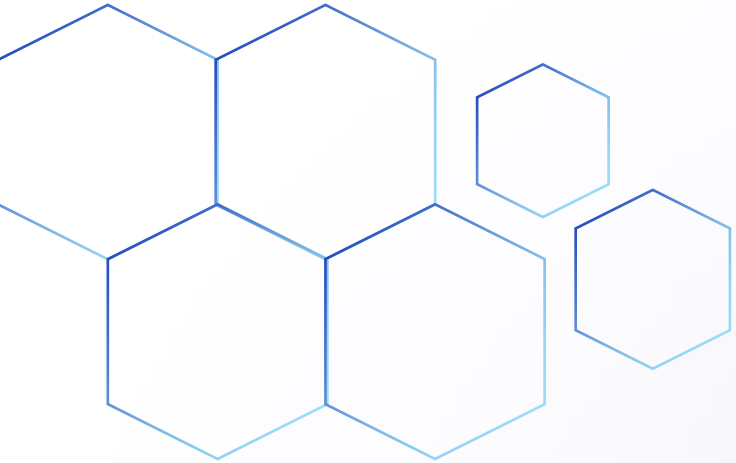


精灵云

New Computing & New Era

Ghostcloud

New Computing & New Era

Linux容器技术介绍

2023年4月13日

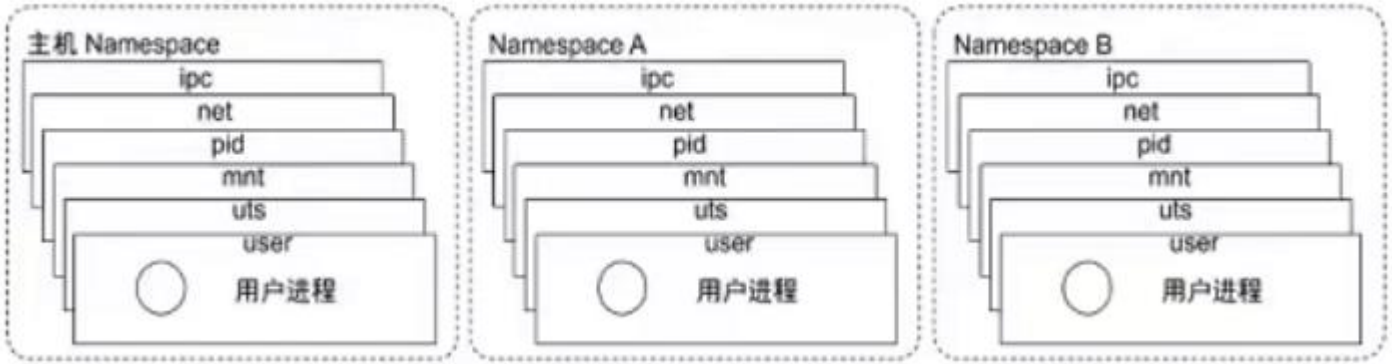
虚拟化技术

虚拟化技术是一种将计算机物理资源进行抽象、转换为虚拟的计算机资源提供给程序使用的技术。这里所指的计算机资源，就包括了 CPU 提供的运算控制资源，硬盘提供的数据存储资源，网卡提供的网络传输资源等。

在虚拟化技术的发展过程中，人们逐渐发现了虚拟化的另一大用途，也就是将之应用于计算机资源的管理。

从虚拟化实现的角度可以大致将虚拟化分为硬件虚拟化和软件虚拟化，从架构角度又可以将虚拟化分为裸金属架构和寄居架构，分类的方式多种多样。

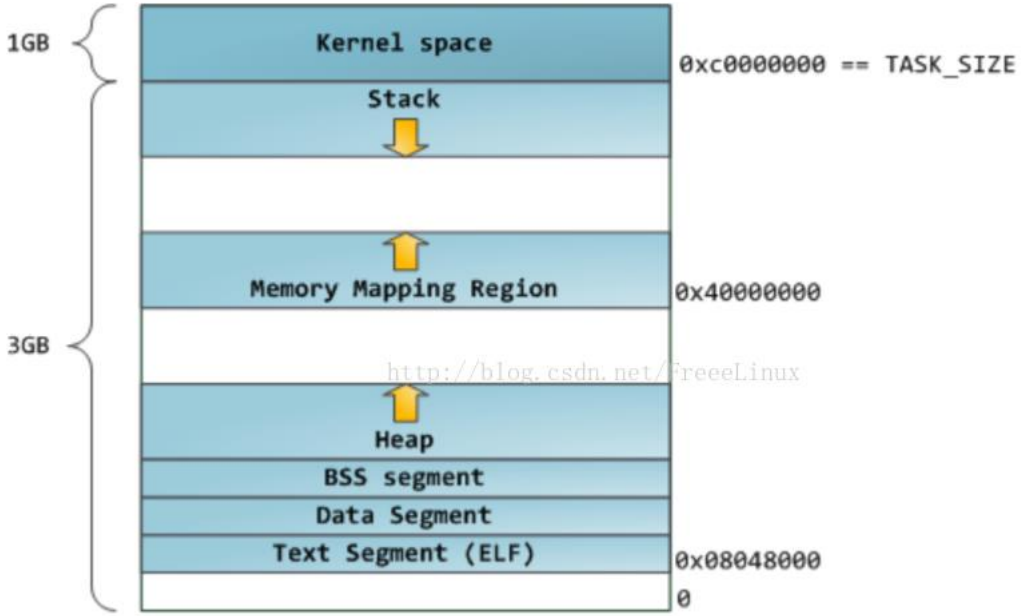
Linux容器相关的虚拟化包括：CPU虚拟化，内存虚拟化，中断虚拟化和设备虚拟化，等等。



