
公 开



KYLIN-OS.TD-103-V2022

Kylin Server V10

服务器整机认证测试套件使用说明文档

麒麟软件有限公司

目录

目录	3
1 引言	1
2 概述	1
2.1 测试环境	1
2.2 测试要求	1
3 使用步骤说明	2
3.1 配置离线源	2
3.1.1 配置离线 yum 源	2
3.1.2 配置离线 pip3 源	4
3.2 客户端安装	4
3.2.1 手动安装	5
3.2.2 脚本安装	6
3.3 服务端安装	7
3.3.1 手动安装	8
3.3.2 脚本安装	8
3.4 工具授权认证	9
3.5 测试准备	10
3.5.1 服务端测试准备	10
3.5.2 客户端测试准备	11
3.6 测试执行	16
3.7 测试结果	20
3.7.1 客户端测试结果查看	20
3.7.2 服务端测试结果查看	20
4 提交认证测试	23
5 测试项说明	24
5.1 整机认证测试集	24
5.1.1 acpi	24
5.1.2 boottime	25
5.1.3 cdrom	25
5.1.4 core	27
5.1.5 cpufreq	28
5.1.6 disk	32
5.1.7 filesystem	34
5.1.8 hardwareinfo	35
5.1.9 infiniband	36
5.1.10 infiniband_netperf	39
5.1.11 ipmi	40

5.1.12 kdump	41
5.1.13 loigcalvolume	42
5.1.14 memory	44
5.1.15 mobileharddisk	46
5.1.16 mobilehdhotplug	47
5.1.17 ethernet	48
5.1.18 nvme	50
5.1.19 roce	51
5.1.20 perf	54
5.1.21 system	55
5.1.22 udiskhotplug	55
5.1.23 usb	56
5.1.24 usb_keyboard	57
5.1.25 usb_mouse	58
5.1.26 usb_udisk	59
5.1.27 usbcdriverhotplug	60
5.1.28 video	61
5.1.29 virtualization	62
5.1.30 watchdog	63
5.2 性能测试集	64
5.2.1 stream	64
5.2.2 fio	66
5.2.3 iozone	67
5.2.4 lmbench	69
5.2.5 netperf	70
5.2.6 speccpu	72
5.2.7 specjvm	73
5.2.8 unixbench	74
5.3 稳定性测试集	75
5.3.1 ltp	75
5.3.2 rebootstress	76
5.3.3 memtester_stress	77
6 FAQ	77
6.1 PERF 测试 FAIL	77
6.2 KDUMP、WATCHDOG、REBOOTSTRESS 测试后没有自动打包测试日志	78
6.3 测试 FAIL 查看报错信息	78
6.4 INTEL CPUFREQ 测试问题	79
6.5 海光测试 CPUFREQ FAIL 问题	80
6.6 终端打开 KYLINCH 提示没有需要运行的测试集，退出测试。	80
6.7 测试完成后无法上传日志到 SERVER 端	81
6.8 测试 KDUMP 或 WATCHDOG 或 REBOOTSTRESS 失败	81
6.9 CPUFREQ 测试 FAIL，LOG 显示无法获取 CPU 频率信息	82
6.10 IPMI 测试项中 CHECK REBOOT LOG 子项测试失败	82

6.11 IPMI 测试项中 FRU 子项测试失败	82
6.12 V10SP1 系统 INTEL 第三代 CPU ICELAKE 测试 VIRTUALIZATION FAIL.....	83
6.13 系统不带 X710 网卡驱动，安装第三方驱动导致 SYSTEM 测试 FAIL.....	85
6.14 SP2 系统测试 LMBENCH 编译失败，日志中显示缺少 RPC.H.....	86
6.15 已知问题	86

KYLINSOFT

1 引言

此文档主要对服务器整机认证测试套件 kylinch 的安装、使用以及测试用例进行了介绍。

2 概述

kylinch 整机认证测试是为了保证麒麟操作系统与硬件设备之间具有良好的协同性。它是一款操作灵活、功能强大的测试工具，它可以自动探测系统的硬件配置信息，并针对已检测到的硬件设备通过执行相关的测试用例进行兼容性和性能等测试。

kylinch 是我们对厂商产品进行整机认证时所使用的一款测试工具。主要包含客户端模块和服务端模块。客户端用于测试执行，服务端除了作为网络测试 server 端提供服务外，还提供了测试日志收集、在线查看以及下载功能。

2.1 测试环境

硬件环境：x86_64、aarch64、mips64el、loongarch、sw64

软件环境：Kylin Linux Advanced Server V10 (SP1)

Kylin Linux Advanced Server V10 (SP2)

2.2 测试要求

系统安装要求：

1. 客户端和服务端安装 kylin 操作系统，安装时“软件选择”请选择“带

UKUI GUI 的服务器”，并勾选“已选环境的附加项”下的全部内容。

2. 除了操作系统安装的硬盘之外，被测服务器还需要一颗额外的磁盘，用于进行磁盘的格式化以及读写测试。比如 Kylin 系统安装在 sda 上，那么被测服务器需要额外一颗 sdb 的盘进行硬盘相关测试。

其他要求：

1. 最好保证测试系统有网络连接。
2. 如设备不自带光驱，需要外接光驱测试 cdrom 项。

3 使用步骤说明

系统安装完毕后，请根据以下步骤则可以完成安装测试工具 kylinch。

kylinch 执行测试主要包括三个步骤：

1. 根据硬件情况自动列出需要执行的测试用例
2. 执行测试
3. 结果输出

工具安装以及测试步骤说明参见下述章节。

3.1 配置离线源

对于离线环境，无法使用系统自带的源，需要配置离线 yum 源和离线 pip3 源。

3.1.1 配置离线 yum 源

对于无法连接到麒麟 yum 源的环境，需要自行配置 yum 源。

1. 查看 yum 配置文件，确定 yum 源

```
# cat /etc/yum.repos.d/kylin_{arch}.repo
```

例如下图 baaeurl 中 yum 源使用的是 V10SP1.1

```
[root@localhost ~]# cat /etc/yum.repos.d/kylin_mips64el.repo
###Kylin Linux Advanced Server 10 - os repo###

[ks10-adv-os]
name = Kylin Linux Advanced Server 10 - Os
baseurl = http://update.cs2c.com.cn:8080/NS/V10/V10SP1.1/os/adv/lic/base/$basearch/
gpgcheck = 0
enabled = 1

[ks10-adv-updates]
name = Kylin Linux Advanced Server 10 - Updates
baseurl = http://update.cs2c.com.cn:8080/NS/V10/V10SP1.1/os/adv/lic/updates/$basearch/
gpgcheck = 0
enabled = 0

[ks10-adv-addons]
name = Kylin Linux Advanced Server 10 - Addons
baseurl = http://update.cs2c.com.cn:8080/NS/V10/V10SP1.1/os/adv/lic/addons/$basearch/
gpgcheck = 0
enabled = 0
```

图 1-mips 架构 yum 源示例

2. 到百度网盘下载对应版本的本地 yum 源，拷贝到客户端和服务端并解压。

3. 使用 kylin-ch 压缩包中的 install 脚本配置本地 yum 源,或手动配置 yum 源，路径为 repodata 文件夹的父目录。

脚本设置：

```
# bash install.sh -setyum <path>
```

例：

```
# bash install.sh -setyum "/root/V10SP1.1"
```

手动配置：

编辑/etc/yum.repos.d/kylin_{arch}.repo

```
# vim /etc/yum.repos.d/kylin_{arch}.repo
```

将原有配置的 enabled = 1 改为 enabled = 0

在文件下新增如下内容：


```
[kylinchlocal]
name = kylinchlocal
baseurl = file:///root/V10SP1.1/
gpgcheck = 0
enabled = 1
```

保存文件并退出

```
# yum makecache
```

命令执行成功，配置完成。

3.1.2 配置离线 pip3 源

1. 在百度网盘下载对应架构的 pip3_package.zip。
2. 安装 pip: yum install python3-pip
3. 使用 kylin-ch 压缩包中的 install 脚本配置本地 pip3 源,或手动配置 pip3

源，路径为 simple 所在目录。

(1) 脚本配置：

```
# bash install.sh -setpip3 <path>
```

例：

```
# bash install.sh -setpip3 'file:///root/pip3_package/simple/'
```

(2) 手动配置：

```
# pip3 config set global.index-url 'file:///root/pip3_package/simple/'
# pip3 config list
```

3.2 客户端安装

需要进行认证测试的机器作为客户端，可以使用脚本自动安装客户端相关服务，或者通过手动安装的方式实现。

使用脚本安装会将需要使用的依赖包全部安装好并做好启动服务等操作，手动安装则需要自行安装工具包，以及开启服务等。

手动解压 vmtest.zip 到/opt/目录，用于测试 virtualization。

推荐使用脚本安装，离线环境请先参考“3.1 配置离线源”后再安装 **kylinch**

注意：如果加载用例集中包括性能测试集和稳定性测试集，需要手动将性能测试工具包“**performance_server.zip**”解压到/opt/performance_server。

3.2.1 手动安装

手动安装的操作步骤如下：

1. 将 **kylinch** 的压缩包拷贝到测试机上并解压
2. 安装 **kylinch** 客户端的 rpm 包


```
# rpm -ivh kylin-ch-<version>-<date>.<os>.<arch>.rpm
# rpm -ivh
kylin-ch-testsuites-<version>-<date>.<os>.noarch.rpm
```
3. 关闭防火墙和 selinux


```
# systemctl stop firewalld
# iptables -F
# setenforce 0
```
4. 安装 pip3 依赖包


```
# pip3 install ptyprocess pexpect urwid
```
5. 在对应架构的文件夹下，找到 **fio**、**netperf** 和 **memtester** 测试工具并安装
 安装完成后，在终端输入'**kylinch**'，如果可以正常运行，则表示安装成功。**需要先进行授权认证才能打开 **kylinch****，详情见 3.4 工具授权认证。
6. 如果加载用例集中包括性能测试集或整机压力测试集，需要手动将性能测试工具包解压到/opt/performance_server（目录结构见图 3）

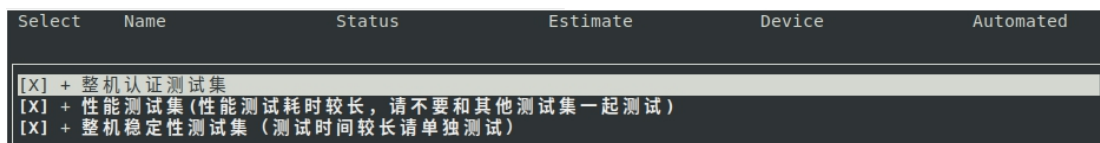


图 2-加载测试集界面

```
[root@localhost performance_server]# pwd
/opt/performance_server
[root@localhost performance_server]# ll
总用量 292
-rwxrwxr-x 1 root root 2376 2月 22 2021 com_syslog.h
drwxr-xr-x 8 root root 88 8月 26 15:07 config
-rwxrwxrwx 1 root root 38056 8月 26 15:09 fio.sh.x
-rwxrwxrwx 1 root root 29240 8月 26 15:09 iozone.sh.x
-rwxrwxrwx 1 root root 28872 8月 26 15:09 lmbench.sh.x
-rwxrwxr-x 1 root root 70 2月 22 2021 netperf.h
-rwxrwxrwx 1 root root 38088 8月 26 15:09 netperf.sh.x
-rwxrwxr-x 1 root root 1779 4月 20 14:11 netserver.sh
drwxr-xr-x 5 root root 47 8月 26 15:07 script
-rwxrwxr-x 1 root root 124 2月 22 2021 speccpu.h
-rwxrwxrwx 1 root root 38472 8月 26 15:09 speccpu.sh.x
-rwxrwxrwx 1 root root 25512 8月 26 15:09 specjvm.sh.x
-rwx--x--x 1 root root 26168 8月 30 16:37 stream.sh.x
drwxr-xr-x 12 root root 284 8月 28 12:07 tools
-rwxrwxrwx 1 root root 30696 8月 26 15:09 Unixbench.sh.x
-rw-rw-r-- 1 root root 34 8月 25 17:11 version.txt
[root@localhost performance_server]#
```

图 3-performance_server 目录结构图

```
[root@localhost vmtest]# pwd
/opt/vmtest
[root@localhost vmtest]# ll
总用量 20974984
-rw----- 1 root root 21478375424 4月 22 19:28 kylin10.0-mips64el.qcow2
-rw-r--r-- 1 root root 3428 11月 20 16:44 kylin10.0-mips64el.xml
[root@localhost vmtest]#
```

图 4-vmtest 目录结构图

3.2.2 脚本安装

脚本安装操作步骤如下：

1. 将 kylinch 的压缩包、performance 压缩包、vmtest 压缩包拷贝到测试机的任一目录下；

注：performance_server_*.zip 用于为性能测试集和稳定性测试集中的

ltp 提供测试资源

vmtest-*.zip 用于为认证测试集中的 virtualization 项提供测试资源，

x86 架构可以查看 lscpu 的 flags 中是 vmx 还是 svm，下载对应的 vmtest 到客户端

2. 解压 kylinch 压缩包，其他压缩包不需要解压，脚本会自动解压放置；
3. 执行安装脚本进行工具安装（如果是离线环境，请先参考 3.1 配置离线 yum 源和离线 pip3 源）

```
# cd 到 kylinch 解压后的文件夹
```

```
# bash install.sh -client
```

4. 安装完毕显示“PASS”证明安装通过，若未显示“PASS”，屏幕会打印未安装的依赖包，可手动安装依赖包，例如：

```
# yum install xxx
```

```
# pip3 install xxx
```

```
.....安装fio工具.....
fio已安装
[INFO]-----安装netperf工具-----
netperf已安装
-----安装memtester工具-----
memtester已安装
setenforce: SELinux is disabled
PASS
```

图 5-client 端安装 pass

```
netperf已安装
-----安装memtester工具-----
memtester已安装
setenforce: SELinux is disabled
----- error summary -----
stress has not been installed
```

图 6-client 端安装 fail

5. 检查/opt/performance_server 目录结构如图 3，/opt/vmtest 目录结构如

图 4

3.3 服务端安装

服务端可以安装在与被测机器网络连通的任意服务器上，多台客户端可以共用同一服务端，但是**连接同一服务端的多台客户端不能同时进行网络测试**。

服务端可以通过脚本自动安装或者手动安装，使用脚本安装会将需要使用的依赖包全部安装好并做好启动服务等操作，手动安装则需要自行安装工具包以及

开启服务等。

推荐使用脚本安装，如果服务端所在的服务器没有网络连接，只能进行离线测试，请参考 3.1 配置离线源。

3.3.1 手动安装

手动安装的操作步骤如下：

1. 将 kylinch 的压缩包拷贝到测试机上并解压

2. 安装 kylinch 服务端的 rpm 包

```
# rpm -ivh
kylin-ch-server-<version>-<date>.<os>.<arch>.rpm
```

