Understanding the Design and Applications of Container based System Virtualization

M.Tech Seminar Presentation

Prashanth, 153050095
Guided by: Prof. Purushottam Kulkarni

Introduction

- IAAS – Provides resources as service
- Virtual machines (VM) helps resource
  - Partitioning
  - Scaling

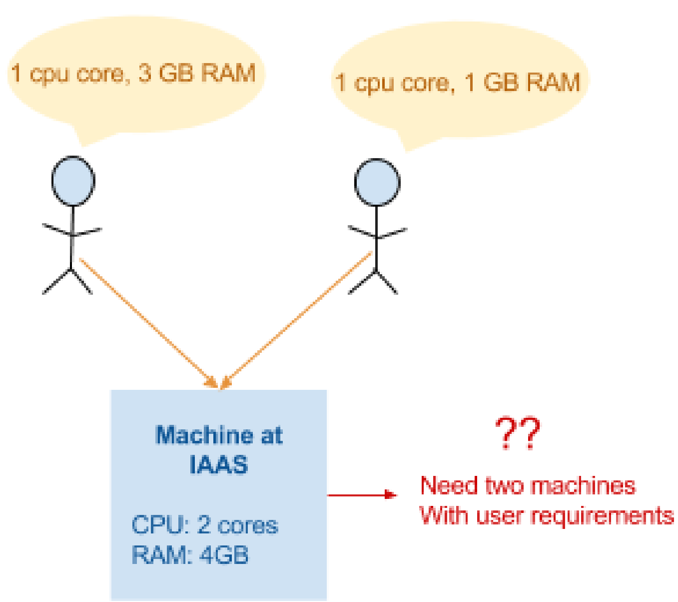


Fig: Without virtualization

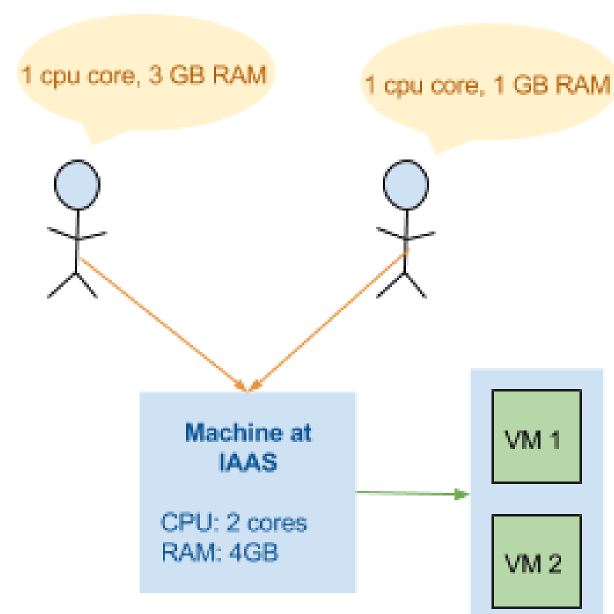


Fig: With virtualization

Drawbacks IAAS architecture using VM

- Complete hardware stack emulation
- Full OS required – Additional memory
- Start up latency
- Dual control loop
- Overheads – bad cost-benefit ratio – Overprice customers

Requirements of a new system

Process groups specify following to OS and OS should enforce these,

- Resource constrains
- Isolation
- Account resource usage
- Minimum overhead

Overview of talk

- Containers
- Building blocks
- Types with examples – Docker, LXC
- Applications
- Comparison with VM

Containers

" Container is a virtual environment that contains a set of processes grouped along with its dependent resources into a single logical OS entity. It enables multiple isolated user-spaces "

- Share the OS kernel – OS-Virtualization
- Building blocks of a container
    1. Control groups – resource control
    2. Namespaces – isolation
    3. Disk images – mount a new ROOTFS

Control Groups (cgroups)

- **Resource controller** for each resource
- 12 different subsystems – CPU, memory etc.
- Perform Accounting
- Follows hierarchy
- User space API – pseudo file-system

Reference: [1]