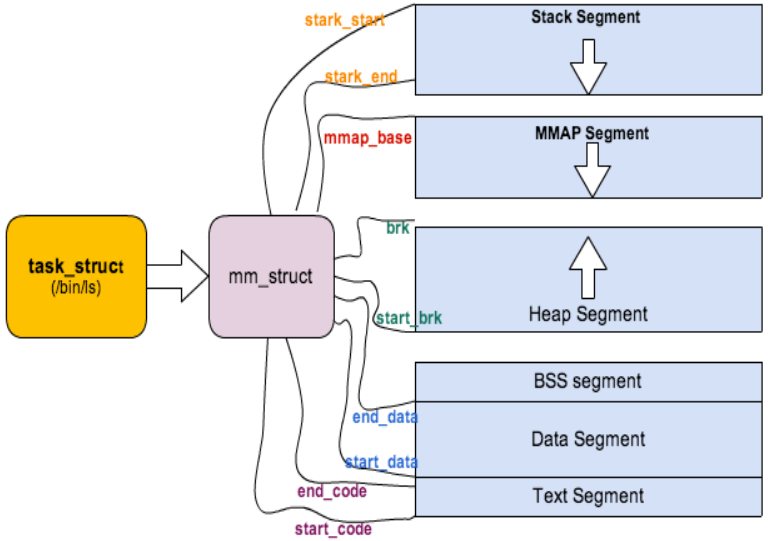
进程 -- 进程虚拟内存管理

进程地址空间从低地址开始依次是代码段(Text)、数据段(Data)、BSS段、堆、内存映射段(mmap)、栈。

2.1.1 32 位模式下进程内存经典布局



这种布局是 Linux 内核 2.6.7 以前的默认进程内存布局形式，mmap 区域与栈区域相对增长，这意味着堆只有 1GB 的虚拟地址空间可以使用，继续增长就会进入 mmap 映射区域，这显然不是我们想要的。这是由于 32 模式地址空间限制造成的，所以内核引入了另一种虚拟地址空间的布局形式，将在后面介绍。但对于 64 位系统，提供了巨大的虚拟地址空间，这种布局就相当好。