3. 关闭防火墙和 selinux

```
# systemctl stop firewalld
# iptables -F
# setenforce 0
```

4. 在对应架构的文件夹下，找到 netperf 测试工具并安装，用于为客户端进行网络测试时提供服务

5. 服务端 web 功能需要安装如下依赖包

```
# pip3 install Flask Flask-bootstrap uwsgi pyyaml
```

如果下载速度慢，可以配置国内 pip 源进行安装，比如：

```
# pip3 install Flask Flask-bootstrap uwsgi pyyaml -i
http://mirrors.aliyun.com/pypi/simple/ --trusted-host
mirrors.aliyun.com
```

6. 服务端默认使用 8080 端口，同时搭配 nginx（默认端口 80）提供

web 服务，请保证这些端口未被占用，之后通过如下命令启动服务：

```
# systemctl start kylinch-server.service
# systemctl start nginx.service
```

3.3.2 脚本安装

脚本安装操作步骤如下：

1. 将 kylinch 的压缩包拷贝到测试机上并解压

2. 执行安装脚本进行工具安装

```
# bash install.sh -server
```

3. 安装完毕显示“PASS”证明安装通过，若未显示“PASS”，屏幕会打印未安装的依赖包，可手动安装依赖包。

```
1:kylin-ch-server-1.0.0-20210805.ky##### [100%]
setenforce: SELinux is disabled
[INFO] kylinch-server 服务开启成功
[INFO] nginx 服务开启成功
PASS
```

图 7-server 端安装 pass

```
Failed to start nginx.service: Unit nginx.service not found.
[ERROR] nginx 服务开启失败
----- error summary -----
nginx has not been installed
python3-pip has not been installed
python3-flask has not been installed
Flask-bootstrap has not been installed
uwsgi has not been installed
nginx 服务开启失败
```

图 8-server 端安装 fail

3.4 工具授权认证

测试前需要先认证工具才能进行测试，**仅客户端需要授权认证，服务端不需要认证。**

1. 在测试机上首次安装 kylinch，执行完以上的安装步骤之后，终端执行命令 kylinch 有如下图显示，表示需要 key 文件来进行授权认证，认证之后才能使用工具。

注意：key 文件有时间期限，到期后请重新获取 key 文件认证。

```
[root@localhost ~]# kylinch
[ERROR] 工具未认证，请先认证后再执行工具。
[root@localhost ~]#
```

图 9-kylinch 授权认证

2. kylinch --auth KylinchAuthorization.key 文件的路径


```
[root@localhost ~]# kylinch --auth /root/KylinchAuthorization.key
[INFO] 认证成功.
[root@localhost ~]#
```

图 10-kylinch 授权认证成功

3. 注意事项:

1) 认证的 key 需要和工具匹配, 否则报错 (1.3.1 之后不对 key 版本与 kylinch 版本做校验, 仅确保未过期即可)

```
[root@localhost License]# kylinch --auth KylinchAuthorization.key
[ERROR] 当前工具版本为 1.0.1, 授权版本是 1.0.0, 版本不匹配, 请重新认证.
[INFO] 认证成功.
[root@localhost License]#
```

图 11-kylinch 工具版本与 key 不匹配

2) 如果出现如下提示工具已过期, 请联系工具提供方重新获取 key

```
[root@localhost License]# kylinch
[ERROR] 您的工具认证已过期, 请重新认证.
[root@localhost License]# kylinch --auth KylinchAuthorization.key
[ERROR] 您的工具认证已过期, 请重新认证.
[ERROR] 认证失败.
```

图 12-kylinch 过期

3.5 测试准备

3.5.1 服务端测试准备

1. 执行如下命令检查 netperf 是否正常安装, 如果可以打印工具版本, 则代表安装成功。

netperf -V

```
[root@localhost ~]# netperf -V
Netperf version 2.7.0
```

图 13-netperf 查看版本

2. 执行如下命令检查 kylinch-server 服务是否正常启动

systemctl status kylinch-server.service

```
[root@localhost ~]# systemctl status kylinch-server.service
● kylinch-server.service - Kylin Hardware Compatibility Test Server
   Loaded: loaded (/usr/lib/systemd/system/kylinch-server.service; disabled; vendor preset: disabled)
   Active: active (running) since Fri 2021-08-06 17:41:30 CST; 20h ago
     Process: 165000 ExecStartPre=/usr/share/kylinch/lib/server/kylinch-server-pre.sh (code=exited, status=0/SUCCESS)
    Main PID: 165002 (uwsgi)
      Status: "uWSGI is ready"
        Tasks: 4
       Memory: 52.8M
      CGroup: /system.slice/kylinch-server.service
              └─165002 /usr/local/bin/uwsgi --ini /usr/share/kylinch/lib/server/uwsgi.ini
                └─165005 /usr/local/bin/uwsgi --ini /usr/share/kylinch/lib/server/uwsgi.ini
                  └─165006 /usr/local/bin/uwsgi --ini /usr/share/kylinch/lib/server/uwsgi.ini
                    └─165007 /usr/local/bin/uwsgi --ini /usr/share/kylinch/lib/server/uwsgi.ini
```

图 14-查看 kylinch-server 服务状态

3.5.2 客户端测试准备

1. 执行如下命令检查 netperf、fio、memtester 是否正常安装，如果可以打印工具版本，则代表安装成功。

netperf -V

```
[root@localhost ~]# netperf -V
Netperf version 2.7.0
```

图 15-查看 netperf 版本

fio -V

```
[root@localhost ~]# fio -v
fio-3.20
```

图 16-查看 fio 版本

memtester

```
[root@localhost ~]# memtester
memtester version 4.5.1 (64-bit)
Copyright (C) 2001-2020 Charles Cazabon.
Licensed under the GNU General Public License version 2 (only).

pagesize is 4096
pagesizemask is 0xfffffffffff000
need memory argument, in MB

Usage: memtester [-p physaddrbase [-d device]] <mem>[B|K|M|G] [loops]
```

图 17-查看 memtester 版本

2. kylinch 执行测试用例之前会检查所有待测项的依赖包是否安装，如果没有安装，会使用 yum 安装依赖包。如果有依赖包未安装成功导致退出测试，请手动安装缺失的依赖包。

注意：批量测试时，先检查所有项的依赖包，只有在所有依赖包都安装成功的情况下才会继续测试，否则所有选中的项都会 fail。

缺少依赖包退出测试如下图：

```
Selection (<number>|all|none|quit|run): r
Installing required packages: stress
Error: There are no enabled repositories in "/etc/yum.repos.d", "/etc/yum/repos.d", "/etc/distro.repos.d".
"yum install -y stress" None
Fail to install required packages.
Required rpm package not installed, test stopped.
----- Summary -----
core                                FAIL
system                             FAIL
```

图 18-kylinch 缺少依赖包退出

3. 正式测试之前，针对不同测试项，请参见“表 1 测试用例前置条件”进

行准备工作。也可以测试时在测试项处按‘m’查看测试用例详细信息中的‘前置条件’。

表 1 测试用例前置条件

测试项	测试模式	测试时间	前置条件
整机认证测试集			
acpi	自动	1min	被测机器支持 acpi
boottime	自动	1min	无
cdrom	交互	20min	1.如无内置光驱需要外接光驱测试 2.准备一张可擦写的 CD 或 DVD 进行测试
core	自动	15min	无
cpufreq	自动	20min	1. 存在 /sys/deivces/system/cpu/cpu0/cpufreq 文件 2. CPU 支持调频
disk	交互	100min*被测分区数	至少存在一个可用分区, 需要满足如下条件: 1. 分区不能是 lvm (逻辑分区) 2. 分区不能被挂载 3. 分区大于 2G 4. 不可以是系统分区

filesystem	交互	5min	至少存在可用分区, 需要满足如下条件: 1. 分区大小大于 2G 2. 分区不能是 lvm (逻辑分区) 3. 分区不能被挂载 4. 不可以是系统分区
hardwareinfo	自动	10min	无
infiniband	交互	5min	1. infiniband 卡已安装驱动 2. 待测网口与服务端 infiniband 卡直连并配置 ip
infiniband_netperf	交互	15min	1. infiniband 卡已安装驱动 2. 待测网口与服务端 infiniband 卡直连并配置 ip
ipmi	自动	2min	被测机器支持 ipmi
kdump	交互	15min	被测机器和系统支持 kdump
logicalvolume	交互	5min	至少存在可用分区, 需要满足如下条件: 1. 分区大小大于 2G 2. 分区不能是 lvm (逻辑分区) 3. 分区不能被挂载 4. 不可以是系统分区
memory	自动	40min	无

mobileharddisk	交互	15min	准备移动硬盘
mobilehdhotplug	交互	10min	准备移动硬盘
ethernet	自动	20min	1. 服务端 kylinch-server 服务开启 2. 客户端和服务端关闭防火墙 3. 客户端与服务端直连 4. 客户端与服务端已安装 netperf 5. 待测网口配置好连通服务端的 IP
nvme	自动	20min	无
roce	交互	5min	1. 服务端与客户端都有一张支持 RDMA 功能的以太网卡 2. 待测网口与服务端同型号网卡直连，并配置 ip
perf	自动	5min	perf 版本与 uname -r 一致
system	自动	1min	无
udiskhotplug	交互	5min	准备 U 盘
usb	交互	10min	准备待测 usb 设备，比如 U 盘
usb_keyboard	交互	2min	被测机器需要插入键盘(此测试必须在图形化环境进行)
usb_mouse	交互	2min	被测机器需要插入鼠标(此测试必须在图形化环境进行)

usb_udisk	交互	15min	准备 U 盘
usbcdriverhotplug	交互	10min	准备 usb 光驱设备
video	交互	10min	被测机器需要连接显示器(此测试必须在图形化环境进行)
watchdog	交互	10min	无
virtualization	自动	25min	1. 准备/opt/vmtest 2. 被测机器支持虚拟化
性能测试集			
stream	自动	10min	准备/opt/performance_server
fio	自动	80min	准备/opt/performance_server
unixbench	自动	90min	准备/opt/performance_server
iozone	自动	360min	准备/opt/performance_server 默认测试/,需要确保/空闲磁盘空间大于 2 倍物理内存。或按 5.2.3 修改测试参数或拔掉一些内存条。
lmbench	自动	120min	准备/opt/performance_server
specjvm	自动	600min	准备/opt/performance_server
speccpu	自动	2880min	准备/opt/performance_server 空闲物理内存>2G*cpu 线程数 /home 空闲磁盘空间 > 3G*cpu 线程数
netperf	交互	15min	准备/opt/performance_server

			准备一台服务器作为 netperf 的 server 端，可使用服务端
稳定性测试集			
ltp	自动	48h	准备/opt/performance_server
rebootstress	交互	8h	无
memtester_stress	自动	12h	无

3.6 测试执行

请参照以下步骤开始执行测试：

1. 客户端执行“kylinch”，首次输入需要按照提示信息，依次输入 Test ID，Product URL，Test Server IP。如果输入错误需要修改，请在终端执行“kylinch --clean”清除以下信息（**注意：如果已测试了部分用例，请先打包再 clean，避免测试结果被清除**）

- Test ID：测试任务 ID，用于在服务端 web 查看在线报告
- Product URL：被测产品网站地址，可不填
- Test Server：kylinch 服务端 IP，服务端与被测机器的 IP 必须是连通的（执行 kylinch --resetserverip 可以重新设置服务端 IP）

```
[root@localhost ~]# kylinch
The Kylin Hardware Compatibility Test Suite
Please provide your Compatibility Test ID:0804
Please provide your Product URL:www.kylinos.cn
Please provide the Compatibility Test Server (Hostname or Ipaddr):192.66.1.28
Compatibility Test ID:    0804
Hardware Info:           Suma H520-C30 Dhyana
Product URL:             www.kylinos.cn
OS Info:                 Kylin Linux Advanced Server V10 (Tercel)
Kernel Info:            4.19.90-23.8.v2101.ky10.x86_64
Test Server:            192.66.1.28
```

图 19-首次运行 kylinch

2. 进入测试用例选择与执行界面，在此界面，框架将自动扫描硬件并选取

当前环境可供测试的测试项，每列表示意义如下：

第一列：是否选中

第二列：测试项名称

第三列：测试状态。NotRun、Fail（红色）、Pass（绿色）、Force（必测项-黄色，代表此项之前测试结果为 pass）

当使用移动光驱测试 cdrom，移除光驱后列表中仍保留测试结果，测试项灰色显示，并且无法选中并测试。

第四列：预计测试时间（仅供参考）

第五列：设备（与硬件设备相关的测试项会显示）

第六列：是否为自动项

Name	Status	Estimate	Device	Automated
[X] - 整机认证测试集				
[X] system	Force	1min		yes
[X] acpi	NotRun	1min		yes
[X] core	NotRun	12min		yes
[X] cpufreq	NotRun	20min		yes
[X] ethernet	NotRun	20min	enp5s0f0	yes
[X] ethernet	NotRun	20min	enp5s0f1	yes
[X] ethernet	NotRun	20min	enp5s0f2	yes
[X] ethernet	NotRun	20min	enp5s0f3	yes
[X] infiniband	NotRun	20min		yes
[X] ipmi	NotRun	15min		yes
[X] memory	NotRun	40min		yes
[X] nvme	NotRun	20min		yes
[X] perf	NotRun	5min		yes
[X] video	NotRun	10min	0000:02:00.0	yes
[X] cdrom	NotRun	20min		no
[X] disk	NotRun	100min		no
[X] filesystem	NotRun	5min		no
[X] logicalvolume	NotRun	5min		no
[X] usb	NotRun	10min		no
[X] usb_keyboard	NotRun	2min		no
[X] usb_mouse	NotRun	2min		no
[X] kdump	NotRun	15min		no
[X] watchdog	NotRun	10min		no
[] + 性能测试集(性能测试耗时较长, 请不要和其他测试集一起测试)				

图 20-kylinch 测试用例执行界面

3. 根据需要选择测试项来执行测试，按键说明如下：

- Enter/+/-: 展开/收起测试集
- 空格: 选中/取消选中（测试项开头的[X]代表选中，[]代表未选中
- s/S: 全选
- d/D: 全不选
- q/Q: 退出测试
- r/R: 开始测试
- m/M: 打开测试项详情界面（Esc 退出）
- h/H: 显示帮助信息（Esc 退出）



