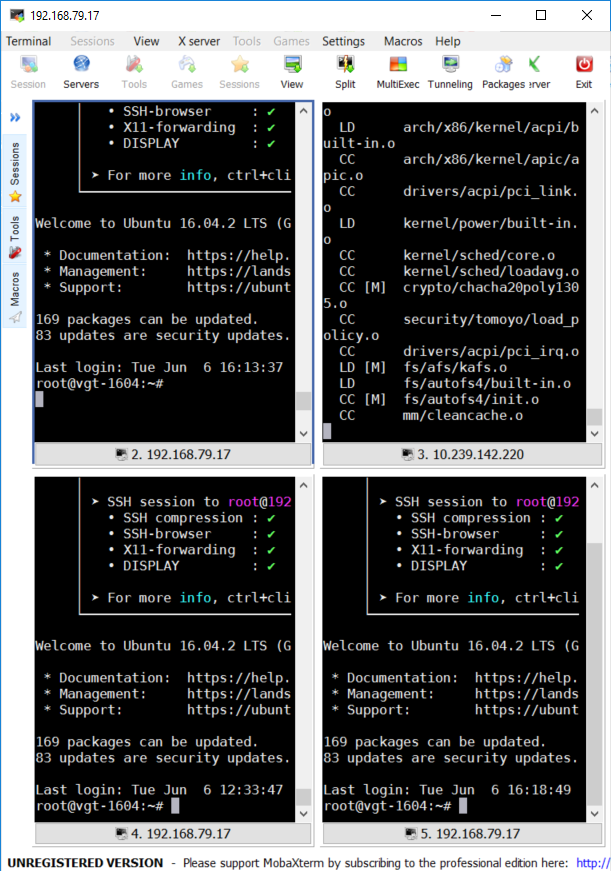


图1-8 分屏操作

1.2.3 文档查阅分析

（1）docker文档

（2）nvidia文档

1.2.4 其他学习内容

（1）mediaSDK安装与配置

当前版本的mediaSDK可以安装在CentOS7.2.1511版本或者添加了patch的linux4.4 kernel，其中CentOS上完全支持脚本自动化安装，如采用linux 4.4 kernel需要将软件包目录下的patch添加进代码中进行编译kernel。

（2）yami安装与配置

Yami的安装需要libva-dev的支持

### 1.3. 职业素养学习培养

（描述对软件工程系统的质量、环境、职业健康、安全和服务意识的学习和认识，对职业道德和规范的理解和遵守。）

1.3.1 软件工程系统

1.3.2 职业道德规范

作为一名软件工程师，为了实现对工程系统质量，安全和利益的承诺目标，我觉得应当坚持以下八项原则：

1. 公众----作为软件工程师应当始终如一地以符合公众利益为目标；
2. 客户和雇主----在保持与公众利益一致的原则下，我们软件工程师应满足客户和雇主的最高利益；
3. 产品----作为软件工程师应当确保我们的产品和相关的改进符合可能达到的最高专业标准；
4. 判断----作为软件工程师在进行相关的专业判断时应该坚持正直，诚实和独立的原则；
5. 管理----作为软件工程的管理和领导人员在软件开发和维护的过程中，应当自觉遵守，应用并推动合乎道德规范的管理方法；
6. 专业----作为软件工程师应当自觉推动本行业所提倡的诚实，正直的道德规范，并自觉维护本行业的声誉，使软件行业更好的为公众利益所服务；
7. 同僚----作为软件工程师对其同僚应持平等互助和支持的态度；
8. 自身----作为软件工程师应终身不断地学习和实践其专业知识，并在学习和实践的过程中不断提高自身的道德规范素养。