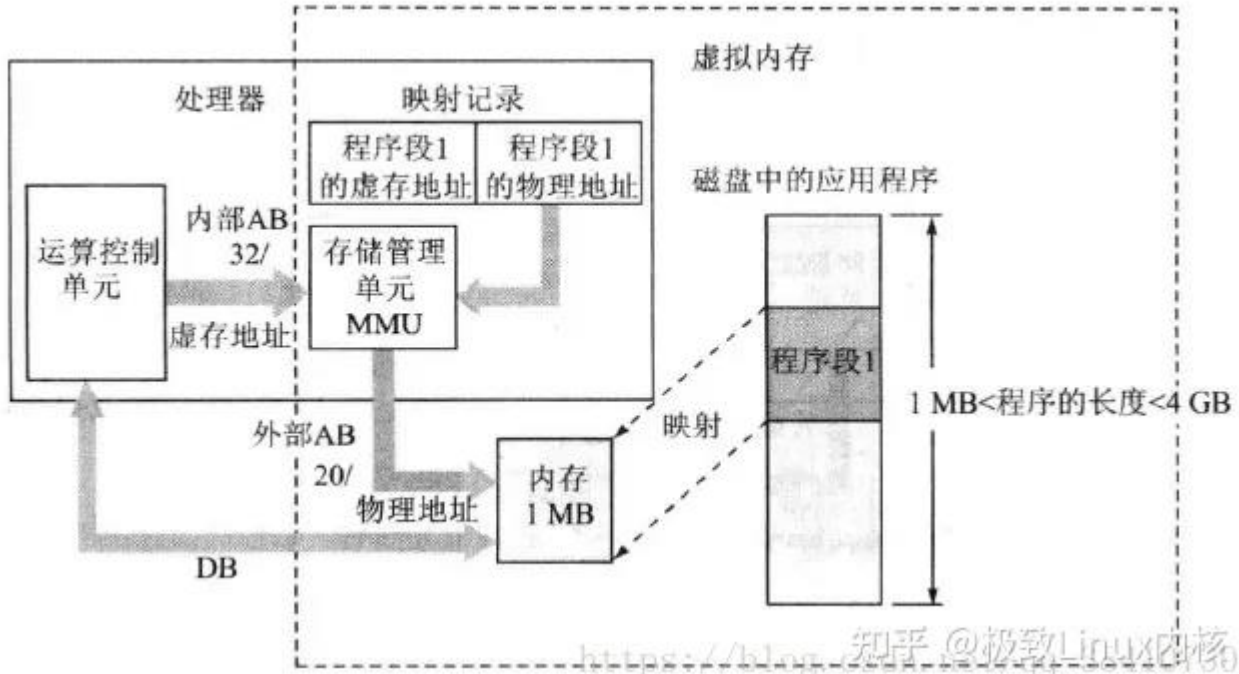
进程 -- 进程虚拟内存管理

Ghostcloud
New Computing & New Era

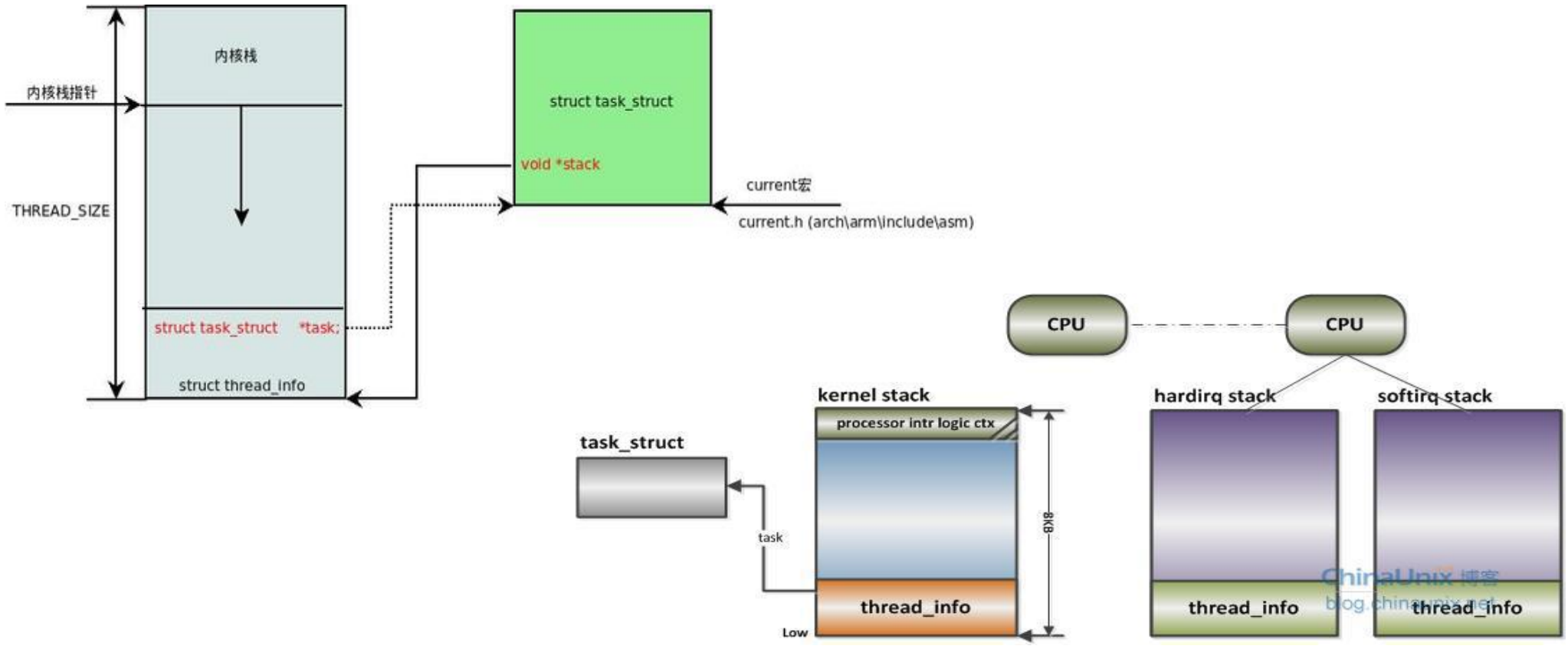
```
root@ubuntu:~# cat  /proc/7257/maps
55bafd0e2000-55bafd1e6000 r-xp 00000000 08:01 789356                     /bin/bash
55bafd3e5000-55bafd3e9000 r--p 00103000 08:01 789356                     /bin/bash
55bafd3e9000-55bafd3f2000 rw-p 00107000 08:01 789356                     /bin/bash
55bafd3f2000-55bafd3fc000 rw-p 00000000 00:00 0
55bafe776000-55bafef90000 rw-p 00000000 00:00 0                          [heap]
7f211cb9f000-7f211cbaa000 r-xp 00000000 08:01 2102768                    /lib/x86_64-linux-gnu/libnss_files-2.27.so
7f211cbaa000-7f211cda9000 ---p 0000b000 08:01 2102768                    /lib/x86_64-linux-gnu/libnss_files-2.27.so
7f211cda9000-7f211cdaa000 r--p 0000a000 08:01 2102768                    /lib/x86_64-linux-gnu/libnss_files-2.27.so
7f211cdaa000-7f211cdab000 rw-p 0000b000 08:01 2102768                    /lib/x86_64-linux-gnu/libnss_files-2.27.so
7f211cdab000-7f211cdb1000 rw-p 00000000 00:00 0
7f211cdb1000-7f211cdc8000 r-xp 00000000 08:01 2102765                    /lib/x86_64-linux-gnu/libnsl-2.27.so
7f211cdc8000-7f211cfc7000 ---p 00017000 08:01 2102765                    /lib/x86_64-linux-gnu/libnsl-2.27.so
7f211cfc7000-7f211cfc8000 r--p 00016000 08:01 2102765                    /lib/x86_64-linux-gnu/libnsl-2.27.so
7f211cfc8000-7f211cfc9000 rw-p 00017000 08:01 2102765                    /lib/x86_64-linux-gnu/libnsl-2.27.so
7f211cfc9000-7f211cfcb000 rw-p 00000000 00:00 0
7f211cfcb000-7f211cfd6000 r-xp 00000000 08:01 2102980                    /lib/x86_64-linux-gnu/libnss_nis-2.27.so
7f211cfd6000-7f211d1d5000 ---p 0000b000 08:01 2102980                    /lib/x86_64-linux-gnu/libnss_nis-2.27.so
7f211d1d5000-7f211d1d6000 r--p 0000a000 08:01 2102980                    /lib/x86_64-linux-gnu/libnss_nis-2.27.so
7f211d1d6000-7f211d1d7000 rw-p 0000b000 08:01 2102980                    /lib/x86_64-linux-gnu/libnss_nis-2.27.so
7f211d1d7000-7f211d1df000 r-xp 00000000 08:01 2102766                    /lib/x86_64-linux-gnu/libnss_compat-2.27.so
7f211d1df000-7f211d3df000 ---p 00008000 08:01 2102766                    /lib/x86_64-linux-gnu/libnss_compat-2.27.so
7f211d3df000-7f211d3e0000 r--p 00008000 08:01 2102766                    /lib/x86_64-linux-gnu/libnss_compat-2.27.so