图 21-kylinch 输入 ‘m’ 显示测试用例详细信息

- 选择完测试项后，输入“R”，开始执行测试。部分测试项需要进行人工交互，请按测试项提示信息进行操作。
- 测试执行完毕后，会提示是否需要收集日志并打包，如果有测试项失败，需要重测，可以在提示打包时输入“n”，或者输入“y”自定义压缩包名并打包日志，等上传测试结果后再合并测试结果。

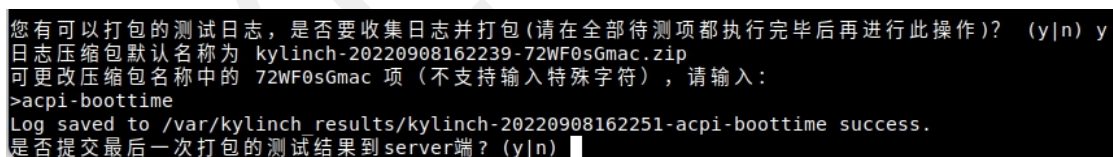


图 22-kylinch 提示是否收集日志并打包

6. 如果对测试结果进行了打包，打包完毕后，会提示是否需要上传测试结果到服务端。输入“y”可以上传测试结果到服务端，在线查看测试结果和详细日志。如果上传失败，请检查网络配置，然后离线上传测试结果，离线上传方式参考 3.7.2。

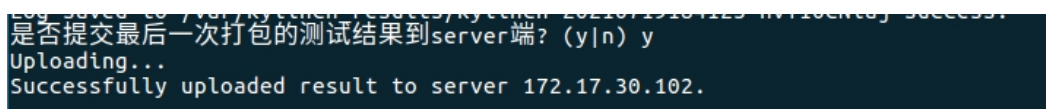


图 23-kylinch 上传测试结果到 server 端

3.7 测试结果

3.7.1 客户端测试结果查看

客户端的日志存放路径如下：

未打包日志存放目录：/usr/share/kylinch/logs/<jobid>，如下图所示：

某一项 log 存放在以测试项命名的.log 文件中，job.log 存放所有的 log。

```
[root@localhost ~]# ll /usr/share/kylinch/logs/t5UYg6yum1/
总用量 2488
-rw-r--r-- 1 root root 596714 8月 7 15:00 acpi.log
-rw-r--r-- 1 root root 245 8月 7 15:11 compatibility.json
-rw-r--r-- 1 root root 1042477 8月 7 15:11 device.json
-rw-r--r-- 1 root root 5418 8月 7 15:11 factory.json
-rw-r--r-- 1 root root 16305 8月 7 15:11 filesystem.log
-rw-r--r-- 1 root root 743783 8月 7 15:11 job.log
-rw-r--r-- 1 root root 130503 8月 7 15:11 system.log
[root@localhost ~]#
```

图 24-客户端未打包日志展示

已打包日志存放目录：/var/kylinch_results/（打包完成后有日志存放路径打印见图 21）可以看到有同名的文件夹和压缩包，文件夹用于本地查看 log，压缩包用于离线上传以及认证测试提供日志。

```
[root@localhost ~]# ls /var/kylinch_results/
kylinch-20210722090045-ftBhXEZCrS  kylinch-20210804172113-LLhRoqfTvp  kylinch-20210804172403-QMP8BfiIyR
kylinch-20210722090045-ftBhXEZCrS.zip  kylinch-20210804172113-LLhRoqfTvp.zip  kylinch-20210804172403-QMP8BfiIyR.zip
[root@localhost ~]# ls /var/kylinch_results/kylinch-20210804172113-LLhRoqfTvp/
compatibility.json  cpufreq.log  device.json  factory.json  job.log  system.log
[root@localhost ~]#
```

图 25-客户端已打包日志展示

3.7.2 服务端测试结果查看

kylinch 的服务端除了与客户端进行网络测试外，还提供 web 功能，用于在线展示上传到服务端的测试结果，并提供测试结果的下载、离线上传等功能。

可以参照如下步骤在线查看测试结果：

1. 浏览器输入“http://<服务端 ip>”，在“测试结果”界面，点击被测

机器信息，找到对应的测试 ID ，进入测试结果查看页面。测试 ID 即客户端执行测试时输入的 Test ID。



图 26-web 页面首页

2. 在测试结果查看页面，可以查看环境信息、执行结果、测试日志、下载附件、查看原始日志、下载 HTML 报告、下载 word 报告等。

Kylin整机认证测试报告

测试机	SpuerCloud-R5215-G11-
ID	1234
Job ID	kylinch-20211213165031-oCmbizfE8V

基本信息 硬件信息 测试日志 下载附件 查看原始日志 下载HTML报告 下载word报告

测试机环境信息

Manufacturer	SpuerCloud
Product Name	R5215 G11
Version	
OS	Kylin Linux Advanced Server V10 (Sword)
kernel	4.19.90-24.4.v2101.ky10.x86_64
ID	1234
Product URL	www.ttyinfo.com
server	172.17.1.99

测试用例	测试设备	测试结果
system		PASS
acpi		PASS
boottime		PASS
core		PASS
cpufreq		PASS
ethernet-em1	em1	PASS
ethernet-em2	em2	PASS
hardwareinfo		PASS
infobased 300-1	300-1	PASS

图 27-web 测试结果查看页面

3. 点击“测试设备”列可查看详细设备信息，点击“PASS”或“FAIL”可以查看详细测试日志

- 硬件信息：查看机器设备信息
- 测试日志：查看测试运行日志
- 下载附件：下载测试日志。**注意不要更改下载附件的 zip 包名称**
- 查看原始日志：显示日志文件夹中的内容
- 下载 HTML 报告：将当前页面结果下载为 HTML 报告。
- 下载 word 报告：自动填充 kylinch 测试结果及性能数据到认证报告模板中，其余信息需要手动填写。

- 显示性能数据：将性能测试结果以表格形式显示（仅性能测试项支持）

KYLINSOFT
麒麟软件

麒麟软件有限公司

测试结果

离线上传

Kylin整机认证测试报告

测试机	Inspur-NF5280M5-00001
ID	2
Job ID	kylinch-20220707094325-plxxOrAvZ1N

基本信息

硬件信息

测试日志

下载附件

查看原始日志

下载HTML报告

下载word报告

显示性能数据

speccpu 测试日志

```

---- start to run test speccpu time: 2022-07-06 17:51:18 ----
[INFO]检查speccpu 工具是否存在...
[INFO] 检测到工具存在
当前配置测试参数:
使用工具版本: speccpu2006-v1.0.1-new-0609.tar
执行脚本参数[server_performance, int all, int single, int multi, fp all, fp single, fp multisingle, multi]: server_performance
最大测试核数（默认满线程）:
测试次数（最大限制3）: 1

-----
是否需要配置测试参数（输入超时: 60 s） (y|n) 请输入使用工具版本(默认speccpu2006-v1.0.1-new-0609.tar): 请输入执行脚本参数[server_performance, int all, int single, int multi, fp all, fp single, fp multisingle, multi]:
设置后配置测试参数:
使用工具版本: speccpu2006-v1.0.1-new-0609.tar
执行脚本参数[server_performance, int all, int single, int multi, fp all, fp single, fp multisingle, multi]: multi
最大测试核数（默认满线程）:
测试次数（最大限制3）: 1

-----
cd /opt/performance_server;./speccpu.sh multi 2>&1
自动化平台读取参数自动执行

.....init.....

用例名: [CPU计算性能]使用speccpu2006测试整型及浮点型计算性能

.....install_rpm.....

[31m libnsl [0m

步骤: [INFO]Install libnsl successfully!

.....environment.....

```

图 28-web 测试结果查看单个测试项

4. 离线上传

如果客户端在测试完成后由于网络或server端服务未开启等原因导致上传测试结果失败，可以使用离线上传功能，将客户端测试日志压缩包上传到服务端并查看。

4 提交认证测试

认证测试完成后需要提交认证测试报告和 kylinch 测试日志到麒麟认证官网。

一、认证测试报告获取及编写

1. 测试完成后在 web 端合并测试结果，下载 word 报告，即可将测试结果与附录二性能测试数据自动填充到 word 报告。
2. 测试人补充 word 报告中的其他信息。由于 kylinch 测试不对性能测试结果做校验，需要人工查看是否有结果缺失。
3. 提交认证测试需要所有测试项 pass，所以要对 N/A 和 Fail 项做出说明。

二、获取 kylinch 测试日志

1. 可以通过服务端 web 页面下载附件或从客户端目录拷贝获取测试日志，客户端测试日志存放在/var/kylinch_results/目录下。
2. zip 压缩包带有密码是正常的。
3. 将所有测试项通过的测试日志压缩包（zip），提交认证。（可以是多份 zip 包）

5 测试项说明

5.1 整机认证测试集

5.1.1 acpi

测试范围：

测试 acpidump

测试流程：

1. 自动列出测试计划；
2. 通过查找“/sys/firmware/acpi/tables”判断机器是否支持 acpi；

3. 执行命令“acpidump”，并打印输出到屏幕。

测试结果：

结果失败：机器不支持 acpi 或 acpidump 命令执行失败。

结果通过：测试过程不出错。

测试时间：

1min

5.1.2 boottime

测试范围：

抓取自 kernel 开始的启动时间

测试流程：

1. 执行命令：systemd-analyze;
2. 筛选启动时间。

测试结果：

结果失败：命令执行失败。

结果通过：命令执行通过。

测试时间：

1min

5.1.3 cdrom

测试范围：

测试设备自带或外接光驱设备。测试光盘格式化、写、读的功能。

测试流程：

注意：测试过程中需要用户交互（根据提示放入光盘，以及关闭光盘仓）

1. 弹出光驱，等待放入光盘；
2. 根据提示放入对应的光盘（可读写的 CD 或 DVD）。测试格式化，写测试，读测试；
3. 格式化光盘：
 - (1) umount 命令卸载光驱设备；
 - (2) 根据光盘类型选择格式化命令：
 - ① BD: dvd+rw-format -format=full [devicename]
 - ② DVD_PLUS: dvd+rw-format -force [devicename]
 - ③ 其他: cdrecord -v dev=[devicename] blank=fast
4. 写测试：
 - (1) umount 命令卸载光驱设备；
 - (2) 生成 iso 文件数据并写入光盘：


```
mkisofs -quiet -R -print-size /usr/share/doc/KylinManual 获取 block
mkisofs -quiet -R /usr/share/doc/KylinManual | cdrecord -v -sao
dev=[devicename] fs=32M tsize=[blocks]s -
```
 - (3) 写入完成与原始数据对比，检查数据一致性；
5. 读测试：
 - (1) umount 命令卸载光驱设备，重新挂载：mount -o ro /dev/sr0 ./mnt_cdrom；
 - (2) 检查光盘是否有数据；
 - (3) 复制光盘文件到本地：cp -dpRf ./mnt_cdrom/. ./device_dir/
 - (4) 将光盘数据与拷贝数据进行对比，检查数据一致性。

测试结果：

结果失败：测试流程中任何子测试项出错则会不能正常进行，最终测试结果则为失败。

结果通过：所有子测试项通过。

测试时间：

20min

5.1.4 core

测试范围：

测试 CPU 时钟准确程度，检测系统 CPU 并且确保 CPU 在大负载下能正常工作。

测试流程：

1. 时钟准确度测试

(1) RTC 硬件时钟基本稳定性测试

获取硬件时钟时间 `rtc_tm1`，睡眠 `sleeptime=120s`，再次获取硬件时钟时间 `rtc_tm2`，验证 `delta=rtc_tm2-rtc_tm1-sleeptime`。

结果失败：`delta` 不等于 0

结果通过：`delta` 等于 0

(2) 时钟抖动测试

抖动是系统时钟之间彼此不同步时出现的一种现象。循环测试 10000 次测试，设定平均抖动时间 `max_ave_jitter`，最大抖动时 `max_adj_jitter` 为 0.2s，每次测试循环计算机内所有运行 CPU，测试每个 CPU 之间的纳秒级时间差，统计最大时间差为 `maxdelta`，平均时间差为 `ave_jitter`。

结果失败：`ave_jitter > 0.2s`，`maxdelta > 0.2s`

结果通过:10000 次循环测试都正确

(3)始终方向测试

记录 starttime 然后睡眠 sleeptime=60s 后记录 stoptime,

$\text{delta} = \text{stoptime} - \text{starttime} - \text{sleeptime}$ 。

结果失败: delta 不等于 0

结果通过: delta 等于 0

2. CPU 负载测试

使用 stress 工具进行 10 分钟压力测试, 命令如下:

```
stress --cpu 12 --io 12 --vm 12 --vm-bytes 128M --timeout 10m
```

结果失败: stress 命令执行失败

结果通过: stress 命令执行成功

测试结果:

结果失败: 有测试项目执行失败

结果通过: 所有测试项目执行成功

测试时间:

15min

5.1.5 cpufreq

测试范围:

主要测试系统的变频功能, 包括 userspace、powersave、performance、ondemand、conservative 模式。

测试流程:

1. 获取 CPU 信息并打印, 包括 CPU 型号、CPU 支持最大频率, CPU 支持

最小频率, CPU 数量, 按拓扑划分的 package, CPU 支持的 governor, CPU flags, CPU 驱动, 当前 governor, 是否支持 aperf。

2. 按照 package 分组 CPU, 分别测试 userspace, ondemand, conservative, powersave, performance 模式。若 cpu 驱动为 intel_pstate 仅支持测试 powersave 和 performance 模式, 预期测试时间比 为 $0.99 \sim 1.2 \times$ 频率比。

3. userspace 模式变频测试:

(1) 设置所有 CPU 为 userspace 模式:

```
cpupower -c [cpu] frequency-set --governor userspace
```

(2) 验证 CPU 模式是否为 userspace:

```
cpupower -c [cpu] frequency-info -p
```

(3) 选择测试 package 中随机 CPU 设置随机频率, 并打印此 CPU 的频

率, 验证 governor 为 userspace;

```
cpupower -c [cpu] frequency-set --freq [random freq]
```

```
cpupower -c [cpu] frequency-info -w
```

```
cpupower -c [cpu] frequency-info -p
```

(4) 设置 package 中所有 CPU 为最低频率;

```
cpupower -c [cpu] frequency-set --freq [min freq]
```

(5) 运行负载测试, 获取平均测试时间 min_freq_average_runtime;

```
taskset -c [cpu] python -u
```

```
/usr/share/kylinch/lib/tests/cpufreq/cal.py
```

