启动裁剪优化方案

背景描述

ISO 镜像在物理机 or 虚拟机上启动时长优化。镜像的启动过程参考 url。

优化点分析定位

Dmesg 信息分析：

Kernel 会将开机信息存储在 ring buffer 中。如果开机来不及查看信息,可以用 dmesg 查看。同时也可以通过/var/log 目录中的 dmesg 文档查看。忽略 dmesg 中的关于系统架构、cpu、挂载硬件等无法操作项。

System-analyze 启动分析：

System-analyze 是 linux 自带的分析系统启动性能的工具。其常用功能如下：

systemd-analyze time：各个时间信息如下，在启动第一个用户态进程(init)之前，内核运行了多长时间； 在切换进入实际的根文件系统之前，initrd(initial RAM disk)运行了多长时间； 进入实际的根文件系统之后，用户空间启动完成花了多长时间。 注意，上述时间只是简单的计算了系统启动过程中到达不同标记点的时间，并没有计入各单元实际启动完成所花费的时间以及磁盘空闲的时间。

systemd-analyze blame 按照每个单元花费的启动时间从多到少

的顺序，列出所有当前正处于活动(active)状态的单元。 需要注意的是，这些信息也可能具有误导性， 因为花费较长时间启动的单元，有可能只是在等待另一个依赖单元完成启动。

systemd-analyze critical-chain [UNIT…] 为指定的单元(省略参数表示默认启动目标单元)以树状形式显示时间关键链(time-critical chain)。 “@”后面的时刻表示该单元的启动时刻；“+”后面的时长表示该单元总计花了多长时间才完成启动。不过需要注意的是， 这些信息也可能具有误导性， 因为花费较长时间启动的单元， 有可能只是在等待另一个依赖单元完成启动。

systemd-analyze plot 输出一个 SVG 图像，详细显示每个单元的启动时刻， 并高亮显示每个单元总计花了多长时间才完成启动。

systemd-analyze dot 按照 GraphViz dot(1) 格式输出单元间的依赖关系图。通常使用 systemd-analyze dot | dot -Tsvg > systemd.svg 命令来最终生成描述单元间依赖关系的 SVG 图像。除非使用了 –order 或 –require 选项限定仅显示特定类型的依赖关系，否则将会显示所有的依赖关系。如果指定了至少一个 PATTERN 参数(例如 *.target 这样的 shell 匹配模式)，那么将会仅显示所有匹配这些模式的单元的直接依赖关系。

Systemd-analyze time

```
[root@192 ~]# systemd-analyze
Startup finished in 1.758s (kernel) + 1.485s (initrd) + 9.753s (userspace) = 12.998s
```

Systemd-analyze blame #消耗时间从多到少排序

```
[root@192 ~]# systemd-analyze blame
          4.007s systemd-udev-settle.service
          2.648s kdump.service
          2.124s postfix.service
          1.559s lvm2-monitor.service
          1.463s dev-mapper-centos\x2droot.device
          1.025s tuned.service
           904ms NetworkManager-wait-online.service
           777ms dracut-initqueue.service
           576ms firewalld.service
           430ms initrd-switch-root.service
           416ms network.service
           298ms boot.mount
```

System-analyze critical-chain system-udev-settle.service

#查看耗时最长的服务，关联启动时间。

```
[root@192 ~]# systemd-analyze critical-chain  systemd-udev-settle.service
The time after the unit is active or started is printed after the "@" character.
The time the unit takes to start is printed after the "+" character.

systemd-udev-settle.service +4.007s
└─systemd-udev-trigger.service @423ms +106ms
  └─systemd-udevd-control.socket @377ms
```

systemctl list-unit-files --type=service | grep enabled

#查看还有哪些开机服务可以被优化

```
[root@192 ~]# systemctl list-unit-files --type=service | grep enabled
auditd.service                              enabled
autovt@.service                             enabled
chronyd.service                             enabled
crond.service                               enabled
dbus-org.fedoraproject.FirewallD1.service   enabled
dbus-org.freedesktop.nm-dispatcher.service  enabled
dmraid-activation.service                   enabled
firewalld.service                           enabled
getty@.service                              enabled
irqbalance.service                          enabled
iscsi-onboot.service                        enabled
iscsi.service                               enabled
kdump.service                               enabled
```