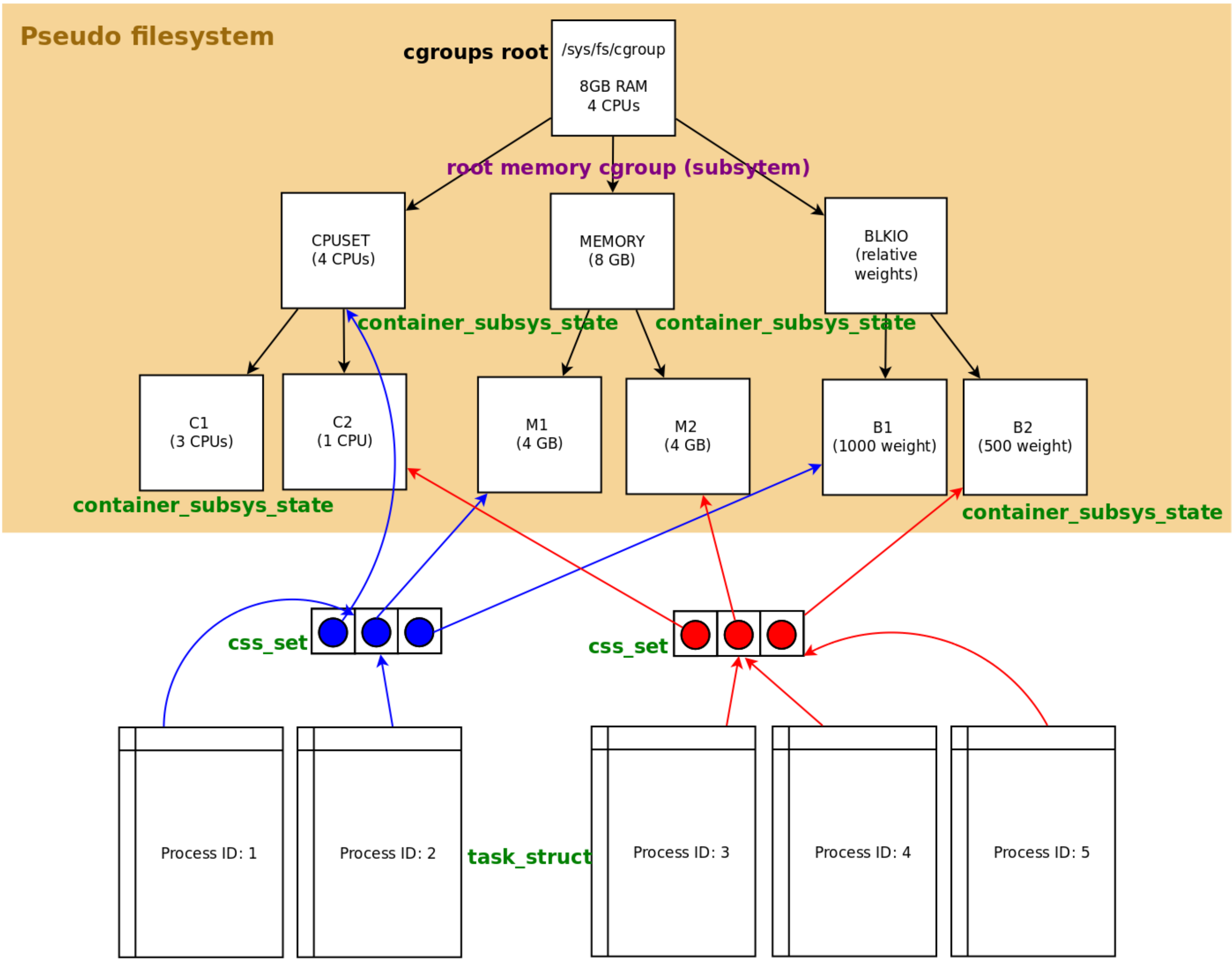


Fig: Control groups illustration using 3 controllers

LABELS

Violet:
Resource controller

Green:
Kernel Data structures

Blue:
Pointers for container 1

Blue:
Pointers for container 2

Black Boxes:
Directories used to manage cgroup nodes

Namespaces

- **Isolated system view**
- May/may not follow hierarchy
- 10 proposed, 6 done
- Namespaces – mount, PID, network, IPC, UTS, user
- clone(), struct ns_proxy