(6) 设置 package 中所有 CPU 为最高频;

```
cpupower -c [cpu] frequency-set --freq [max freq]
```

(7) 运行负载测试, 获取平均测试时间 max_freq_average_runtime;

```
taskset -c [cpu] python -u
```

```
/usr/share/kylinch/lib/tests/cpufreq/cal.py
```

(8) 判定: 预期测试时间比 $\text{expect_speed} = \text{max_freq} / \text{min_freq}$

实际测试时间比 $\text{measured} = \text{min_freq_average_runtime} / \text{max_freq_average_runtime}$

允许存在 20% 的误差，即 $\text{expect_speed} * 80\% \leq \text{measured} \leq \text{expect_speed} * 120\%$

4. ondemand 模式变频测试：

- (1) 设置所有 CPU 为 powersave 模式；
- (2) 设置所有 CPU 为 ondemand 模式，并验证；
- (3) 获取 package 的所有 CPU 频率，并打印；
- (4) 运行负载测试，获取 CPU 频率并判定是否为最高频，否则此项测试失败；

- (5) 获取各 CPU 负载测试时间，计算平均值 average_runtime ；
- (6) 如果存在 $\text{userspace_min_freq_runtime}$ ，则进行测试时间判定
实际测试时间比 $\text{measured} = \text{min_freq_average_runtime} / \text{average_runtime}$

允许存在 20% 的误差，即 $\text{expect_speed} * 80\% \leq \text{measured} \leq \text{expect_speed} * 120\%$

5. conservative 模式变频测试：

- (1) 设置所有 CPU 为 performance 模式；
- (2) 设置所有 CPU 为 conservative 模式，并验证；
- (3) 获取 package 的所有 CPU 频率，并打印；
- (4) 运行负载测试，获取 CPU 频率并判定是否在最低频与最高频之间；
- (5) 取各 CPU 负载测试时间，计算平均值 average_runtime ；
- (6) 打印 package 中 CPU 的频率；
- (7) 如果存在 $\text{userspace_min_freq_runtime}$ ，则进行测试时间判定

实际测试时间比 $\text{measured} = \text{min_freq_average_runtime} / \text{average_runtime}$

允许存在 20% 的误差，即 $\text{expect_speed} * 80\% \leq \text{measured} \leq \text{expect_speed} * 120\%$

6. powersave 模式变频测试：

- (1) 设置所有 CPU 为 powersave 模式；
- (2) 验证 package 中 CPU 模式为 powersave，且频率为最低频；
- (3) 运行负载测试，并获取平均测试时间 average_runtime ；
- (4) 打印测试后的 CPU 频率；
- (5) 如果存在 $\text{userspace_max_freq_runtime}$ ，则进行判定：

实际测试时间比 $\text{measured} = \text{average_runtime} / \text{userspace_max_freq_runtime}$

允许存在 20% 的误差，即 $\text{expect_speed} * 80\% \leq \text{measured} \leq \text{expect_speed} * 120\%$

- (6) 如果存在 $\text{userspace_min_freq_runtime}$ ，则进行判定：

$$0.9 \leq \text{average_runtime} / \text{userspace_min_freq_runtime} \leq 1.1$$

7. performance 模式变频测试：

- (1) 设置所有 CPU 为 powersave 模式；
- (2) 设置所有 CPU 为 performance 模式，并验证；
- (3) 获取 package 所有 CPU 频率并打印，检查是否为最大频；
- (4) 运行负载测试，获取平均测试时间 average_runtime ；
- (5) 优先取 powersave 的 runtime，若没有则取 $\text{userspace_min_freq}$ 的 runtime，存在 min_freq_runtime 则进行判定：

实际测试时间比 $\text{measured} = \text{min_freq_runtime} / \text{average_runtime}$

允许存在 20% 的误差，即 $\text{expect_speed} * 80\% \leq \text{measured} \leq$

`expect_speed*120%`

(6) 如果存在 `userspace_max_freq_runtime`，则进行判定：

$$0.9 \leq \text{average_runtime}/\text{userspace_max_freq_runtime} \leq 1.1$$

8. 如果 CPU 支持超频，则测试此项：

选择此 `package` 中任一 CPU 运行负载测试，获取测试时间 `runtime`，判定 `runtime` 需小于 `performance_runtime`。

9. 循环测试 3~7 直到所有 `package` 测试完成。

测试结果：

结果失败：命令执行失败或没有达标

结果通过：所有子测试项执行成功

测试时间：

20min

5.1.6 disk

测试范围：

1. 收集 LVM 磁盘信息，收集设备系统信息，包括内核信息、分区表、交换分区、磁盘状态、分区挂载信息、内核模块信息和 PCI 设备信息；
2. 验证 I/O 子系统数据，并获取性能数据。测试系统所有存储设备。

测试流程：

注意：进行 `disk` 测试之前，需要提前确认系统至少存在一个分区，需要满足不是 `lvm` 分区，分区没有被挂载。

1. 获取磁盘和操作系统信息，包括 LVM 磁盘信息、内核信息、分区表、交换分区、磁盘状态、分区挂载信息、内核模块信息和 PCI 设备信息；

结果失败：有任意命令失败不能正确输出结果

结果通过：所有命令正常运行，有输出结果

2. 查找可用磁盘分区，即分区盘符例如 `sdb1`，不在以下四个文件中

`/proc/swaps /proc/mdstat /etc/mtab /proc/mounts`，若没有可用分区则退出测试，并打印 “[ERROR] No suite disk found to test.”

3. 测试人员选定待测分区，进行 `fio` 测试。若仅有一个可用磁盘分区，无需选择；

4. 下面开始使用独立的测试工具 `fio` 进行硬盘测试：

(1) 使用 `fio` 工具进行 RAW I/O 测试：

`size` 参数用命令 “`cat /sys/block/sdb/size`” 获取，最大为 1GB。块大小取 4K 到 64K 以 2 倍递增（5 个值），对每个块大小进行顺序读写和随机读写测试。共 10 条指令和输出数据；

以 `sdb1` 分区、1G 测试文件、4K 块大小、顺序读写为例：

```
fio -filename=/dev/sdb1 -size=1048576K -bs=4K -direct=1
-iodepth 4 -rw=rw -rwmixread=50 -group_reporting -name=file
-runtime=300
```

(2) 进行 VFS（buffered filesystem）测试：

① 新建文件夹 `vfs_test`

② 将待测分区格式化为 `ext4` 后，挂载至 `vfs_test`

③ 开始 `fio` 测试：

`size` 参数用命令 “`cat /sys/block/sdb/size`” 获取，最大为 1GB。块大小取 4K 到 64K 以 2 倍递增（5 个值），对应每个块大小进行 顺序读写和随机读写测试。共 10 条指令和输出数据；

以 `sdb1` 分区、1G 测试文件、4K 块大小、顺序读写为例：

```
fio -directory=./vfs_test -size=1048576K -bs=4K -direct=1 -iodepth 4
-rw=rw -rwmixread=50 -name=directoy -runtime=300
```

结果失败：fio 测试过程返回值有任意一个非零值则失败

结果通过：fio 测试的返回值均为 0

测试结果：

结果失败：子测试项任意一个失败

结果通过：所有子测试项通过

测试时间：

100min

5.1.7 filesystem

测试范围：

1. 收集 LVM 磁盘信息，收集设备系统信息，包括内核信息、分区表、交换分区、磁盘状态、分区挂载信息、内核模块信息和 PCI 设备信息；
2. 验证分区格式化。

测试流程：

注意：进行 filesystem 测试之前，需要提前确认系统至少存在一个大于 2G 的分区，并且满足两个要求：分区不是 lvm，分区没有被挂载。

1. 获取磁盘和操作系统信息，包括 LVM 磁盘信息、内核信息、分区表、交换分区、磁盘状态、分区挂载信息、内核模块信息和 PCI 设备信息；

结果失败：有任意命令失败不能正确输出结果

结果通过：所有命令正常运行，有输出结果

2. 查找可用磁盘分区，即分区盘符例如 sdb1，不在以下四个文件中
/proc/swaps /proc/mdstat /etc/mtab /proc/mounts，若没有可用分区则退出测

试，并打印 “[ERROR] No suite disk found to test.”

3. 测试人员选定待测分区，进行格式化测试。若仅有一个可用磁盘分区，无需选择；

(1) 使用 mkfs 命令格式化 ext2/ext3/ext4/ntfs/xfs/vfat；

(2) 对格式化后的磁盘，挂载到/mnt；

(3) 检查分区是否格式为 df -T -h /dev/[disk]；

(4) 执行 dd 命令写入 1G 文件：

```
dd if=/dev/zero of=/mnt/test.img bs=1M count=1024
oflag=direct
```

(5) 卸载/mnt；

测试结果：

结果失败：子测试项任意一个失败

结果通过：所有子测试项通过

测试时间：

5min

5.1.8 hardwareinfo

测试范围：

用于抓取系统各硬件信息。

测试流程：

1. 使用命令 cat /etc/os-release, cat /etc/.productinfo, nkvers 抓取系统版本信息；

2. 使用命令 lscpu, /proc/cpuinfo, dmidecode -t processor 抓取系统 cpu

信息；

3. 使用命令 `free, /proc/meminfo, dmidecode -t memory` 抓取系统内存信

息；

4. 使用命令 `fdisk -l, df -h, lsblk, hdparm` 抓取系统磁盘信息，脚本抓取组raid 信息；

5. 使用命令 `dmidecode -t baseboard` 查看主板信息；

6. 使用命令 `lspci` 查看 VGA 信息；

7. 使用命令 `lspci, ifconfig` 查看网卡信息；

8. 使用命令 `lspci` 查看 raid 卡信息；

9. 使用 `dmidecode -t bios` 查看 BIOS 信息；

10. 抓取 `sosreport`；

11. 打印重点关注信息。

测试结果：

结果失败：命令执行返回值非 0

结果成功：命令执行返回值为 0

测试时间：

10min

5.1.9 infiniband

测试范围：

用于测试 infiniband 卡的 RDMA 功能。

测试流程：

1. 打印网口信息，检查服务端 ip 和待测网口 IP，关闭非待测网口。

- (1) 打印待测网口信息，包括 ethtool 信息，mac 地址，biosdevname -d，

网口驱动信息

```
# ifconfig
# ethtool <interface>
# ip link show <interface>
# biosdevname -d
# ethtool -i <interface>
```

- (2) 获取待测网口 ip，ping 服务端 ip 确保网络联通

- (3) 关闭非待测网口，ip link set down <other interface>

- (4) 获取待测网口速率

```
# ethtool <interface>
```

2. 检查 ibstatus

- (1) 开启 opensm 服务

- (2) 加载 ib_umad 模块

- (3) 获取 ibstatus

3. 检查 IB 卡连接状态与 id

4. icmp（丢包率）测试，ping -q -c 500 -i 0 server_ip，没有丢包，则测试

通过，若有丢包则尝试重测，最多重测 3 次：

```
# ping -q -c 5000 -i 0 <server_ip>
```

5. rdma 测试：

- (1) rping 测试：

- ① 在服务端开启 rping：# rping -s

- ② 客户端执行命令：# rping -c -a <server_ip> -C 50 -v

- ③ 验证客户端命令返回值为 0

- (2) rcopy 测试

- ① 创建测试文件：# dd if=/dev/urandom of=./testfile bs=128k

count=speed/8

- ② 在服务端开启 rcopy: # rcopy
- ③ 客户端执行命令: # rcopy testfile <server_ip>:/tmp/testfile
- ④ 验证客户端命令返回值为 0
- ⑤ 杀死服务端 rcopy 进程

(3) ib_read_bw 测试

- ① 在服务端开启服务: # ib_read_bw
- ② 客户端执行测试命令: # ib_read_bw <server_ip> -d <ib_device> -i<ib_port>
- ③ 获取测试带宽并将单位转为 Mb/s, 检查测试带宽是否达到网口速率的 90%

(4) ib_write_bw 测试

- ① 在服务端开启服务: # ib_write_bw -F -s 65536 -n 25000 -m 2048
- ② 客户端执行测试命令: # ib_write_bw -F -s 65536 -n 25000 -m 2048 <server_ip> -d <ib_device> -i<ib_port>
- ③ 获取测试带宽并将单位转为 Mb/s, 检查测试带宽是否达到网口速率的 90%

(5) ib_send_bw 测试

- ① 在服务端开启服务: # ib_send_bw
- ② 客户端执行测试命令: # ib_send_bw <server_ip> -d <ib_device> -i<ib_port>
- ③ 获取测试带宽并将单位转为 Mb/s, 检查测试带宽是否达到网口速率的 90%

6. 清理测试环境, 恢复端口连接。

测试结果：

结果失败：有命令执行失败，测试带宽不达标。

结果通过：所有命令执行成功，并且测试带宽达标。

测试时间：

5min

5.1.10 infiniband_netperf

测试范围：

使用 netperf 工具测试 infiniband 卡。

测试流程：

1. 打印网口信息，检查服务端 ip 和待测网口 IP，关闭非待测网口。
- (1) 打印待测网口信息，包括 ethtool 信息，mac 地址，biosdevname -d，

网口驱动信息

```
# ifconfig
# ethtool <interface>
# ip link show <interface>
# biosdevname -d
# ethtool -i <interface>
```

- (2) 获取待测网口 ip，ping 服务端 ip 确保网络联通
- (3) 关闭非待测网口，ip link set down <other interface>
- (4) 获取待测网口速率

```
# ethtool <interface>
```

2. netperf 测试，最多重试 3 次：

- (1) 在服务端执行 # netserver，开启 netserver 服务
 - (2) tcp stream 测试，带宽达到待测端口速率的 90%，则结果通过；
- ```
netperf -H <server_ip> -t TCP_STREAM -l 120
```

(3) tcp rr 测试，有测试结果打印，则结果通过；

```
netperf -H <server_ip> -t TCP_RR -l 120 -r 32,1024
```

(4) tcp crr 测试，有测试结果打印，则结果通过；

```
netperf -H <server_ip> -t TCP_CRR -l 120
```

(5) udp stream 测试，有测试结果打印，error 为 0，则结果通过；

```
netperf -H <server_ip> -t UDP_STREAM -l 120
```

(6) udp rr 测试，有测试结果打印，则结果通过；

```
#netperf -H <server_ip> -t UDP_RR -l 120
```

(7) omni 测试，有测试结果打印，则结果通过。

```
#netperf -H <server_ip> -t omni -- -d rr -l %u -O
"THROUGHPUT,THROUGHPUT_UNITS,MIN_LATENCY,MAX_LATENCY,M
EAN_LATENCY"
```