针对 ISO 文件优化

（1）服务裁剪

通过 systemd-analyse 分析系统启动性能，针对可优化或者可裁剪服务进行处理，分析过程可参考 优化点分析定位。

Systemctl list-units-files –type=service --all

#当前所有服务，针对业务不需要的服务进行裁剪。

```
[root@192 ~]# systemctl list-unit-files --type=service --all
UNIT FILE                                   STATE
anaconda-direct.service                     static
anaconda-nm-config.service                  static
anaconda-noshell.service                    static
anaconda-pre.service                        static
anaconda-shell@.service                     static
anaconda-sshd.service                       static
anaconda-tmux@.service                      static
anaconda.service                            static
auditd.service                              enabled
autovt@.service                             enabled
blk-availability.service                    disabled
```

法一:

发现影响启动的服务，可禁用该服务。

sudo systemctl disable NetworkManager-wait-online.service

```
[root@192 ~]# sudo systemctl disable NetworkManager-wait-online.service
Removed symlink /etc/systemd/system/network-online.target.wants/NetworkManager-wait-online.s
ervice.
```

法二:

编辑/lib/systemd/system/NetworkManager-wait-online.service 文件。将文件中的超时时间由 30 改为 10。

[Service] Type=oneshot ExecStart=/usr/bin/nm-online -s -q --timeout=30

法三:

直接删除服务所依赖的文件列表。.service 文件配置的服务常用 systemd 管理。然而，systemd 有系统和用户区分；系统（/user/lib/systemd/system/）、用户（/etc/lib/systemd/user/）。

Cat http.service # .service 信息如下，更详细的参考 url

```
[Unit]
Description=httpd       #当前配置文件的描述信息
After=network.target    #表示当前服务是在那个服务后面启动，一般定义为网络服务启动后启动

[Service]
Type=forking            #定义启动类型
ExecStart=/usr/local/apache/bin/apachectl start     #定义启动进程时执行的命令。
ExecReload=/usr/local/apache/bin/apachectl restart  #重启服务时执行的命令
ExecStop=/usr/local/apache/bin/apachectl stop       #定义关闭进程时执行的命令。
PrivateTmp=true                                     #是否分配独立空间

[Install]
WantedBy=multi-user.target    #表示多用户命令行状态
```

（2）Rpm 包裁剪

Mount xxx.iso /mnt

Mkdir iso

挂载文件系统，对根文件记性操作

Rpm -qa | sort > rpm.txt

获取系统上所有的 rpm 包名称信息，并输出到 rpmlsit.txt 文档中。

在后期裁剪中可以通过标记该文档中 rpm 包是否剪裁，以及剪裁失败的原因等，进行版本控制。

法一：

Rpm -qa | grep yum 或者 yum -version

查看系统是否安装 yum。

Yum remove [包名]

删除 rpm 包。

Mkiso xxxxx

重新构建 iso 文件，并在物理机或者虚拟机上验证是否可正常启动，启动后是否可正常运行。

法二:

当没有检测到安装了 yum 工具，或者 yum 功能已被裁减掉。可通过直接删除 rpm 包安装的文件路径，通过删除文件方式删除安装的 rpm 包。

```
Rpm -ql [完整的 rpm 包名]    #获取 rpm 包安装的文件路径
Rm -rf addressName           #循环删除所有查找出的文件
```

法三:

在使用 kicksart 文件构建 ISO 镜像时，通过修改 ks 文件中的%package 选项，选择是否安装相应的 rpm 包。

注意：于 rpm 包的安装依赖问题，可通过 yum remove [软件包名]获取依赖包信息，先在%package 中删除依赖包，在删除当前包。

软件包段：

```
%packages #表示开始
    @group #要安装的包组
    Package#要安装的包
    -Package #不要安装的包
%end          #结束
```