7f211d3e0000-7f211d3e1000 rw-p 00009000 08:01 2102766                    /lib/x86_64-linux-gnu/libnss_compat-2.27.so
7f211d3e1000-7f211d6bf000 r--p 00000000 08:01 1575513                    /usr/lib/locale/locale-archive
7f211d6bf000-7f211d8a6000 r-xp 00000000 08:01 2102674                    /lib/x86_64-linux-gnu/libc-2.27.so
7f211d8a6000-7f211daa6000 ---p 001e7000 08:01 2102674                    /lib/x86_64-linux-gnu/libc-2.27.so
7f211daa6000-7f211daaa000 r--p 001e7000 08:01 2102674                    /lib/x86_64-linux-gnu/libc-2.27.so
7f211daaa000-7f211daac000 rw-p 001eb000 08:01 2102674                    /lib/x86_64-linux-gnu/libc-2.27.so
7f211daac000-7f211dab0000 rw-p 00000000 00:00 0
7f211dab0000-7f211dab3000 r-xp 00000000 08:01 2102719                    /lib/x86_64-linux-gnu/libdl-2.27.so
7f211dab3000-7f211dcb2000 ---p 00003000 08:01 2102719                    /lib/x86_64-linux-gnu/libdl-2.27.so
7f211dcb2000-7f211dcb3000 r--p 00002000 08:01 2102719                    /lib/x86_64-linux-gnu/libdl-2.27.so
7f211dcb3000-7f211dcb4000 rw-p 00003000 08:01 2102719                    /lib/x86_64-linux-gnu/libdl-2.27.so
7f211dcb4000-7f211dcd9000 r-xp 00000000 08:01 2103199                    /lib/x86_64-linux-gnu/libtinfo.so.5.9
7f211dcd9000-7f211ded9000 ---p 00025000 08:01 2103199                    /lib/x86_64-linux-gnu/libtinfo.so.5.9
7f211ded9000-7f211dedd000 r--p 00025000 08:01 2103199                    /lib/x86_64-linux-gnu/libtinfo.so.5.9
7f211dedd000-7f211dede000 rw-p 00029000 08:01 2103199                    /lib/x86_64-linux-gnu/libtinfo.so.5.9
7f211dede000-7f211df07000 r-xp 00000000 08:01 2097245                    /lib/x86_64-linux-gnu/ld-2.27.so
7f211e06b000-7f211e0f1000 rw-p 00000000 00:00 0
7f211e100000-7f211e107000 r--s 00000000 08:01 1716171                    /usr/lib/x86_64-linux-gnu/gconv/gconv-modules.cache
7f211e107000-7f211e108000 r--p 00029000 08:01 2097245                    /lib/x86_64-linux-gnu/ld-2.27.so
7f211e108000-7f211e109000 rw-p 0002a000 08:01 2097245                    /lib/x86_64-linux-gnu/ld-2.27.so
7f211e109000-7f211e10a000 rw-p 00000000 00:00 0
7ffc58d78000-7ffc58d99000 rw-p 00000000 00:00 0                          [stack]
7ffc58dba000-7ffc58dbd000 r--p 00000000 00:00 0                          [vvar]
7ffc58dbd000-7ffc58dbf000 r-xp 00000000 00:00 0                          [vdso]
ffffffffff600000-ffffffffff601000 --xp 00000000 00:00 0                  [vsyscall]
```