3. 清理测试环境，恢复端口连接。

### 测试结果：

结果失败：有命令执行失败，测试带宽不达标。

结果通过：所有命令执行成功，并且测试带宽达标。

### 测试时间：

15min

### 5.1.11 ipmi

#### 测试范围：

用于测试操作系统与 BMC 兼容性，打印 fru 和传感器信息。

#### 测试流程：

1. 检查是否存在 /dev/ipmi0 /dev/ipmi/0 /dev/ipmidev/0，若没有以上文件则加载模块：ipmi\_watchdog ipmi\_poweroff ipmi\_devintf ipmi\_msghandler ipmi\_si

2. 开启 ipmi 服务并检查 ipmi 服务状态；

```
systemctl start ipmi
```

3. 执行命令 `ipmitool mc info`，打印 BMC 版本信息，检查命令是否执行成功并且输出信息中是否存在 `Firmware Revision`；
4. 执行命令 `ipmitool fru print 0`，打印内建的 Field Replaceable Unit (FRU) 信息，检查命令是否执行成功，返回代码是否为 0；
5. 执行命令 `ipmitool sensor`，打印详细的传感器信息，检查命令是否执行成功并且输出信息中是否存在 `CPU`；
6. 执行命令 `ipmitool lan print 1 test`，打印 BMC IP 信息，检查命令是否执行成功并且输出信息中存在 “IP Address :”，验证 IP Address 值是 ip；
7. 执行命令 `ipmitool sel list`，抓取 BMC 日志信息；
8. 执行命令 `ipmitool sdr elist`，抓取 BMC 各 sensor 信息，检查命令是否执行成功；
9. 执行命令 `ipmitool mc reset cold`，在系统下进行 BMC 冷重启，验证命令执行成功，输出信息中有打印 “Sent cold reset command to MC”；

#### 测试结果：

结果失败：服务开启失败，命令执行返回值非 0

结果成功：服务开始成功，命令执行返回值为 0

#### 测试时间：

2min

### 5.1.12 kdump

#### 测试范围：

测试系统的 kdump 的本地转存能力以及生成 crash 文件的准确性。

注意：需要 kexec-tools

## 测试流程：

1. 检查/proc/cmdline 设置了 crashkernel;
2. 修改/etc/kdump.conf, 将 path 改为/var/crash, 如果有"kdump\_obj  
kbox"则改为"kdump\_obj all", 修改完成后重启 kdump 服务并检查 kdump 服务  
状态为开启:

```
path /var/crash
kdump_obj all
```

3. 询问是否重启;
  - > 是: 继续执行
  - > 否: 测试失败, 退出测试
4. 执行以下命令触发崩溃, 抓取 vmcore:
 

```
sync;echo c >/proc/sysrq-trigger;
```
5. 开机后先检查 last reboot 自开始测试此项到当前时间仅存在一条记录;
 

```
last reboot -s '<自开始测试此项到当前时间间隔>s ago'
```
6. 验证/var/crash 目录下是否有 vmcore 文件。

## 测试结果：

结果失败: 没有设置 crashkernel, 开启 kdump 服务失败, 没有生成 vmcore  
文件

结果通过: 可以正常模拟 kdump 并能生成 vmcore 文件

## 测试时间：

15min

### 5.1.13 loigcalvolume

## 测试范围：

1. 收集 LVM 磁盘信息，收集设备系统信息，包括内核信息、分区表、交换分区、磁盘状态、分区挂载信息、内核模块信息和 PCI 设备信息；
2. 创建逻辑卷，删除逻辑卷测试。

### 测试流程：

注意：进行 logicalvolume 测试之前，需要提前确认系统至少存在一个大于 2G 的分区，并且满足两个要求：分区不是 lvm，分区没有被挂载。

1. 获取磁盘和操作系统信息，包括 LVM 磁盘信息、内核信息、分区表、交换分区、磁盘状态、分区挂载信息、内核模块信息和 PCI 设备信息；

结果失败：有任意命令失败不能正确输出结果

结果通过：所有命令正常运行，有输出结果

2. 查找可用磁盘分区，即分区盘符例如 sdb1，不在以下四个文件中 /proc/swaps /proc/mdstat /etc/mtab /proc/mounts，若没有可用分区则退出测试，并打印 “[ERROR] No suite disk found to test.”

3. 测试人员选定待测设备，进行逻辑卷测试。若仅有一个可用磁盘分区，无需选择；

4. 对选择分区执行以下操作，以 sdb 分区为例

(1) 创建逻辑卷；

```
mkfs -t ext4 -F /dev/sdb
pvcreate --force /dev/sdb
pvdisplay
vgcreate test /dev/sdb
vgdisplay
lvcreate -L 1G -n lv1 test
```

(2) 查看逻辑卷；

```
lvdisplay
```

(3) 删除逻辑卷；



```
lvremove --force /dev/test/lv1
vgremove /dev/test
pvremove /dev/sdb
```

### 测试结果：

结果失败：子测试项任意一个失败

结果通过：所有子测试项通过

### 测试时间：

5min

## 5.1.14 memory

### 测试范围：

主要测试系统的 RAM 和 swap 分区。

注意：需要安装 libhugetlfs-utils 用于大页内存测试

### 测试流程：

1. 通过/proc/meminfo 获取内存相关信息：总内存（System Memory）、空闲内存（Free Memory）、可用内存（Available Memory）、交换分区（Swap Memory）、HugePages Total、HugePages Free、HugePages Size；

#### 2. 内存读写测试

(1) 获取测试内存 test\_mem=空闲内存\*90/100，最大为 4G；

(2) 执行命令测试：

```
memtester test_memM 1。
```

#### 3. 内存压力测试

(1) 打印内存信息；

```
System Memory = MemTotal
Free Memory = MemFree + Cached + Buffers
Available Memory = MemAvailable
```

Swap Memory = SwapTotal

要求 Swap Memory 大于 512M

(2) 计算测试内存:

free\_mem=free Memory 和 available memory 相比取较小值

extra\_mem = 取 test\_mem/100 ,swap memory/2,512M 三个值中的最小值;

test\_mem = free\_mem + extra\_mem

(3) 执行内存测试脚本做进程加压测试

# ./eatmem\_test -m test\_memM

测试时间 120s, 创建 num\_cpus\*2 数量的线程用来加压, 每个线程映射的内存大小为 test\_mem/( num\_cpus\*2 ), 最后返回一个值判定是否测试通过;

#### 4. 大页内存测试

(1) 打印大页内存信息;

HugePages Total= HugePages\_Total

HugePages Free = HugePages\_Free

HugePages Size = Hugepagesize

(2) 大页内存信息校验:

如果 HugePages Size 大于 512M 则 huge\_pages = 10 否则

huge\_pages = 1000

如果没有 HugePages Total 值需要手动创建:

# hugeadm --create-mounts

# hugeadm --pool-pages-min <hugepages size>MB:<huge\_pages>

如果 HugePages Free 小于 huge\_pages, 需要执行以下命令:

# hugeadm --pool-pages-min <hugepages size>MB:<HugePages Total> + <huge\_pages>

要求 HugePages Free 大于 huge\_pages / 2

### (3) 执行测试：./hugettlb\_test

#### 测试结果：

结果失败：存在测试子项测试失败，返回值为非 0

结果通过：所有测试子项通过测试，返回值为 0

#### 测试时间：

40min

### 5.1.15 mobileharddisk

#### 测试范围：

用于测试移动硬盘的识别、读写、卸载功能。

#### 测试流程：

1. 按照提示在测试前移除待测移动硬盘；
2. 提示连接移动硬盘，连接后按 y 继续测试；
3. 获取连接的移动硬盘盘符并打印 lsblk；
4. 卸载移动硬盘，格式化为 ext4 格式；
5. 挂载移动硬盘至/mnt；
6. 执行 dd if=/dev/zero of=/mnt/test.img bs=1M count=1024

oflag=direct 写入 1G 文件，展示写入速率；

7. 执行 dd if=/mnt/test.img of=/dev/null bs=1M 读出 1G 文件到内存，展示读取速率；

8. 执行 umount 卸载移动硬盘。

#### 测试结果：

结果失败：移动硬盘未连接，识别移动硬盘超时，命令执行返回值非 0

结果成功：命令执行返回值为 0

测试时间：

15min

### 5.1.16 mobilehdhotplug

测试范围：

用于测试移动硬盘热插拔功能。

测试流程：

1. 按照提示在测试前移除待测移动硬盘；
2. 提示将已连接的移动硬盘移除，按“y”继续测试；
3. 提示连接移动硬盘，连接后按“y”继续测试；
4. 识别是否有移动硬盘连接，超时时间 15s：
  - 是：打印 lsblk
  - 否：退出测试
5. 提示移除第 2 步连接的移动硬盘，按“y”继续测试；
6. 检查移动硬盘是否移除，超时时间 15s：
  - 是：打印 lsblk
  - 否：退出测试
7. 循环第 2 步到第 5 步 10 次。

测试结果：

结果失败：移动硬盘识别失败

结果成功：所有子项测试通过

测试时间：

10min

### 5.1.17 ethernet

#### 测试范围：

测试系统所有的以太网卡设备。

注意：需要服务端和客户端安装 netperf 用于 netperf 测试

#### 测试流程：

1. 检查已获取待测网口，服务端 ip 和待测网口 IP。打印待测网口信息，包括 ethtool 信息，mac 地址，biosdevname -d，网口驱动信息

```
ifconfig
ethtool <interface>
ip link show <interface>
biosdevname -d
ethtool -i <interface>
```

2. 使用 ifconfig 和 ip link set 测试端口断开连接，确认能够连接服务端并

获取测试网口速度；

(1) 关闭非待测网口，ip link set down <other interface>

(2) 待测网口断开连接：

```
ip link set down <interface>
```

(3) 待测网口恢复连接：

```
ip link set up <interface>
```

(4) 待测网口断开连接：

```
ifdown <interface>
```

(5) 待测网口恢复连接

```
ifup <interface>
```

(6) ping 服务端 ip

```
ping -I <interface> -Ac1 <server_ip>
```

(7) 获取待测网口速度

```
ethtool <interface>
```

3. icmp (丢包率) 测试, `ping -q -c 500 -i 0 server_ip`, 没有丢包, 则测试通过, 若有丢包则尝试重测, 最多重测 3 次:

```
ping -q -c 5000 -i 0 <server_ip>
```

4. netperf 测试, 最多重试 3 次:

(1) 在服务端执行 `# netserver`, 开启 netserver 服务

(2) tcp stream 测试, 带宽达到待测端口速度的 90%, 则结果通过;

```
netperf -H <server_ip> -t TCP_STREAM -l 120
```

(3) tcp rr 测试, 有测试结果打印, 则结果通过;

```
netperf -H <server_ip> -t TCP_RR -l 120 -r 32,1024
```

(4) tcp crr 测试, 有测试结果打印, 则结果通过;

```
netperf -H <server_ip> -t TCP_CRR -l 120
```

(5) udp stream 测试, 有测试结果打印, error 为 0, 则结果通过;

```
netperf -H <server_ip> -t UDP_STREAM -l 120
```

(6) udp rr 测试, 有测试结果打印, 则结果通过;

```
netperf -H <server_ip> -t UDP_RR -l 120
```

(7) omni 测试, 有测试结果打印, 则结果通过。

```
netperf -H <server_ip> -t omni -- -d rr -l %u -O
"THROUGHPUT,THROUGHPUT_UNITS,MIN_LATENCY,MAX_LATENCY,MEAN_LATENCY"
```

5. http 测试, 主要测试文件的上传及下载的时间及速率, 以及验证下载文件与本地文件的一致性;

(1) 创建测试文件

```
dd if=/dev/urandom of=testfile bs=128k count=<网口速度除 8>
```

(2) 上传测试文件到 kylinch 服务端;

(3) 下载测试文件到客户端并打印下载时间及速率;

(4) 验证上传文件与下载文件一致性;

6. 清理测试环境, 恢复端口连接。

## 测试结果：

结果失败：获取信息失败，netperf 运行失败，丢包率不达标，http 上传下载失败

结果通过：所有信息收集正确且带宽和丢包率达标

## 测试时间：

20min \* 测试端口

### 5.1.18 nvme

#### 测试范围：

1. 收集 LVM 磁盘信息，收集设备系统信息，包括内核信息、分区表、交换分区、磁盘状态、分区挂载信息、内核模块信息和 PCI 设备信息；
2. 使用 nvme 命令测试 nvme 磁盘的格式化、读、写、抓取 log 功能。

注意：需要安装依赖 nvme-cli

#### 测试流程：

注意：机器上有 nvme 磁盘才会加载此测试项。

1. 确认待测设备未挂载可使用，即待测设备不在以下四个文件中  
/proc/swaps /proc/mdstat /etc/mtab /proc/mounts，若没有可用的 nvme 磁盘则打印“磁盘 is in use now,skip this test”

2. 磁盘格式化测试

```
nvme format -l 0 -i 0 /dev/<nvme 盘符>
```

3. 磁盘写测试

```
nvme write -z <size> -s 0 -d /dev/urandom /dev/<nvme 盘符>
```

size 等于/sys/block/<nvme 盘符>/size 文件值除 4，并且 size 最大值为

10\*1024\*1024\*1024

#### 4. 磁盘读测试

```
nvme read -s 0 -z <size> /dev/<nvme 盘符>
```

#### 5. 磁盘抓取 log 测试

```
nvme smart-log /dev/<nvme 盘符>
```

```
nvme get-log -i 1 -l 128 /dev/<nvme 盘符>
```

#### 6. 打印 nvme 磁盘信息

```
nvme list
```

测试结果：

结果失败：待测设备不可用或命令执行失败

结果通过：待测设备可用且命令执行成功

测试时间：

20min

### 5.1.19 roce

测试范围：

对支持 RDMA 功能的以太网卡，测试其 RDMA 功能。

测试流程：

1. 打印网口信息，检查服务端 ip 和待测网口 IP，关闭非待测网口。

(1) 打印待测网口信息，包括 ethtool 信息，mac 地址，biosdevname -d，

网口驱动信息

```
ifconfig
ethtool <interface>
ip link show <interface>
biosdevname -d
ethtool -i <interface>
```