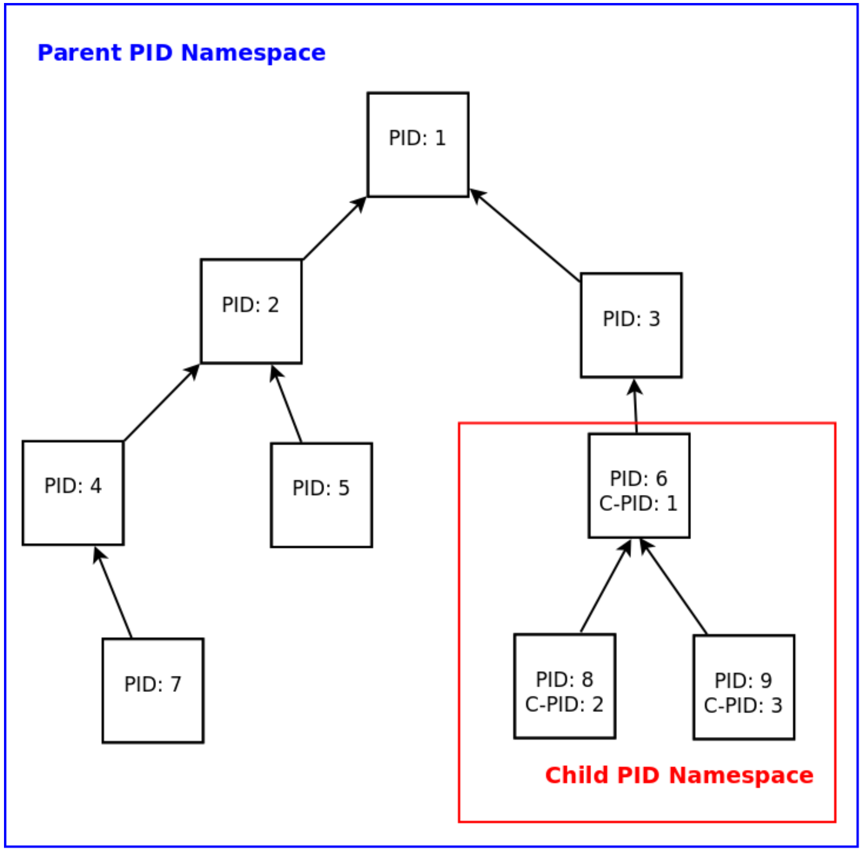


Fig: Example of PID Namespace in which pids 6,8,9 in parent map to 1,2,3 in child

Reference: [2], [3]