	start	end	size(KB)	filename	permission
6	00cdd000	2aaa8000	685868	NOT USED	unknown
383	7f900000	7fe58000	5472	NOT USED	unknown
323	7a200000	7a3b9000	1764	NOT USED	unknown
310	78a00000	78bb6000	1752	NOT USED	unknown
385	7fe79000	80000000	1564	NOT USED	unknown
285	75100000	7523f000	1276	NOT USED	unknown
376	7ed00000	7ee00000	1024	NOT USED	unknown
333	7b901000	7ba00000	1020	NOT USED	unknown
381	7f701000	7f800000	1020	NOT USED	unknown
365	7e702000	7e800000	1016	NOT USED	unknown
348	7d52e000	7d600000	840	NOT USED	unknown
328	7af38000	7b000000	800	NOT USED	unknown
278	74a50000	74b00000	704	NOT USED	unknown
300	783b5000	78400000	300	NOT USED	unknown
315	79bb8000	79c00000	288	NOT USED	unknown
341	7bf00000	7bf39000	228	NOT USED	unknown
0	0	8000	32	NOT USED	unknown
2	0001d000	1.00E+00	4	NOT USED	unknown
362	7df00000	7df01000	4	NOT USED	unknown

进程 -- 进程虚拟内存管理


虚拟内存
处理器
映射记录
程序段1
的虚存地址
程序段1
的物理地址
磁盘中的应用程序
运算控制
单元
内部AB
32/
存储管理
单元
MMU
虚存地址
程序段1
1 MB<程序的长度<4 GB
映射
外部AB
20/
内存
1 MB
物理地址
DB

进程 -- 线程切换


内核栈
内核栈指针
THREAD_SIZE
struct task_struct *task;
struct thread_info
struct task_struct
void *stack
current宏
current.h (arch\arm\include\asm)
CPU
CPU
kernel stack
processor intr logic ctx
task_struct
task
thread_info
Low
8KB
hardirq stack
softirq stack
thread_info
thread_info

进程控制组 -- CGroup

CGroups，其名称源自控制组群（control groups）的缩写，是内核的一个特性，用于限制、记录和隔离一组进程的资源使用（CPU、内存、磁盘 I/O、网络等）。

资源限制：可以配置 cgroup，从而限制进程可以对特定资源（例如内存或 CPU）的使用量

优先级：当资源发生冲突时，您可以控制一个进程相比另一个 cgroup 中的进程可以使用的资源量（CPU、磁盘或网络）

记录：在 cgroup 级别监控和报告资源限制

控制：您可以使用单个命令更改 cgroup 中所有进程的状态（冻结、停止或重新启动）

进程控制组 -- CGroup

Ghostcloud
New Computing & New Era

Cgroups功能的实现依赖于四个核心概念：子系统、控制组、层级树、任务

控制组（cgroup）：表示一组进程和一组带有参数的子系统的关联关系。

层级树（hierarchy）：由一系列的控制组按照树状结构排列组成的。

子系统（subsystem）：一个子系统代表一类资源调度控制器(又叫 controllers)。

任务（task）：在cgroup中，任务就是一个进程，一个任务可以是多个cgroup的成员，子进程自动成为父进程cgroup的成员，可按需求将子进程移到不同的cgroup中