(2) 获取待测网口 ip，ping 服务端 ip 确保网络联通



(3) 关闭非待测网口，ip link set down <other interface>

(4) 获取待测网口速率

```
ethtool <interface>
```

2. 检查 ibstatus

(1) 开启 opensm 服务

(2) 加载 ib\_umad 模块

(3) 获取 ibstatus

3. 使用 ifconfig 和 ip link set 测试端口断开连接，确认能够连接服务端并

获取测试网口速度；

(1) 待测网口断开连接：

```
ip link set down <interface>
```

(2) 待测网口恢复连接：

```
ip link set upl <interface>
```

(3) 待测网口断开连接：

```
ifdown <interface>
```

(4) 待测网口恢复连接

```
ifup <interface>
```

(5) ping 服务端 ip

```
ping -l <interface> -Ac1 <server_ip>
```

4. icmp（丢包率）测试，ping -q -c 500 -i 0 server\_ip，没有丢包，则测试

通过，若有丢包则尝试重测，最多重测 3 次：

```
ping -q -c 5000 -i 0 <server_ip>
```

5. rdma 测试：

(1) rping 测试：

① 在服务端开启 rping：# rping -s

② 客户端执行命令：# rping -c -a <server\_ip> -C 50 -v

- ③ 验证客户端命令返回值为 0

## (2) rcopy 测试

- ① 创建测试文件：`# dd if=/dev/urandom of=./testfile bs=128k count=speed/8`
- ② 在服务端开启 rcopy：`# rcopy`
- ③ 客户端执行命令：`# rcopy testfile <server_ip>:/tmp/testfile`
- ④ 验证客户端命令返回值为 0
- ⑤ 杀死服务端 rcopy 进程

## (3) ib\_read\_bw 测试

- ① 在服务端开启服务：`# ib_read_bw -R`
- ② 客户端执行测试命令：`# ib_read_bw -R <server_ip> -d <ib_device> -i<ib_port>`
- ③ 获取测试带宽并将单位转为 Mb/s，检查测试带宽是否达到网口速率的 90%

## (4) ib\_write\_bw 测试

- ① 在服务端开启服务：`# ib_write_bw -F -s 65536 -n 25000 -m 2048 -q 64 -R`
- ② 客户端执行测试命令：`# ib_write_bw -F -s 65536 -n 25000 -m 2048 -q 64 -R <server_ip> -d <ib_device> -i<ib_port>`
- ③ 获取测试带宽并将单位转为 Mb/s，检查测试带宽是否达到网口速率的 90%

## (5) ib\_send\_bw 测试

- ① 在服务端开启服务：`# ib_send_bw -R`
- ② 客户端执行测试命令：`# ib_send_bw -R <server_ip> -d`

<ib\_device> -i<ib\_port>

③ 获取测试带宽并将单位转为 Mb/s，检查测试带宽是否达到网口速率的 90%

6. 清理测试环境，恢复端口连接。

#### 测试结果：

结果失败：有命令执行失败，测试带宽不达标。

结果通过：所有命令执行成功，并且测试带宽达标。

#### 测试时间：

5min

### 5.1.20 perf

#### 测试范围：

测试 perf 工具是否能够正常使用。

注意：需要安装 perf

#### 测试流程：

注意：perf 版本要和内核版本一致。

1. 验证 perf 版本与内核版本一致；
2. 检查循环事件
 

```
perf record -a -e cycles -o hwcompatible-perf.data sleep 5
```
3. 检查是否检测到 cycles 事件，执行以下命令检查返回值是否包括 cycles
 

```
perf evlist -i hwcompatible-perf.data
```
4. 检查样本是否被收集

执行以下命令，输出日志正常

```
perf report -i hwcompatible-perf.data --stdio。
```

#### 测试结果：

结果失败：有命令执行失败

结果通过：所有命令执行成功

**测试时间：**

5min

### 5.1.21 system

**测试范围：**

测试并收集系统信息：工具是否被修改、os 版本与内核是否匹配、内核是否被修改。

**测试流程：**

1. 打印内核、模块、dmidecode 信息；
2. 检查本工具是否被修改；
3. 检查 OS 版本和 kernel 版本是否匹配；
4. 检查内核是否被修改/污染。

**测试结果：**

结果失败：所有信息收集中任意一个失败。

结果通过：所有信息收集正确，内核与工具未被修改。

**测试时间：**

1min

### 5.1.22 udiskhotplug

**测试范围：**

测试 U 盘热插拔功能。

**测试流程：**

1. 按照提示将待测 U 盘移除，按“y”继续测试；

2. 按照提示连接待测 U 盘，连接后按 “y” 继续测试；
3. 识别是否有 U 盘连接：
  - 是：打印 lsblk
  - 否：退出测试
4. 提示移除第 2 步连接的 U 盘，按 “y” 继续测试；
5. 检查 U 盘是否移除：
  - 是：打印 lsblk
  - 否：退出测试
6. 循环第 2 步到第 5 步 10 次。

#### 测试结果：

结果失败：U 盘识别失败

结果通过：所有子项测试通过

#### 测试时间：

5min

### 5.1.23 usb

#### 测试范围：

测试 USB 端口热插拔功能，保证 USB 设备能够在系统下正常使用。

#### 测试流程：

1. 按照提示在测试前移除待测试 usb 设备；
2. 打印 lsblk 和 lsusb -t 并从 udevadm info --export-db 获取所有 usb 设备；
3. 按照提示选择一个 USB 端口插入待测试的 USB 设备：
  - 是：继续下一步

➤ 否：重复 3

4. 检测到新增 USB 设备, 打印 `lsusb -t` 和 `lsblk`, 如果没有检测到新增 usb 设备则退出测试;

5. 按照提示拔出第 3 步插入的 USB 设备:

➤ 是：继续下一步

➤ 否：重复 5

6. 检测到 USB 设备拔出, 打印 `lsusb -t` 和 `lsblk`, 如果没有检测到 USB 设备被拔出则退出测试;

7. 询问所有端口是否已测试 (需要测试除连接鼠标和键盘外的所有 usb 端口):

➤ 是：退出

➤ 否：重复 2-7

**测试结果:**

结果失败: 没有检查到 usb 设备连接或拔出

结果通过: 所有项测试通过

**测试时间:**

10min

#### **5.1.24 usb\_keyboard**

**测试范围:**

测试 usb 键盘输入, 保证键盘能够在系统下正常使用。

注意: 需要图形界面, 测试机需要连接显示器测试。

**测试流程:**

1. 提示插入键盘：
  - 是：继续下一步
  - 否：重复 1
2. 在弹框里输入字符；
3. 验证输入与显示一致，结果通过。

#### 测试结果：

结果失败：键盘输入字符错误

结果通过：键盘输入字符正确

#### 测试时间：

2min

### 5.1.25 usb\_mouse

#### 测试范围：

测试 usb 鼠标点击，保证鼠标能够在系统下正常使用。

注意：需要图形界面，测试机需要连接显示器测试。

#### 测试流程：

1. 提示插入鼠标：
  - 是：继续下一步
  - 否：重复 1
2. 点击弹框中的按钮关闭弹框。

#### 测试结果：

结果失败：通过其他方式关闭弹框

结果通过：点击按钮关闭弹框

测试时间：

2min

### 5.1.26 usb\_udisk

测试范围：

测试 U 盘识别、读写、格式化、卸载功能。

测试流程：

1. 按照提示移除待测 U 盘设备；
2. 按照提示连接移动硬盘，按 “y” 继续测试；
3. 获取连接的 U 盘盘符并打印 lsblk 和 lsusb -t；
4. 卸载 U 盘，格式化为 ext4 格式；  
  
# mkfs -t ext4 -F <u 盘>
5. 挂载 U 盘至/mnt；  
  
# mount <u 盘> /mnt
6. 检查 U 盘是否格式化为 ext4；  
  
# df -T -h <u 盘>
7. 写 1G 文件，并打印写入速率  
# dd if=/dev/zero of=/mnt/test.img bs=1M count=1024 oflag=direct
8. 读 1G 文件到内存，并打印读取速率  
# dd if=/mnt/test.img of=/dev/null bs=1M
9. 在 U 盘挂载目录，创建 test 文件夹并删除；  
# cd /mnt  
# mkdir test  
# rm -rf test
10. 执行 umount 卸载 U 盘。

测试结果：



结果失败：U 盘未识别到或命令执行返回值非 0

结果通过：命令执行返回值 0

**测试时间：**

15min

### 5.1.27 usbcddriverhotplug

**测试范围：**

测试 usb 光驱热插拔功能。

**测试流程：**

1. 按照提示移除待测 usb 光驱，按“y”继续测试；
2. 提示连接 usb 光驱，连接后按“y”继续测试；
3. 识别是否有 usb 光驱连接：
  - > 是：打印 lsblk
  - > 否：退出测试
4. 提示移除第 2 步连接的 usb 光驱，按“y”继续测试；
5. 检查 usb 光驱是否移除：
  - > 是：打印 lsblk
  - > 否：退出测试
6. 循环第 2 步到第 5 步 10 次。

**测试结果：**

结果失败：usb 光驱识别失败

结果通过：所有子项测试通过

**测试时间：**

10min

### 5.1.28 video

#### 测试范围：

测试视频连接，xserver 服务运行。

需要 xorg-x11-utils mesa-demos 用于测试

注意：需要图形界面，测试机需要连接显示器测试。

#### 测试流程：

注意：此项测试不能通过远程 ssh 连接测试。

1. 测试显示器连接执行 xrandr 命令；

# xrandr

结果失败：xrandr 命令执行失败

结果通过：xrandr 命令执行成功

2. 齿轮测试 5 次；

# glxgears -display :0

3. 查看模块。

(1) 打开日志文件 /var/log/Xorg.0.log

(2) 找到所有包含“(II) Loading /”的行尾的模块文件，(II) Loading 后是模块文件

(3) 验证模块文件

# rpm -Vf <模块文件>

4. 记录驱动信息

(1) 打开日志文件 /var/log/Xorg.0.log

(2) 找到包含“/usr/lib64/xorg/modules/drivers”的行，打印这行及下面几行直到空行或遇到第二个包含“(II)”的行

(3) 执行以下命令：

```
xinfo
xdriinfo
glxinfo
```

测试结果：

结果失败：有测试项目执行失败

结果通过：所有测试项目执行成功

测试时间：

10min

### 5.1.29 virtualization

测试范围：

测试虚拟机基本功能以及抓取虚拟机基本信息。

测试流程：

1. 检查系统是否支持虚拟化，不支持则退出；
2. 检查是否存在测试文件，不存在则退出；
3. 开启虚拟机并获取虚拟机 IP；
4. 宿主机与虚拟机 ping 测试；
5. 抓取虚拟机实际 CPU、内存、磁盘是否与 xml 设置一致；
6. 抓取虚拟机 sosreport；
7. 挂起虚拟机；
8. 恢复虚拟机；
9. 关闭虚拟机；
10. 删除虚拟机域。

测试结果：

结果失败：任一测试项执行失败

结果通过：所有测试项执行通过

测试时间：

25min

### 5.1.30 watchdog

测试范围：

触发 watchdog，测试系统是否可以正常复位。

测试流程：

1. 检查有无/dev/watchdog 文件；
  - 有，下一步
  - 无，加载 softdog 模块
2. 查看当前 watchdog 超时时间；
 

```
wdctl
```
3. 如果超时时间大于 20s 则设置超时时间为 20s；
 

```
wdctl -s 20
```
4. 如果是前台进程，询问是否重启，否则直接测试 watchdog；
  - 是：执行测试
  - 否：退出并测试失败
5. 测试 watchdog
 

```
sync
echo a > /dev/watchdog
```
6. 触发 watchdog 超时等待 60 秒，若超时未触发机器重置则测试失败；
7. 重启后检查 last reboot 自开始测试此项到当前时间仅存在一条记录；
 

```
last reboot -s '<自开始测试此项到当前时间间隔>s ago'
```

8. 检查重启是否成功。

- 是，打印“Recover from watchdog”到日志
- 否，退出测试

**测试结果：**

结果失败：触发 watchdog 超时，检查重启失败

结果通过：所有测试项执行成功

**测试时间：**

10min

## 5.2 性能测试集

### 5.2.1 stream

**测试范围：**

使用 stream 测试工具测试内存带宽。

**测试流程：**

1. 检查工具是否存在，不存在直接退出；
2. 打印默认配置参数信息，并询问是否需要调整，超时时间 60s:

```

---- start to run test stream time: 2021-09-02 15:35:56 ----
[INFO]check stream tools exists...
[INFO] 检测到工具存在
当前配置测试参数:
使用工具版本: stream-5.9-1.tar.bz2
测试次数: 1
编译参数ntimes: 100
编译参数array_size: 80000000
测试最大线程: 满线程

是否需要配置测试参数 (输入超时: 60 s) (y|n) y
请输入使用工具版本(默认stream-5.9-1.tar.bz2):
请输入测试次数(默认1): 3
请输入编译参数ntimes(默认100):
请输入编译参数array_size(默认80000000):
请输入测试最大线程(默认满线程): 8

设置后配置测试参数:
使用工具版本: stream-5.9-1.tar.bz2
测试次数: 3
编译参数ntimes: 100
编译参数array_size: 80000000
测试最大线程: 8

cd /opt/performance_server;./stream.sh.x -server_performance 2>&1

```

图 29-stream 配置参数

- 是：进入参数配置页面，完成后打印配置后的参数
- 否：使用默认参数测试
- 超时：使用默认参数测试

参数包括：

- 1) 使用工具：str 类型，工具包名称
- 2) 测试次数：int 类型，用于多次测试
- 3) 编译参数 ntimes：int 类型，用于编译，取测试 ntimes 次最优值
- 4) 编译参数 array\_size：int 类型，用于编译至少为 4 倍 L3 缓存
- 5) 测试最大线程：int 类型，默认测试单线程和满线程，此处可以设置测试满线程为指定值。

### 3. 开始测试 stream;

4. 测试完成打印 json 格式信息；
5. 拷贝测试日志、测试结果到 log 存储路径的 stream 文件夹下。

#### 测试结果：

结果失败：工具不存在，执行测试失败

结果通过：所有测试项执行成功

#### 测试时间：

10min

### 5.2.2 fio

#### 测试范围：

使用 fio 测试工具测试磁盘读写速率。

#### 测试流程：

1. 检查工具是否存在，不存在直接退出；
2. 打印默认配置参数信息，并询问是否需要调整，超时时间 60s:
  - 是：进入参数配置页面，完成后打印配置后的参数
  - 否：使用默认参数测试
  - 超时：使用默认参数测试

参数包括：

- 1) 使用工具版本：str 类型，工具包名称
- 2) 测试线程数：int 类型，默认为机器核数，限制最大为 16
- 3) 队列深度：int 类型，默认 32
- 4) 测试路径：str 类型，默认测试在根目录/fio\_test
- 5) 测试次数：int 类型，默认 3，用于测试多次取平均值

