Topics in Operating Systems

Advanced isolation: Application containers

Debadatta Mishra, CSE, IITK

Linux kernel enablers for containers

1

Namespaces

Kernel abstraction for restricted system view

2

Control groups (cgroups)

Techniques for resource isolation (quota, usage etc.)

API

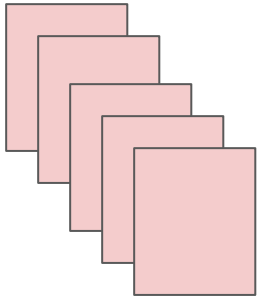
System calls: clone( ), unshare( ) and setns( )

API

Kernel sysfs API (/sys/fs/cgroup)

Container management frameworks (e.g., LXC, docker)

Containers



Linux namespace API

- There are several namespaces (8 currently), each enabling a restricted view for one subsystem. {ls -ltr /proc/self/ns}
- pid (processes)
- uts (hostname)
- mnt (mount point, file system)
- Ipc (system-v IPC)
- net (network stack)
- ...

Linux namespace API - system calls

- **clone( ):** System call to create process/thread. Flags like CLONE_NEWUTS, CLONE_NEWIPC, CLONE_NEWPID, CLONE_NEWNET etc. are used to create a process in separate namespaces

- Example: clone( ) system call with CLONE_NEWPID creates a new PID namespace and creates the first process (with pid = 1) in the new namespace.

  Note: The child getpid( ) returns 1 but the parent process can see the global PID of the child process

Linux namespace API - system calls

- **unshare(flags):** Based on the value of flags (CLONE_NEWUTS, CLONE_NEWNET etc.)
  - Disassociate the calling process from shared namespaces
  - Create a new namespace
  - Attach the calling process to the new namespace

- Example: *unshare(NEW_UTS)* followed by a *sethostname( )* system call with a new hostname creates a new UTS namespace for the calling process

  Notes: 1) CLONE_NEWPID is not allowed, 2) "unshare" command line utility

Linux namespace API - system calls

- **setns(fd, nstype):** Associate the calling process with an existing namespace
  - “fd” represents the existing namespace
  - “nstype” is used to specify the namespace

- Example:
- 1. Create a process (P) with NEWUTS (using clone) and change the hostname in child
- 2. In another process(Q), call setns with fd = open(“/proc/P/ns/uts)” and nstype = 0
- 3. Call execl(bash) and check the hostname

Linux namespace API - NET and MNT

- NET namespace is a logically isolated network stack
- Has its own device, stack, routes firewall rules etc.
- Requires support of network utilities like veth, software bridge

- MNT provides a separate view of mounts
- Example demo

Linux kernel enablers for containers

1

Namespaces

Kernel abstraction for restricted system view

2

Control groups (cgroups)

Techniques for resource isolation (quota, usage etc.)

API

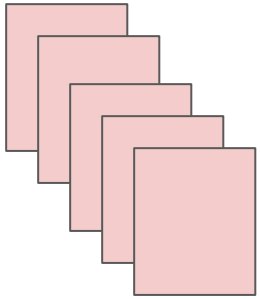
System calls: clone( ), unshare( ) and setns( )

API

Kernel sysfs API (/sys/fs/cgroup)

Container management frameworks (e.g., LXC, docker)

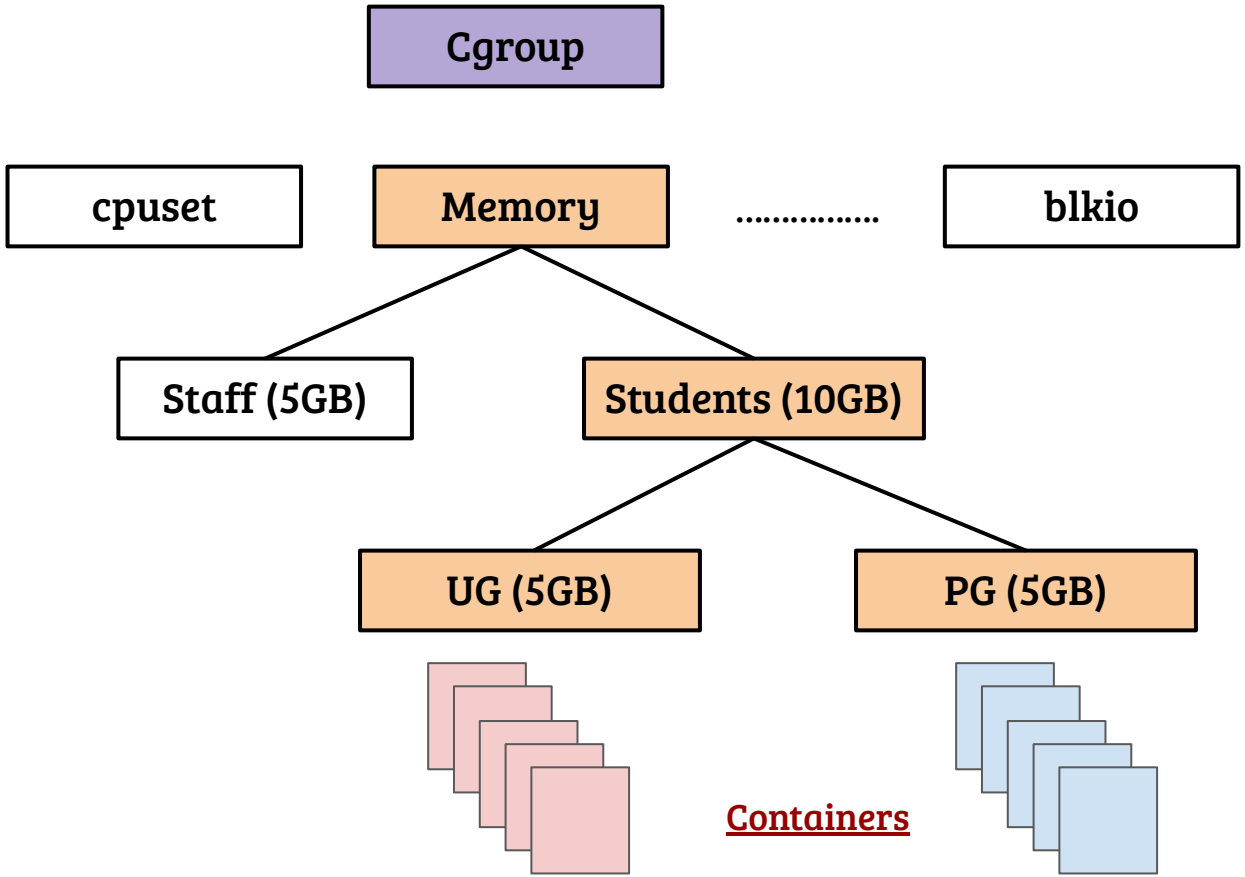
Containers



Linux Cgroup API

- Provides resource usage and monitoring in several resource dimensions
- Inherent support for hierarchical resource management
- Currently linux systems support 12 cgroups, each for one resource type
- Most prominently used cgroups: memory, cpuset, blkio
- Container frameworks use both cgroups and namespaces
- Cgroups has broader applicability: can be used independent of namespaces

Linux Cgroup hierarchy example: Memory



- Kernel supports dynamic extension of the hierarchy
- Process can be added/removed dynamically

Cgroup setup and usage

- Step 1: mount cgroup file system (if not already done)
- Step 2: mount subsystems (cpuset, memory)
- Step 3: Create the cgroup hierarchy
- Step 4: Apply resource limits as per the policy
- Step 5: Add processes to the cgroup to enforce cgroup level resource limits
- DEMO