进程控制组 -- 子系统

cpu：使用调度程序控制任务对cpu的使用

cpuacct：自动生成cgroup中任务对cpu资源使用情况的报告

cpuset：可以为cgroup中的任务分配独立的cpu和内存

blkio：可以为块设备设定输入 输出限制，比如物理驱动设备

devices：可以开启或关闭cgroup中任务对设备的访问

freezer： 可以挂起或恢复cgroup中的任务

pids：限制任务数量

memory：可以设定cgroup中任务对内存使用量的限定，并且自动生成这些任务对内存资源使用情况的报告

perf_event：使用后使cgroup中的任务可以进行统一的性能测试

net_cls：docker没有直接使用它，它通过使用等级识别符标记网络数据包，从而允许linux流量控制程序识别从具体cgroup中生成的数据包

进程控制组 -- 目录结构

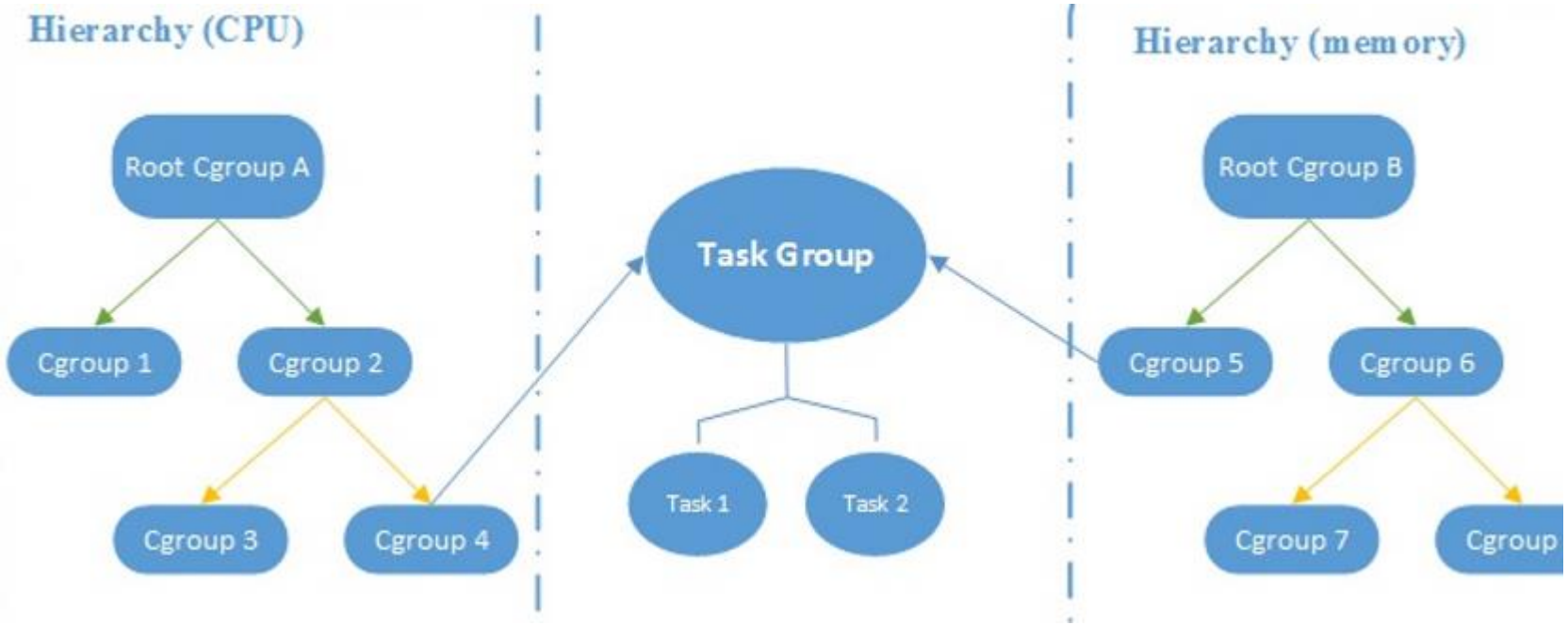
Ghostcloud
New Computing & New Era

```
root@ubuntu:/sys/fs/cgroup# ll
total 0
drwxr-xr-x 15 root root 380 Apr 12 18:50 ./
drwxr-xr-x  9 root root   0 Apr 12 18:50 ../
dr-xr-xr-x  5 root root   0 Apr 12 18:50 blkio/
lrwxrwxrwx  1 root root  11 Apr 12 18:50 cpu -> cpu,cpuacct/
lrwxrwxrwx  1 root root  11 Apr 12 18:50 cpuacct -> cpu,cpuacct/
dr-xr-xr-x  5 root root   0 Apr 12 18:50 cpu,cpuacct/
dr-xr-xr-x  3 root root   0 Apr 12 18:50 cpuset/
dr-xr-xr-x  5 root root   0 Apr 12 18:50 devices/
dr-xr-xr-x  3 root root   0 Apr 12 18:50 freezer/
dr-xr-xr-x  3 root root   0 Apr 12 18:50 hugetlb/
dr-xr-xr-x  5 root root   0 Apr 12 18:50 memory/
lrwxrwxrwx  1 root root  16 Apr 12 18:50 net_cls -> net_cls,net_prio/
dr-xr-xr-x  3 root root   0 Apr 12 18:50 net_cls,net_prio/
lrwxrwxrwx  1 root root  16 Apr 12 18:50 net_prio -> net_cls,net_prio/
dr-xr-xr-x  3 root root   0 Apr 12 18:50 perf_event/
dr-xr-xr-x  5 root root   0 Apr 12 18:50 pids/
dr-xr-xr-x  3 root root   0 Apr 12 18:50 rdma/
dr-xr-xr-x  6 root root   0 Apr 12 18:50 systemd/
dr-xr-xr-x  6 root root   0 Apr 12 18:50 unified/
```