3. 开始测试 fio;
4. 测试完成打印 json 格式测试结果信息;
5. 拷贝测试日志、测试结果到 log 存储路径的 fio 文件夹下。

#### 测试结果:

结果失败: 工具不存在, 执行测试失败

结果通过: 所有测试项执行成功

#### 测试时间:

80min

### 5.2.3 iozone

#### 测试范围:

使用 iozone 测试工具测试磁盘读写速率。

#### 测试流程:

1. 检查工具是否存在, 不存在直接退出;
2. 打印默认配置参数信息, 并询问是否需要调整, 超时时间 60s:
  - > 是: 进入参数配置页面, 完成后打印配置后的参数
  - > 否: 使用默认参数测试
  - > 超时: 使用默认参数测试

参数包括:

- 1) 执行脚本参数: str 类型, 默认 r16ms2
  - a) desktop\_performance 测试 16M 块大小文件大小分别为 2 倍内存、1 倍内存、0.5 倍内存的随机读写、顺序读写速率。



b) `server_performance` 默认测试 2 倍内存，块范围 64K 到 16M

下的随机读写、顺序读写。

c) `m2test` 同上

d) `r64kt4` 测试命令 `iozone -s 1g -r 64k -i 0 -i 1 -t 4 -F /iozone1  
/iozone2 /iozone3 /iozone 4`

e) `r64kt1` 测试命令 `iozone -s 1g -r 64k -i 0 -i 1 -f /iozone`

f) `r16ms2` 测试 16M 块大小，`-s` 参数为 2 倍内存的随机读写、顺序读写速率

2) 使用工具版本: `str` 类型，工具包名称

3) 测试路径: `str` 类型，默认根目录/

4) 测试次数: `int` 类型，默认 1

3. 如果执行参数为 `server_performance` 或 `m2test` 则会触发此参数下的参数配置，打印默认配置参数，并询问是否需要配置：

- 是：进入参数配置页面，完成后打印配置后的参数
- 否：使用默认参数测试

参数包括：

1) 全面测试块范围最小值: `str` 类型，默认 64K

2) 全面测试块范围最大值: `str` 类型，默认 16M

3) 全面测试文件范围最小值: `str` 类型，默认 2 倍内存，需要带单位，  
例如 10g

4) 全面测试文件范围最大值: `str` 类型，默认 2 倍内存，需要带单位，  
例如 20g

4. 开始测试 iohome;
5. 测试完成打印 json 格式测试结果信息;
6. 拷贝测试日志、测试结果到 log 存储路径的 iohome 文件夹下。

#### 测试结果:

结果失败: 工具不存在, 执行测试失败

结果通过: 所有测试项执行成功

#### 测试时间:

360min

### 5.2.4 Imbench

#### 测试范围:

使用 Imbench 测试工具测试系统性能。

#### 测试流程:

1. 检查工具是否存在, 不存在直接退出;
2. 打印默认配置参数信息, 并询问是否需要调整, 超时时间 60s:
  - > 是: 进入参数配置页面, 完成后打印配置后的参数
  - > 否: 使用默认参数测试
  - > 超时: 使用默认参数测试

参数包括:

- 1) 使用工具版本: str 类型, 工具包名称
- 2) 测试内存: int 类型, 默认 1000, 单位 M, 限制最大值为 16384
3. 开始测试 Imbench;
4. 测试完成打印 json 格式测试结果信息;

5. 拷贝测试日志、测试结果到 log 存储路径的 Imbench 文件夹下。

#### 测试结果：

结果失败：工具不存在，执行测试失败

结果通过：所有测试项执行成功

#### 测试时间：

120min

### 5.2.5 netperf

#### 测试范围：

使用 netperf 测试工具测试网络性能。

#### 测试流程：

1. 检查工具是否存在，不存在直接退出；
2. 打印默认配置参数信息，并询问是否需要调整：
  - 是：进入参数配置页面，完成后打印配置后的参数
  - 否：使用默认参数测试

#### 参数包括：

- 1) 使用工具版本：str 类型，工具包名称
- 2) 执行脚本参数：不可输入多个参数
  - a) server：测试  
tct\_stream,udp\_stream,tcp\_rr,udp\_rr,trans\_rate,omni,tcp\_crr
  - b) tcp\_stream：测试命令 netperf -H <serverip> -t TCP\_STREAM -l <time>
  - c) udp\_stream：测试命令 netperf -H <serverip> -t UDP\_STREAM -l <time>

- d) tcp\_rr: 测试命令 netperf -H <serverip> -t TCP\_RR -l <time>
  - e) udp\_rr: 测试命令 netperf -H <serverip> -t UDP\_RR -l <time>
  - f) ping\_sh: 测试命令 ping <serverip> -c 10
  - g) trans\_rate: 测试命令 netperf -H <serverip> -l <time>
  - h) omni: 测试命令 netperf -H <serverip> -t omni -- -d rr -l  
<time> -O  
"THROUGHPUT,THROUGHPUT\_UNITS,MIN\_LATENCY,MAX\_LATENCY,MEAN\_LATENCY"
  - i) tcp\_crr:测试命令 netperf -H <serverip> -t TCP\_CRR -l <time>
- 3) 测试时间: int 类型, 默认 120, 单位 s
  - 4) server 端 IP: 需要输入 ip, 默认为 kylinch 服务端 ip
  - 5) server 端用户名: str 类型, 默认 root
  - 6) server 端密码: str 类型, 默认 123456
  - 7) server 端安装目录: str 类型, 默认/home/kylin, 用于在 server 端安装 netperf

```

---- start to run test netperf time: 2021-09-02 19:32:35 ----
[INFO] 测试前请确保已连接服务端且关闭服务端的防火墙
[INFO]check netperf tools exists...
[INFO] 检测到工具存在
当前配置测试参数:
使用工具版本: netperf-2.7.0.zip
执行脚本参数[server tcp_stream udp_stream tcp_rr udp_rr ping_sh trans_rate omni tcp_crr]: server
测试时间-单位s: 120
server端IP: 192.66.1.234
server端用户名: root
server端密码: 123456
server端脚本安装目录: /home/kylin

是否需要配置测试参数(输入超时: 0 s) (y|n) y
请输入使用工具版本(默认netperf-2.7.0.zip):
请输入执行脚本参数[server tcp_stream udp_stream tcp_rr udp_rr ping_sh trans_rate omni tcp_crr](默认server): tcp_rr
请输入测试时间-单位s(默认120): 60
请输入server端IP(默认192.66.1.234):
请输入server端用户名(默认root):
请输入server端密码(默认123456): qwer13
请输入server端脚本安装目录(默认/home/kylin): /opt

设置后配置测试参数:
使用工具版本: netperf-2.7.0.zip
执行脚本参数[server tcp_stream udp_stream tcp_rr udp_rr ping_sh trans_rate omni tcp_crr]: tcp_rr
测试时间-单位s: 60
server端IP: 192.66.1.234
server端用户名: root
server端密码: qwer13
server端脚本安装目录: /opt

```

图 30-netperf 参数配置

### 3. 开始测试 netperf;

4. 测试完成打印 json 格式测试结果信息；
5. 拷贝测试日志、测试结果到 log 存储路径的 netperf 文件夹下。

#### 测试结果：

结果失败：工具不存在，执行测试失败

结果通过：所有测试项执行成功

#### 测试时间：

15min

### 5.2.6 speccpu

#### 测试范围：

使用 speccpu 测试工具测试 CPU 性能。

#### 测试流程：

1. 检查工具是否存在，不存在直接退出；
2. 打印默认配置参数信息，并询问是否需要调整：
  - 是：进入参数配置页面，完成后打印配置后的参数
  - 否：使用默认参数测试

参数包括：

- 1) 使用工具版本：str 类型，工具包名称
- 2) 测试核数：int 类型，默认满线程，此参数为设置多线程测试的核数
- 3) 测试次数：int 类型，默认 1
3. 开始测试 speccpu；
4. 测试完成打印 json 格式测试结果信息；
5. 拷贝测试日志、测试结果到 log 存储路径的 speccpu 文件夹下。

## 测试结果：

结果失败：工具不存在，执行测试失败

结果通过：所有测试项执行成功

## 测试时间：

2880min

## 5.2.7 specjvm

### 测试范围：

使用 specjvm 测试工具测试 java 基准性能。

### 测试流程：

1. 检查工具是否存在，不存在直接退出；
2. 打印默认配置参数信息，并询问是否需要调整：
  - 是：进入参数配置页面，完成后打印配置后的参数
  - 否：使用默认参数测试

参数包括：

- 1) 执行脚本参数：str 类型，默认 all
  - a) base：测试命令 `java -jar SPECjvm2008.jar -ikv -base`
  - b) peak：测试命令 `java -Xms1024m -Xmx<2/3 内存>m -jar SPECjvm2008.jar -ikv -peak -Dspecjvm.benchmark.threads=<CPUN>`
  - c) all：包括 base 和 peak
3. 开始测试 specjvm；
4. 测试完成打印 json 格式测试结果信息；
5. 拷贝测试日志、测试结果到 log 存储路径的 specjvm 文件夹下。

## 测试结果：

结果失败：工具不存在，执行测试失败

结果通过：所有测试项执行成功

## 测试时间：

600min

## 5.2.8 unixbench

### 测试范围：

使用 unixbench 测试工具测试系统性能。

### 测试流程：

1. 检查工具是否存在，不存在直接退出；
2. 打印默认配置参数信息，并询问是否需要调整：
  - 是：进入参数配置页面，完成后打印配置后的参数
  - 否：使用默认参数测试

参数包括：

- 1) 执行脚本参数：str 类型，默认 base
  - a) single：测试命令 Run -c 1
  - b) multi：测试命令 Run -c <CPUN>
  - c) base：测试命令 Run -c 1 -c <CPUN>
  - d) graphics：Run graphics
  - e) all：包括 base、graphics
- 2) 使用工具版本：str 类型，工具包名称
- 3) 测试线程数：int 类型，默认为 CPU 核数，仅对 multi 和 base 参数

的 CPUN 生效

3. 开始测试 unixbench;
4. 测试完成打印 json 格式测试结果信息;
5. 拷贝测试日志、测试结果到 log 存储路径的 unixbench 文件夹下。

**测试结果:**

结果失败: 工具不存在, 执行测试失败

结果通过: 所有测试项执行成功

**测试时间:**

90min

### 5.3 稳定性测试集

#### 5.3.1 ltp

**测试范围:**

使用 ltp 测试工具测试系统稳定性。

**测试流程:**

1. 检查工具是否存在, 不存在直接退出;
2. 打印默认配置参数信息, 并询问是否需要调整:
  - > 是: 进入参数配置页面, 完成后打印配置后的参数
  - > 否: 使用默认参数测试

参数包括:

- 1) 测试时间参数, int 类型, 默认 48, 单位小时
3. 开始测试 ltp;
4. 测试完成打印 ltp 每项测试结果信息;



5. 拷贝测试日志、测试结果到 log 存储路径的 ltp 文件夹下。

#### 测试结果：

结果失败：工具不存在，执行测试失败

结果通过：所有测试项执行成功

#### 测试时间：

48h

### 5.3.2 rebootstress

#### 测试范围：

系统重启压力测试。

#### 测试流程：

1. 打印默认配置参数信息，并询问是否需要调整：
  - 是：进入参数配置页面，完成后打印配置后的参数
  - 否：使用默认参数测试

参数包括：

- 1) 测试次数参数，int 类型，默认 100
2. 清除系统日志 messages 和 dmesg
3. 抓取系统磁盘、网卡、内存信息；
4. 如果是前台进程，询问是否重启，否则直接重启：
  - 是：执行测试
  - 否：退出并测试失败
5. 重启后检查是否重启成功，硬件信息是否与重启前抓取一致，不一致则

退出测试；

6. 全部测试完成后，查看并收集系统日志，无 error 或 fail 信息。

#### 测试结果：

结果失败：触发重启失败，重启前后硬件信息不一致。

结果通过：所有测试项执行成功

#### 测试时间：

8h

### 5.3.3 memtester\_stress

#### 测试范围：

使用 memtester 工具对内存进行 12 小时压力测试

#### 测试流程：

1. 打印系统内存信息
2. 执行压力测试./memtester memM ( mem=空闲内存的 85%，单位：M )
3. 12 小时后，自动停止测试

#### 测试结果：

结果失败：工具不存在，执行测试失败

结果成功：对内存进行 12 小时压力测试，无报错，无宕机

## 6 FAQ

### 6.1 Perf 测试 FAIL

报错信息：“[ERROR] perf 版本与内核版本不一致,无法测试 perf,请更新 perf”

## 解决方案：

1. `uname -r` 查看内核版本，`rpm -q perf` 查看 `perf` 版本；
2. `yum list perf --showduplicates` 列出源里所有 `perf` 版本
3. 卸载 `perf` 包  
# `rpm -e perf`
4. # `yum upgrade/downgrade perf-<内核版本>` 升级或降级 `perf` 版本
5. 若 `base` 源中的 `perf` 仍与内核版本不一致，需要挂载系统镜像，安装镜像中的 `perf`

像中的 `perf`

挂载系统镜像 `iso` 或者镜像光盘或系统启动盘，在 `Packages` 文件夹下执行

如下命令安装：

```
rpm -ivh perf-4.19.90-*.rpm
```

## 6.2 `kdump`、`watchdog`、`rebootstress` 测试后没有自动打包测试日志

这三项涉及重启，需要等待测试完成后在终端输入“`kylinch`”进行日志打包上传。

## 6.3 测试 `fail` 查看报错信息

1. 如果 `fail` 项包含测试子项，可查看对应测试项 `log` 最底部的 `subtest`，如下图：

|          |                |      |
|----------|----------------|------|
| -----    |                |      |
| SubTest: | check certrpm  | FAIL |
| SubTest: | check kernel   | FAIL |
| SubTest: | check selinux  | PASS |
| SubTest: | check cpuinfo  | PASS |
| SubTest: | check meminfo  | PASS |
| SubTest: | check VGA      | PASS |
| SubTest: | check ethernet | PASS |

图 31-测试子项

可以看到具体失败的测试子项（有些测试项比较简单，不包括测试子项，则不显示 SubTest）

2. 在日志中搜索[ERROR]可以看到错误输出，如下图

```
[+] Checking installed cert package...
S.5..... /usr/share/kylinch/lib/tests/cpufreq/cpufreq.so
[ERROR] Files in kylin-ch have been tampered.

[+] Checking kernel...
Kernel RPM: kernel-4.19.90-22.3.ky10.mips64el-rt35
OS Version: Kylin Linux Advanced Server V10 (Tercel)
Detailed OS Version: Kylin Linux Advanced Serverrelease V10 (SP1) /(Tercel)-mips64el-Build08/20201030

[INFO] Checking debug kernel...
[INFO] 内核未受到污染
[ERROR] Files from kernel-4.19.90-22.3.ky10.mips64el-rt35 were modified.

Boot Parameters: root=/dev/sda3 ro rhgb quiet console=tty quiet crashkernel=512M rd_start=0xffffffff887b006
```