（3）内核模块裁剪

挂载 ISO 镜像的根文件系统，使用 chroot 进入根文件系统。

/usr/lib/modules/`uname -r`/kernel #进入内核模块目录

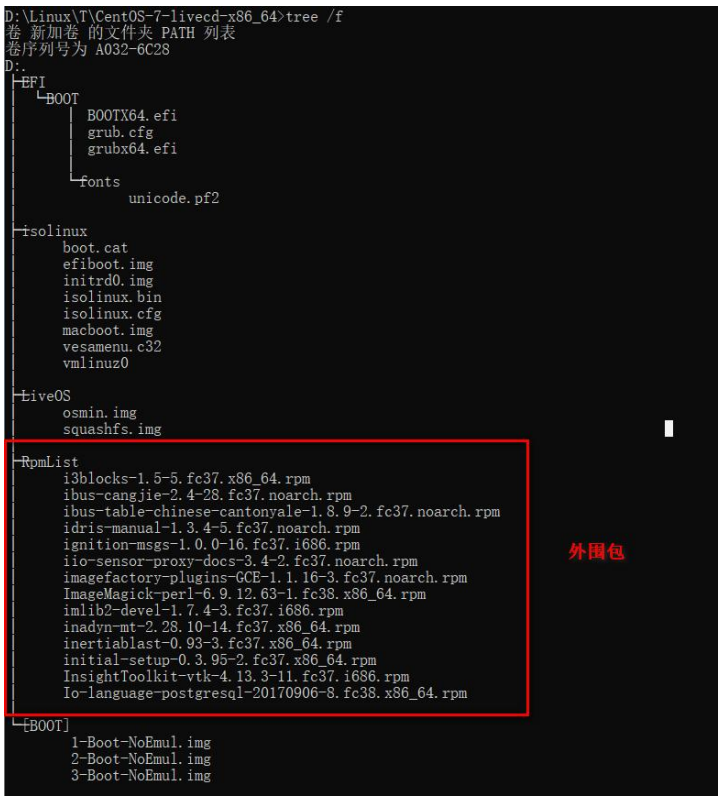
明确项目不需要的内核模块。通过删除内核模块文件(.ko 文件)，达到删除不需要的内核功能。

注意：同厂商的操作系统，内核模块文件的组织方式也会不同，Centos 的内核模块文件以.ko.xz （进一步压缩文件系统的体积）。FusionOS 的内核模块文件以 .ko （可直接加载）

Rm -rf [内核模块文件] #循环删除所有的内核模块文件

（4）软件包加载优化

将外围包（系统启动过程中安装的一些 rpm 包），直接原装到 initrd.img 镜像中。这样可以缩短 rpm 包的安装耗时，但是同时 initrd.img 的镜像体积会增大，在 copy image 到内存中的耗时会增加，可以对比耗时选择方案。



ISO 构建过程优化

（1）kickstart 文档调优

在使用 livemedia-creator 工具构建 ISO 镜像的时候，kickstart 提供了部分或完整的自动化安装程序，包含了安装过程需要询问的问题。通过修改 kickstart 文档，可以控制 ISO 文件内容。

```
part / --size 10240 --fstype ext4    #调整 size，控制分区大小
```

在 ISO 文件启动加载过程，初始化分区过大会消耗部分时间，可通过控制分区大小，优化初始化过程。

```
    启动预执行段：
%post #表示开始
    [shell script]   #在安装完成后，系统第一次重新开启之前，要在系统上执行的指令。
%end            #结束
```

可以将上述针对服务裁剪，针对内核模块裁剪的脚本相关指令写在此处。

（2）ISO 构建工具参数调优

修改 livemedia-creator 工具的参数信息，优化启动速度。可调整参数如下：

```
--anaconda-arg=”—nosave=all_ks”
--dracut-arg=”—lz4”#修改 initramfs 的压缩方式，更改为解压
```

缩率更高的算法

--dracut-arg=”—strip”

（4）编译内核

工程 tips

版本控制

使用 git 进行版本控制，记录每次修改记录，方便进行回滚。

进展同步

和下游信息对齐：

明确处理的 ISO 和其中 kernel 版本信息，剪裁对象一致。

实时交互处理精度和优化思路。

及时会议，理解清楚自己的任务和目标。

优化差错分析定位

裁剪过度：

小批量 rpm 包裁剪和验证，发现问题后可以通过二分法进行缺失 rpm 包的定位。此方法也可适用于其他的裁剪方案中。