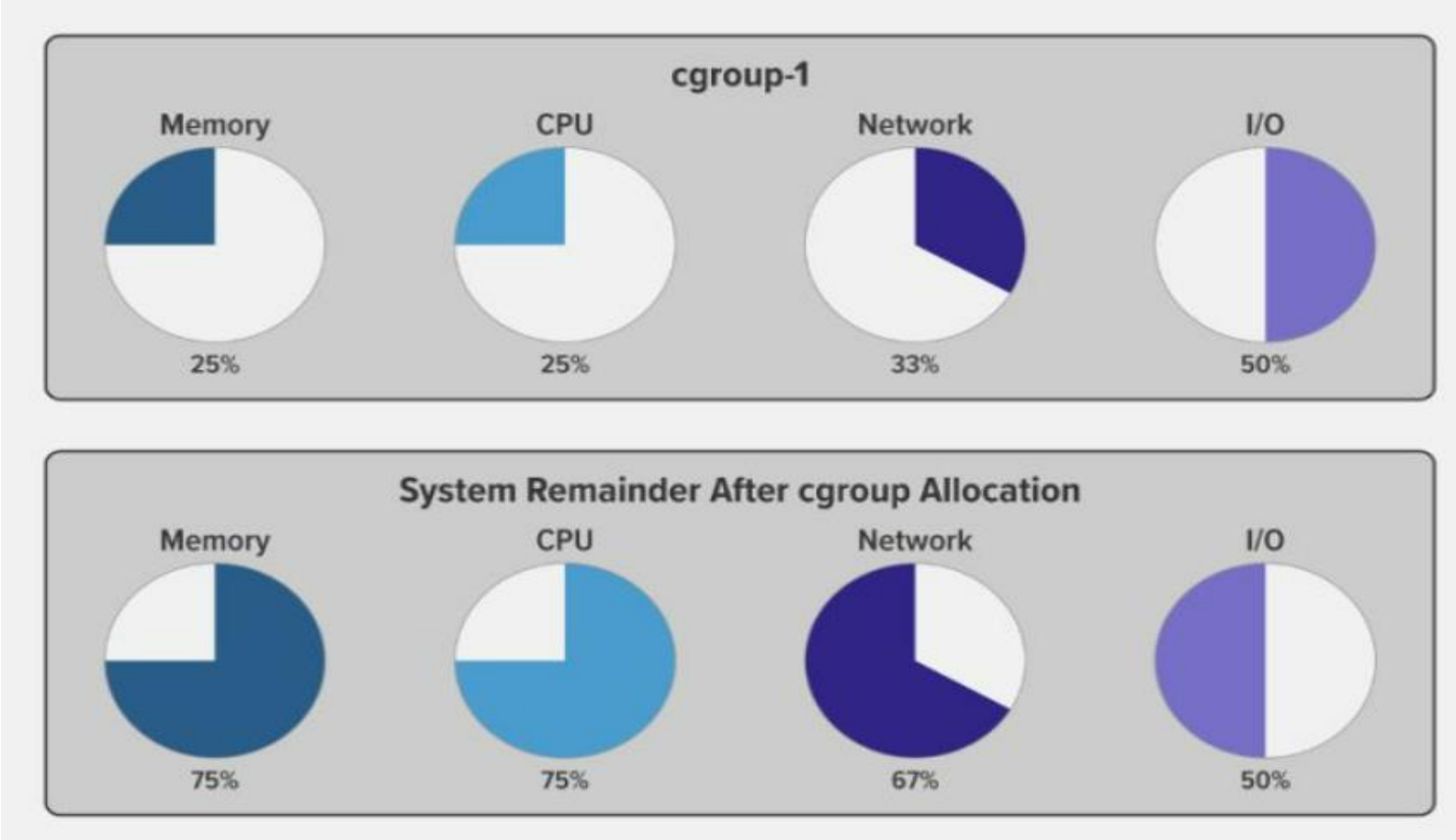
```
root@ubuntu:/sys/fs/cgroup/cpu# ll
total 0
dr-xr-xr-x  5 root root   0 Apr 12 18:50 ./
drwxr-xr-x 15 root root 380 Apr 12 18:50 ../
-rw-r--r--  1 root root   0 Apr 12 18:51 cgroup.clone_children
-rw-r--r--  1 root root   0 Apr 12 18:50 cgroup.procs
-r--r--r--  1 root root   0 Apr 12 18:51 cgroup.sane_behavior
-r--r--r--  1 root root   0 Apr 12 18:51 cpuacct.stat
-rw-r--r--  1 root root   0 Apr 12 18:51 cpuacct.usage
-r--r--r--  1 root root   0 Apr 12 18:51 cpuacct.usage_all
-r--r--r--  1 root root   0 Apr 12 18:51 cpuacct.usage_percpu
-r--r--r--  1 root root   0 Apr 12 18:51 cpuacct.usage_percpu_sys
-r--r--r--  1 root root   0 Apr 12 18:51 cpuacct.usage_percpu_user
-r--r--r--  1 root root   0 Apr 12 18:51 cpuacct.usage_sys
-r--r--r--  1 root root   0 Apr 12 18:51 cpuacct.usage_user
-rw-r--r--  1 root root   0 Apr 12 18:51 cpu.cfs_period_us
-rw-r--r--  1 root root   0 Apr 12 18:51 cpu.cfs_quota_us
-rw-r--r--  1 root root   0 Apr 12 18:51 cpu.shares
-r--r--r--  1 root root   0 Apr 12 18:51 cpu.stat
drwxr-xr-x  4 root root   0 Apr 12 18:51 kubepods.slice/
-rw-r--r--  1 root root   0 Apr 12 18:51 notify_on_release
-rw-r--r--  1 root root   0 Apr 12 18:51 release_agent
drwxr-xr-x 73 root root   0 Apr 12 18:50 system.slice/
-rw-r--r--  1 root root   0 Apr 12 18:51 tasks
drwxr-xr-x  2 root root   0 Apr 12 18:50 user.slice/
root@ubuntu:/sys/fs/cgroup/cpu#
```

```
root@ubuntu:/sys/fs/cgroup# ll memory/
total 0
dr-xr-xr-x  5 root root   0 Apr 12 18:50 ./
drwxr-xr-x 15 root root 380 Apr 12 18:50 ../
-rw-r--r--  1 root root   0 Apr 12 18:51 cgroup.clone_children
--w--w--w-  1 root root   0 Apr 12 18:51 cgroup.event_control
-rw-r--r--  1 root root   0 Apr 12 18:50 cgroup.procs
-r--r--r--  1 root root   0 Apr 12 18:51 cgroup.sane_behavior
drwxr-xr-x  4 root root   0 Apr 12 18:51 kubepods.slice/
-rw-r--r--  1 root root   0 Apr 12 18:51 memory.failcnt
--w-------  1 root root   0 Apr 12 18:51 memory.force_empty
-rw-r--r--  1 root root   0 Apr 12 18:51 memory.kmem.failcnt
-rw-r--r--  1 root root   0 Apr 12 18:51 memory.kmem.limit_in_bytes
-rw-r--r--  1 root root   0 Apr 12 18:51 memory.kmem.max_usage_in_bytes
-r--r--r--  1 root root   0 Apr 12 18:51 memory.kmem.slabinfo
-rw-r--r--  1 root root   0 Apr 12 18:51 memory.kmem.tcp.failcnt
-rw-r--r--  1 root root   0 Apr 12 18:51 memory.kmem.tcp.limit_in_bytes
-rw-r--r--  1 root root   0 Apr 12 18:51 memory.kmem.tcp.max_usage_in_bytes
-r--r--r--  1 root root   0 Apr 12 18:51 memory.kmem.tcp.usage_in_bytes
-r--r--r--  1 root root   0 Apr 12 18:51 memory.kmem.usage_in_bytes
-rw-r--r--  1 root root   0 Apr 12 18:50 memory.limit_in_bytes
-rw-r--r--  1 root root   0 Apr 12 18:51 memory.max_usage_in_bytes
-rw-r--r--  1 root root   0 Apr 12 18:51 memory.move_charge_at_immigrate
-r--r--r--  1 root root   0 Apr 12 18:51 memory.numa_stat
-rw-r--r--  1 root root   0 Apr 12 18:51 memory.oom_control
----------  1 root root   0 Apr 12 18:51 memory.pressure_level
-rw-r--r--  1 root root   0 Apr 12 18:51 memory.soft_limit_in_bytes
-r--r--r--  1 root root   0 Apr 12 18:51 memory.stat
-rw-r--r--  1 root root   0 Apr 12 18:51 memory.swappiness
-r--r--r--  1 root root   0 Apr 12 18:51 memory.usage_in_bytes
-rw-r--r--  1 root root   0 Apr 12 18:50 memory.use_hierarchy
-rw-r--r--  1 root root   0 Apr 12 18:51 notify_on_release
-rw-r--r--  1 root root   0 Apr 12 18:51 release_agent
drwxr-xr-x 73 root root   0 Apr 12 18:50 system.slice/
-rw-r--r--  1 root root   0 Apr 12 18:51 tasks
drwxr-xr-x  2 root root   0 Apr 12 18:50 user.slice/
root@ubuntu:/sys/fs/cgroup# ll cpu
```