图 32-测试日志中的报错信息

## 6.4 Intel cpufreq 测试问题

**问题描述：**cpufreq 测试 pass 但是日志中 powersave 模式和 performance 模式的测试时间几乎一样。

**解决方案：**

1. 查看 cpufreq 驱动，可以通过查看 log 或执行命令的方式查看

### cpufreq 测试日志

```
---- start to run test cpufreq ----

CPU Info#####
CPU Model : Loongson-3A R4 (Loongson-3B4000)
CPU Max Frequency : 1800000
CPU Min Frequency : 900000
CPU Nums : 8
CPU Package 0 : [0, 1, 2, 3]
CPU Package 1 : [4, 5, 6, 7]
CPU Available Governors : ['conservative', 'ondemand', 'userspace', 'powersave', 'performance']
CPU Flags : None
CPU Driver : loongson3
```

图 33-测试日志查看 cpu 驱动

或者执行命令：

```
cpupower -c 0 frequency-info
```

```
[root@localhost ~]# cpupower -c 0 frequency-info
analyzing CPU 0:
driver: acpi-cpufreq
CPUs which run at the same hardware frequency: 0
CPUs which need to have their frequency coordinated by software: 0
maximum transition latency: Cannot determine or is not supported.
hardware limits: 1.60 GHz - 2.50 GHz
available frequency steps: 2.50 GHz, 2.00 GHz, 1.60 GHz
available cpufreq governors: conservative ondemand userspace powersave performance
current policy: frequency should be within 1.60 GHz and 2.50 GHz.
 The governor "performance" may decide which speed to use
 within this range.
current CPU frequency: 2.50 GHz (asserted by call to hardware)
boost state support:
 Supported: no
 Active: no
 Boost States: 0
```

图 34-命令查看 cpu 驱动

2. 如果 driver 是 intel\_pstate, 那么此项 pass。因为使用 intel\_pstate 驱动调频时 powersave 模式更类似于 ondemand 模式, 频率不会保持最低, 会随着 cpu 压力而自行调节, 导致加压时 powersave 模式和 performance 模式的测试时间几乎一样。
3. 如果 driver 不是 Intel\_pstate, 那么这是一个 bug。

## 6.5 海光测试 cpufreq fail 问题

kylinch 1.4.0 版本之前由于海光 CPU 获取到的最大频率为主频, 且支持超频, 导致测试时执行速度比超过工具允许范围。此为工具 bug, 1.4.0 修改。

此项测试需要通过人工检查测试数据判定。

## 6.6 终端打开 kylinch 提示没有需要运行的测试集, 退出测试。

问题描述:

安装 kylin-ch 后, 终端运行 kylinch, 提示“没有需要运行的测试集, 退出测试”如下图

```
[root@localhost ~]# kylinch
The Kylin Hardware Compatibility Test Suite
Please provide your Compatibility Test ID:00
Please provide your Product URL:kylin
Please provide the Compatibility Test Server (Hostname or Ipaddr):192.66.1.1
 Compatibility Test ID: 00
 Hardware Info: GreatWall 擎天DF723 NA
 Product URL: kylin
 OS Info: Kylin Linux Advanced Server V10 (Tercel)
 Kernel Info: 4.19.90-23.13.v2101.ky10.aarch64
 Test Server: 192.66.1.1

[ERROR]没有需要运行的测试集，退出测试。
[root@localhost ~]#
```

图 35-没有需要运行的测试集

### 解决方案：

没有安装 kylin-ch-testsuites 导致此问题，安装后就可以加载出测试集了。

## 6.7 测试完成后无法上传日志到 **server** 端

问题描述：客户端能够 ping 通 server 端 ip，客户端或其他机器无法打开 server 端网页。

解决方案：可能是由于 server 端重启过导致服务未开启或防火墙未关闭，手动

执行以下命令：

```
systemctl start kylinch-server
systemctl start nginx
systemctl stop firewalld
iptables -F
```

## 6.8 测试 **kdump** 或 **watchdog** 或 **rebootstress** 失败

问题描述：测试 kdump、watchdog、rebootstress 时，log 打印只到重启前，或 log 打印到 last reboot

解决方案：

1. 查看 last reboot
2. 查看重启时间是否正确，若与系统时间相差 8 小时请按如下步骤操作
  - (1) 终端输入 `# timedatectl`，查看输出的 RTC in local TZ
  - (2) 若此项为 yes 则执行命令 `timedatectl set-local-rtc 0`
  - (3) 再次查看 `# timedatectl`，查看输出的 RTC in local TZ: no
  - (4) 清理 last reboot 日志：
 

```
echo >/var/log/wtmp
```

## 6.9 cpufreq 测试 fail，log 显示无法获取 cpu 频率信息

问题描述：

“[ERROR] 无法获取 cpu 频率信息，请检查 CPU 是否支持调频或者 BIOS 中是否开启调频。”

解决方案：

需要在 BIOS 中开启相应设置，loongarch 3C5000 暂不支持调频。

## 6.10 ipmi 测试项中 check reboot log 子项测试失败

问题描述：日志中显示 bmc cold reboot 项测试 pass，check reboot log 项 fail，报错“没有检测到新增日志”

解决方案：

因为 bmc 的固件不同，有些不记载 bmc 重启日志到 ipmitool sel list 中  
kylinch 1.4.0 去掉了 bmc 冷重启后日志检查。

## 6.11 ipmi 测试项中 fru 子项测试失败

问题描述：

日志 ipmi.log 中显示如下

"ipmitool fru" None

[ERROR] 执行命令 ipmitool fru print 0 失败

解决方案:

可能是 bmc 硬件没有刷 fru, 需要设置 bmc 验证是否有 fru 信息

```
yum install OpenIPMI ipmitool
```

```
systemctl start ipmi
```

```
ipmitool lan set 1 ipsrc static ##设置静态
```

```
ipmitool lan set 1 ipaddr 110.204.3.81 ##设置 ip
```

```
ipmitool lan set 1 netmask 255.255.248.0 ## 设置掩码
```

```
ipmitool lan set 1 defgw ipaddr 110.204.7.254 ##设置网关
```

```
ipmitool user set name 2 aaaaaa ## 设置用户
```

```
ipmitool user set password 2 'aaaaaa' ##设置密码
```

```
ipmitool user list ## 查看管理口用户
```

```
ipmitool lan print 1 ## 查看管理口配置情况
```

## 6.12 V10SP1 系统 Intel 第三代 CPU icelake 测试 virtualization

**fail**

问题描述:

日志显示如下:

[INFO] 检查系统是否支持 HVM(全虚拟化)

[INFO] 系统不支持虚拟化测试

查看 virsh capabilities 打印只有 host 没有 guest

```
systemctl stop libvirtd
```

```
systemctl start libvirtd
```



安装 qemu 后，重启 libvirtd 服务后 guest 出现，但使用 virsh start 开启虚拟机报错

报错信息 1:

```
[root@localhost ~]# virsh define kylin10.0-x86-vmx.xml
定义域 kylin10.0-x86-vmx (从 kylin10.0-x86-vmx.xml)

[root@localhost ~]# virsh start kylin10.0-x86-vmx
错误: 开始域 kylin10.0-x86-vmx 失败
错误: 内部错误: qemu unexpectedly closed the monitor: 2022-04-07T07:42:20.029120Z qemu-kvm: can't apply global Icelake-Server-
x86_64-cpu.pconfig=off: Property '.pconfig' not found
```

图 36-virsh start 报错信息 1

解决方案:

更新 libvirt 版本至 libvirt-5.5.0-6.p01.ky10.x86\_64.rpm 或更新至最新版本。

```
yum list libvirt --showduplicates
yum upgrade libvirt-5.5.0-6.p01.ky10
```

报错信息 2: 缺少 libreadline.so.8

```
[INFO] 检查系统是否支持HVM(全虚拟化)
virsh capabilities
virsh: error while loading shared libraries: libreadline.so.8: cannot open shared object file: No such file or directory
"virsh capabilities" None
[ERROR] Could not parse virsh capabilities
[ERROR] virsh capabilities没有guest或者guest中os_type不是hvm
```

图 37-virsh start 报错信息 2

解决方案:

/lib64/libreadline.so.8 由 readline 提供，更新 readline 至 8.0 以上可以解决此问题。

```
yum upgrade readline
```

报错信息 3:

```
- kylin10.0-x86-vmx shut off
[INFO] 开启虚拟机
error: Failed to start domain kylin10.0-x86-vmx
error: internal error: qemu unexpectedly closed the monitor: 2022-04-26T09:56:59.59423
1Z qemu-kvm: -machine pc-i440fx-4.1,accel=kvm,usb=off,vmpport=off,dump-guest-core=off:
unsupported machine type
Use -machine help to list supported machines

[ERROR] 开启虚拟机失败
---- stop to run test virtualization time: 2022-04-26 17:56:59 ----
[INFO] 清理测试环境
```

图 38-virsh start 报错信息 3

解决方案:

修改/opt/vmtest/\*.xml 文件中

```
<os>
 <type arch='x86_64' machine='pc-i440fx-4.1'>hvm</type>
 <boot dev='hd'/>
</os>
```

为:

```
<os>
 <type arch='x86_64' machine='pc-i440fx-4.0'>hvm</type>
 <boot dev='hd'/>
</os>
```

### 6.13 系统不带 X710 网卡驱动，安装第三方驱动导致 system 测试 fail

问题描述:

V10SP1 和 V10SP2 服务器操作系统开机无法加载 X710 网卡

解决方法:

V10 SP1 更新内核至 23.18 及更高版本

V10 SP2 更新内核至 25.9 及更高版本

1. 检查 yum repo 配置文件，确保 update 源 enabled=1，如果 enabled =

0 请先改为 1，并执行 yum makecache。

```
cat /etc/yum.repos.d/kylin_x86_64.repo
###Kylin Linux Advanced Server 10 - os repo###
```

```
[ks10-adv-os]
```

```
name = Kylin Linux Advanced Server 10 - Os
```

```
baseurl
```

```
https://update.cs2c.com.cn/NS/V10/V10SP3/os/adv/lic/base/$basearch/
```

```
gpgcheck = 1
```

```
gpgkey=file:///etc/pki/rpm-gpg/RPM-GPG-KEY-kylin
```

```
enabled = 1
```

```
[ks10-adv-updates]
```

```
name = Kylin Linux Advanced Server 10 - Updates
```

```
baseurl
```

```
https://update.cs2c.com.cn/NS/V10/V10SP3/os/adv/lic/updates/$basearch/
```

```
gpgcheck = 1
```

```
gpgkey=file:///etc/pki/rpm-gpg/RPM-GPG-KEY-kylin
```

enabled = 1

## 2. 更新内核

# rpm -qa |grep kernel // 列出当前内核相关包名

# yum upgrade <内核包名> //依次更新内核相关包

## 3. 重启测试机

### 6.14 SP2 系统测试 Imbench 编译失败，日志中显示缺少 rpc.h

问题描述：

Imbench 测试失败，查看日志显示编译失败，找不到 rpc.h

解决方法：

1. 执行 rpm -qa|grep libtirpc 检查 libtirpc-devel 是否安装
2. 未安装执行 yum install libtirpc-devel-1.2.6-2.ky10 安装
3. 检查是否存在 rpc.h # find /usr -name rpc.h

### 6.15 已知问题

机型	问题描述	问题原因
CPU: ZHAOXIN KaiSheng KH-37800D	perf 测试 fail: The sys_perf_event_open() syscall returned with 16 (设备或资源忙) for event (cycles).	系统原因， V10SP1-0518 版本 还未合入修改。 25.10 内核已合入修 改
CPU:龙芯 3B4000	cpufreq 测试 fail: package0 测试 pass,	CPU 设计如此， 3B4000 不支持除

	package1 测试 fail, package1 的低频与高频的 测试时间相同	package 0 以外 cpu 调频
CPU:龙芯 3B4000	acpi 测试 fail	机器不支持 acpi 测 试
CPU:龙芯 3B4000	kdump 测试 fail: kdump 服 务开启失败	机器不支持 kdump
泰山 200 服务器 CPU: kunpeng920	watchdog 测试失败, 无法 触发重启	BIOS 问题 环境中执行 rmmod sbsa_gwdt; modprobe sbsa_gwdt action=1 之后测试 可以通过
CPU:Loongson-3C5000L	cpufreq fail	缺少 cpupower 命令 CPU 固件不支持获 取 cpu 频率等信息
CPU:Loongson-3C5000L	disk fail 没有/proc/mdstat	需要手动加载 md-mod 模块 # modprobe md-mod
CPU:Loongson-3C5000L	filesystem fail	同上
CPU:Loongson-3C5000L	logicalvolume fail	同上

CPU:Loongson-3C5000L	nvme fail	同上
CPU:Loongson-3C5000L	video fail	enabled update 源, 更新 mesa* # yum install mesa* sp1 更新至 18.3.6-3.p02.a 及以上 版本 sp3 更新至 20.1.4-1.p05 及以上 版本
CPU:Loongson-3C5000L	kdump fail	需要安装捕获内核, 手动测试。
华诚金锐服务器 CPU: SW3231 CPU @ 2.40GHz	acpi 测试 fail	机器不支持 acpi 测试
华诚金锐服务器 CPU: SW3231 CPU @ 2.40GHz	core 测试 fail	缺少 stress 软件包
华诚金锐服务器 CPU: SW3231 CPU @ 2.40GHz	cpufreq 测试 fail	缺少 kernel-tools 软件包
华诚金锐服务器	hardwareinfo 子测试项 get	系统下没有 nkvers

CPU: SW3231 CPU @ 2.40GHz	OS info 测试 fail	命令
华诚金锐服务器 CPU: SW3231 CPU @ 2.40GHz	kdump 测试 fail	缺少 kexec-tools 软件包
华诚金锐服务器 CPU: SW3231 CPU @ 2.40GHz	perf 测试 fail	缺少 perf 软件包
华诚金锐服务器 CPU: SW3231 CPU @ 2.40GHz	system 测试 fail	uname -r 与 rpm -q kernel 查到的 kernel 版本不一致
华诚金锐服务器 CPU: SW3231 CPU @ 2.40GHz	virtualization 测试 fail	1. 无法使用 virt-manager 安装 虚拟机  2. virsh shutdown xxx 命令无法关闭虚拟机