### 1.4. 工程协作与交流情况

（阐述在实习执行过程中，针对特定的目标或问题，与工程项目组成员，包括与其他学科的成员合作并开展工作的情况。）

本项目在公司第二季度开始实施的，每周二中午都会进行各自工作的汇总和下一步工作的重点分析。

## 2. 存在问题与解决方案

### 2.1. 存在的主要问题

（分析、总结和归纳实习执行过程中存在的主要问题与具体的应对措施，能通过具体事例来说明在心理素质培养以及应对项目风险与挑战的能力锻炼等方面是否有所收获。）

2.1.1 问题总结

2.1.2 解决方式

2.1.3 收获

### 2.2. 解决方案与可行性研究

（针对当前尚未解决的复杂工程问题，分析文献寻求解决方案，并在安全、环境、法律等约束条件下，通过技术经济评价对设计方案的可行性进行研究，说明方案的合理性。）

2.2.1 尚未解决的问题

设计Docker plugin，用于在创建容器之前进行驱动版本检查，以避免因驱动版本问题造成的影响。

2.2.2 解决方案

解决方案一：静态检测法，获取driver lable进行直接比较；

解决方案二：动态检测法，通过运行标准测试用例，根据结果进行判断；

解决方案三：后台隐藏法，第一个起的container处于后台运行模式，在该模式下运行预置的脚本和测试用例，如果测试不通过，则执行退出container操作并删除container，显示相关driver错误。

解决方案四：插件检测法，

2.2.3 可行性分析

2.2.3.1 nvidia 同类产品分析

Nvdia针对Docker Container推出了自己的解决方案—nvidia docker + nvidia docker plugin的形式，nvidia docker是在原生docker上封装了一层，对docker命令行接口进行一些替换和处理，如create和run命令；nvidia docker plugin则是充当一个daemon进程，用于发现host的驱动文件和GPU硬件，并对docker daemon发起的存储挂载请求做出响应。下面是关于nvidia docker对于驱动版本检查机制的分析，整体分析如图2-1所示。

1. 整体架构分析



图2-1 nvidia docker整体架构

Nvidia docker作为在原生docker之上的一层，对于create和run参数有着其自定义的处理步骤，之后才将处理结果传给原生docker进行处理。Nvidia docker与原生的docker交互发生在cudaSupported函数，其具体实现如图2-3所示。在nvidia docker获取image驱动版本的实现中考虑了remote和local两种情况，remote模式下采用json格式的文本传输获取CUDA版本；local模式下采用GetCUDAVersion函数实现CUDA版本的获取，该函数具体分析如图2-2所示。

1. 获取CUDA版本



图2-2 local模式获取cuda版本

在local模式下获取cuda版本，比较特殊的一步是driverGetVersion的本质是采用了go语言调用c语言是形式，引入cuda\_runtime\_api头文件，用go语言调用其cuda api里面的cudaDriverGetVersion函数来获取cuda版本。这里所获取到的版本将会用于和容器image里的lable版本进行比较得出驱动是否匹配的信息，以此决定是否创建一个容器。与Image里面的lable所进行比较的是该驱动所能支持的最大的CUDA版本。其具体流程如图2-3所示。

1. 获取lable版本



图2-3 获取image版本标签进行匹配

在编译CUDA image的Dockerfile里我们可以看见关于com.nvidia.cuda.

-version的定义，通过Lable可以指定image的版本，说明等；nvidia docker本质上还是调用了原生的docker的inspect命令，来获取image里面所定义的lable信息，docker inspect –-format 命令可以用于获取image或container里面的元数据，即这里所需要的com.nvidia.cuda.version标签信息。

2.2.4 结论

## 3. 前期任务完成度与后续实施计划

（前期任务完成度自我评价及后续工程环节的实施计划安排。除针对复杂工程问题的实施方案执行计划，还包括针对个人或职业发展需求的知识技能学习计划。）

### 3.1 前期任务完成度

### 3.2 后续实施计划

3.2.1 实施方案计划

3.2.2 知识技能学习计划

## 参考文献

[1] 王浩刚，聂在平.三维矢量散射积分方程中奇异性分析[J]. 电子学报，1999, 27(12): 68-71

[2] X. F. Liu, B. Z. Wang, W. Shao. A marching-on-in-order scheme for exact attenuation constant extraction of lossy transmission lines[C]. China-Japan Joint Microwave Conference Proceedings, Chengdu, 2006, 527-529

[3] 竺可桢.物理学[M]. 北京：科学出版社，1973, 56-60