进程控制组 -- 层级结构


Hierarchy (CPU)
Root Cgroup A
Cgroup 1
Cgroup 2
Cgroup 3
Cgroup 4
Task Group
Task 1
Task 2
Hierarchy (memory)
Root Cgroup B
Cgroup 5
Cgroup 6
Cgroup 7
Cgroup

进程控制组 -- 作用


cgroup-1
Memory
CPU
Network
I/O
25%
25%
33%
50%
System Remainder After cgroup Allocation
Memory
CPU
Network
I/O
75%
75%
67%
50%

namespace -- 种类

unshare命令，取消共享父进程中指定的命名空间，然后执行指定的程序。要取消共享的命名空间通过选项来指示。不可共享的命名空间有

(1) Mount Namespace：用来隔离不同的进程或进程组看到的挂载点。通俗地说，就是可以实现在不同的进程中看到不同的挂载目录。

(2)PID Namespace

PID Namespace 的作用是用来隔离进程。在不同的 PID Namespace 中，进程可以拥有相同的 PID 号，利用 PID Namespace 可以实现每个容器的主进程为 1 号进程，而容器内的进程在主机上却拥有不同的PID。

(3)UTS Namespace

UTS Namespace 主要是用来隔离主机名的，它允许每个 UTS Namespace 拥有一个独立的主机名。

(4)IPC Namespace

IPC Namespace 主要是用来隔离进程间通信的。

(5)User Namespace

User Namespace 主要是用来隔离用户和用户组的。

(6)Net Namespace

Net Namespace 是用来隔离网络设备、IP 地址和端口等信息的。

namespace -- 种类

```
root@ubuntu:/proc/8/ns# ll
total 0
dr-x--x--x 2 root root 0 Apr 12 23:21 ./
dr-xr-xr-x 9 root root 0 Apr 12 23:20 ../
lrwxrwxrwx 1 root root 0 Apr 12 23:21 cgroup -> 'cgroup:[4026531835]'
lrwxrwxrwx 1 root root 0 Apr 12 23:21 ipc -> 'ipc:[4026531839]'
lrwxrwxrwx 1 root root 0 Apr 12 23:21 mnt -> 'mnt:[4026531840]'
lrwxrwxrwx 1 root root 0 Apr 12 23:21 net -> 'net:[4026531992]'
lrwxrwxrwx 1 root root 0 Apr 12 23:21 pid -> 'pid:[4026531836]'
lrwxrwxrwx 1 root root 0 Apr 12 23:21 pid_for_children -> 'pid:[4026531836]'
lrwxrwxrwx 1 root root 0 Apr 12 23:21 user -> 'user:[4026531837]'
lrwxrwxrwx 1 root root 0 Apr 12 23:21 uts -> 'uts:[4026531838]'
```

```
root@ubuntu:/proc/73361/ns# ll
total 0
dr-x--x--x 2 root root 0 Apr 12 23:22 ./
dr-xr-xr-x 9 root root 0 Apr 12 23:22 ../
lrwxrwxrwx 1 root root 0 Apr 12 23:23 cgroup -> 'cgroup:[4026531835]'
lrwxrwxrwx 1 root root 0 Apr 12 23:23 ipc -> 'ipc:[4026531839]'
lrwxrwxrwx 1 root root 0 Apr 12 23:23 mnt -> 'mnt:[4026531840]'
lrwxrwxrwx 1 root root 0 Apr 12 23:23 net -> 'net:[4026531992]'
lrwxrwxrwx 1 root root 0 Apr 12 23:23 pid -> 'pid:[4026531836]'
lrwxrwxrwx 1 root root 0 Apr 12 23:23 pid_for_children -> 'pid:[4026531836]'
lrwxrwxrwx 1 root root 0 Apr 12 23:23 user -> 'user:[4026531837]'
lrwxrwxrwx 1 root root 0 Apr 12 23:23 uts -> 'uts:[4026531838]'
root@ubuntu:/proc/73361/ns#
```

namespace -- unshare

unshare命令，取消共享父进程中指定的命名空间，然后执行指定的程序。要取消共享的命名空间通过选项来指示。不可共享的命名空间有

```
root@ubuntu:/sys/fs/cgroup/cpu# unshare --help

Usage:
 unshare [options] [<program> [<argument>...]]

Run a program with some namespaces unshared from the parent.

Options:
 -m, --mount[=<file>]      unshare mounts namespace
 -u, --uts[=<file>]        unshare UTS namespace (hostname etc)
 -i, --ipc[=<file>]        unshare System V IPC namespace
 -n, --net[=<file>]        unshare network namespace
 -p, --pid[=<file>]        unshare pid namespace
 -U, --user[=<file>]       unshare user namespace
 -C, --cgroup[=<file>]     unshare cgroup namespace
 -f, --fork                fork before launching <program>
     --mount-proc[=<dir>]  mount proc filesystem first (implies --mount)
 -r, --map-root-user       map current user to root (implies --user)
     --propagation slave|shared|private|unchanged
                           modify mount propagation in mount namespace
 -s, --setgroups allow|deny  control the setgroups syscall in user namespaces

 -h, --help                display this help
 -V, --version             display version

For more details see unshare(1).
root@ubuntu:/sys/fs/cgroup/cpu#
```

namespace -- unshare

Ghostcloud
New Computing & New Era

```
int clone(int (*fn)(void *), void *child_stack, int flags, void *arg);
```

这里fn是函数指针，我们知道进程的4要素，这个就是指向程序的指针，就是所谓的“剧本"；child_stack是为子进程分配堆栈空间, 无论子进程是否与父进程共享内存空间，该指针都不能为NULL，否则，无法创建子进程。由于linux中栈是栈底在高地址，所以，应将栈底的地址传给clone函数。flags就是标志用来描述你需要从父进程继承那些资源， arg就是传给子进程的参数）。下面是flags可以取的值

标志	含义
CLONE_PARENT	创建的子进程的父进程是调用者的父进程，新进程与创建它的进程成了“兄弟”而不是“父子”
CLONE_FS	子进程与父进程共享相同的文件系统，包括root、当前目录、umask
CLONE_FILES	子进程与父进程共享相同的文件描述符（file descriptor）表
CLONE_NEWNS	在新的namespace启动子进程，namespace描述了进程的文件hierarchy
CLONE_SIGHAND	子进程与父进程共享相同的信号处理（signal handler）表
CLONE_PTRACE	若父进程被trace，子进程也被trace
CLONE_VFORK	父进程被挂起，直至子进程释放虚拟内存资源
CLONE_VM	子进程与父进程运行于相同的内存空间
CLONE_PID	子进程在创建时PID与父进程一致
CLONE_THREAD	Linux 2.4中增加以支持POSIX线程标准，子进程与父进程共享相同的线程群

下面的例子是创建一个线程（子进程共享了父进程虚存空间，没有自己独立的虚存空间不能称其为进程）。父进程被挂起当子线程释放虚存资源后再继续执行。

chroot

Ghostcloud
New Computing & New Era

```
root@ubuntu:/proc/73361/ns# chroot --help
Usage: chroot [OPTION] NEWROOT [COMMAND [ARG]...]
  or:  chroot OPTION
Run COMMAND with root directory set to NEWROOT.

  --groups=G_LIST        specify supplementary groups as g1,g2,..,gN
  --userspec=USER:GROUP  specify user and group (ID or name) to use
  --skip-chdir           do not change working directory to '/'
      --help     display this help and exit
      --version  output version information and exit

If no command is given, run '"$SHELL" -i' (default: '/bin/sh -i').

GNU coreutils online help: <http://www.gnu.org/software/coreutils/>
Full documentation at: <http://www.gnu.org/software/coreutils/chroot>
or available locally via: info '(coreutils) chroot invocation'
root@ubuntu:/proc/73361/ns#
```

Linux容器 -- LXC

由于inux Kernel支持Cgroup(控制组)和NameSpace(命名空间)技术，这意味着一切都可在Linux内核中实现虚拟化，因此推动了容器的进一步发展。

2008年前后，作为一个开源容器平台，Linux容器项目LXC是众所周知的工具，模板，库和语言绑定集。它的级别很低，非常灵活，几乎涵盖了上游内核支持的每个遏制功能。同年，RedHat公司发布了用于管理虚拟化平台的Libvirt工具。

LXC是Linux内核包含功能的用户空间接口。通过功能强大的API和简单的工具，它使Linux用户可以轻松地创建和管理系统或应用程序容器。LXC采用简单的命令行界面，可改善容器启动时的用户体验。