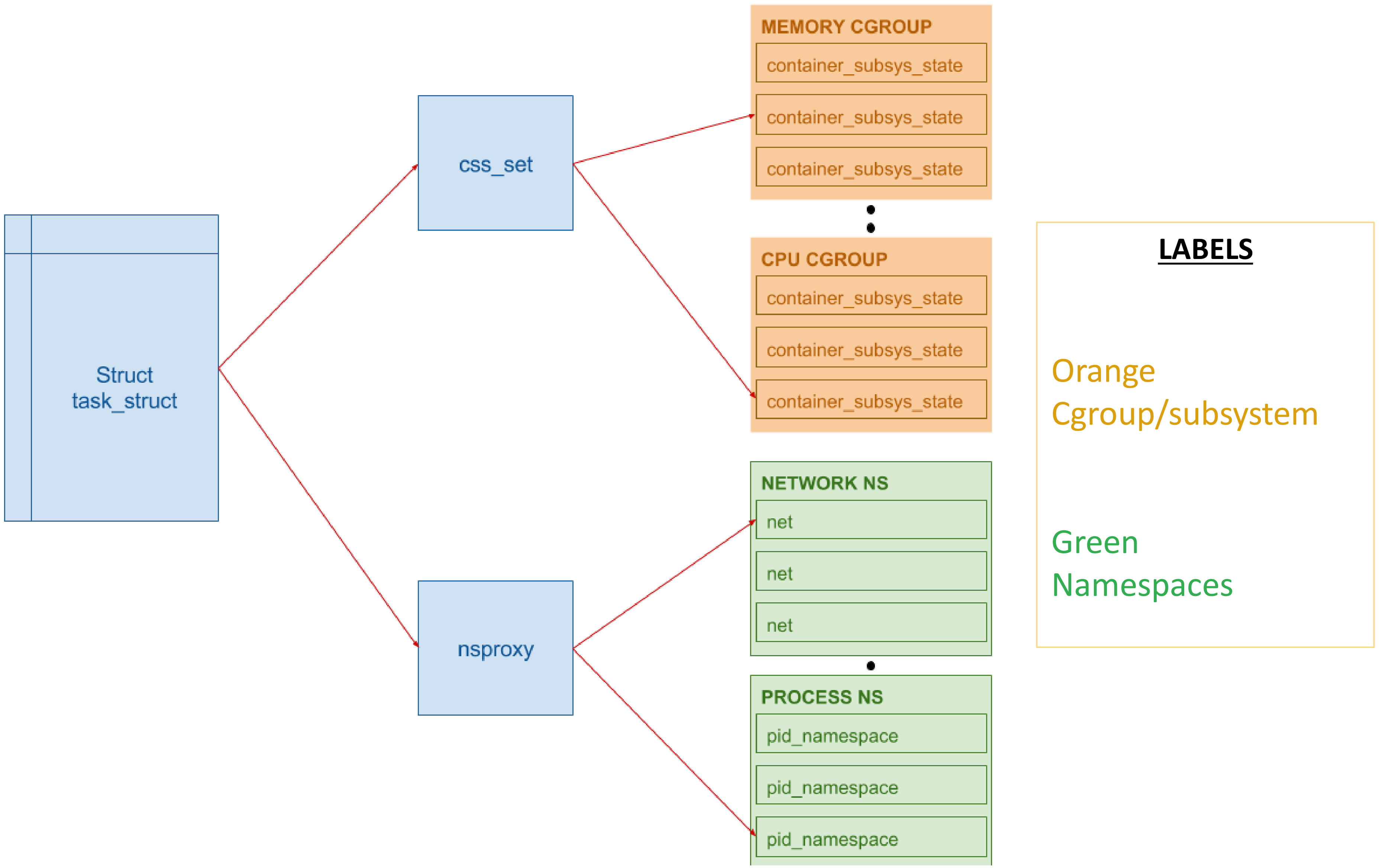


Fig: Kernel Data structure modifications

Container Disk Images

- Provides new mount point – avoid changing data of host
- New ROOTFS – mount namespace
- Smaller than the normal OS-disk image – No kernel
- Disk image could also contain only application

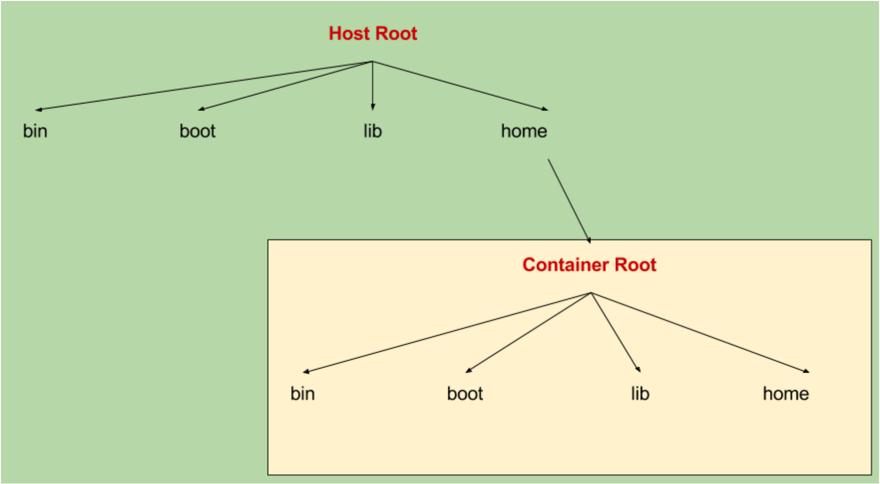
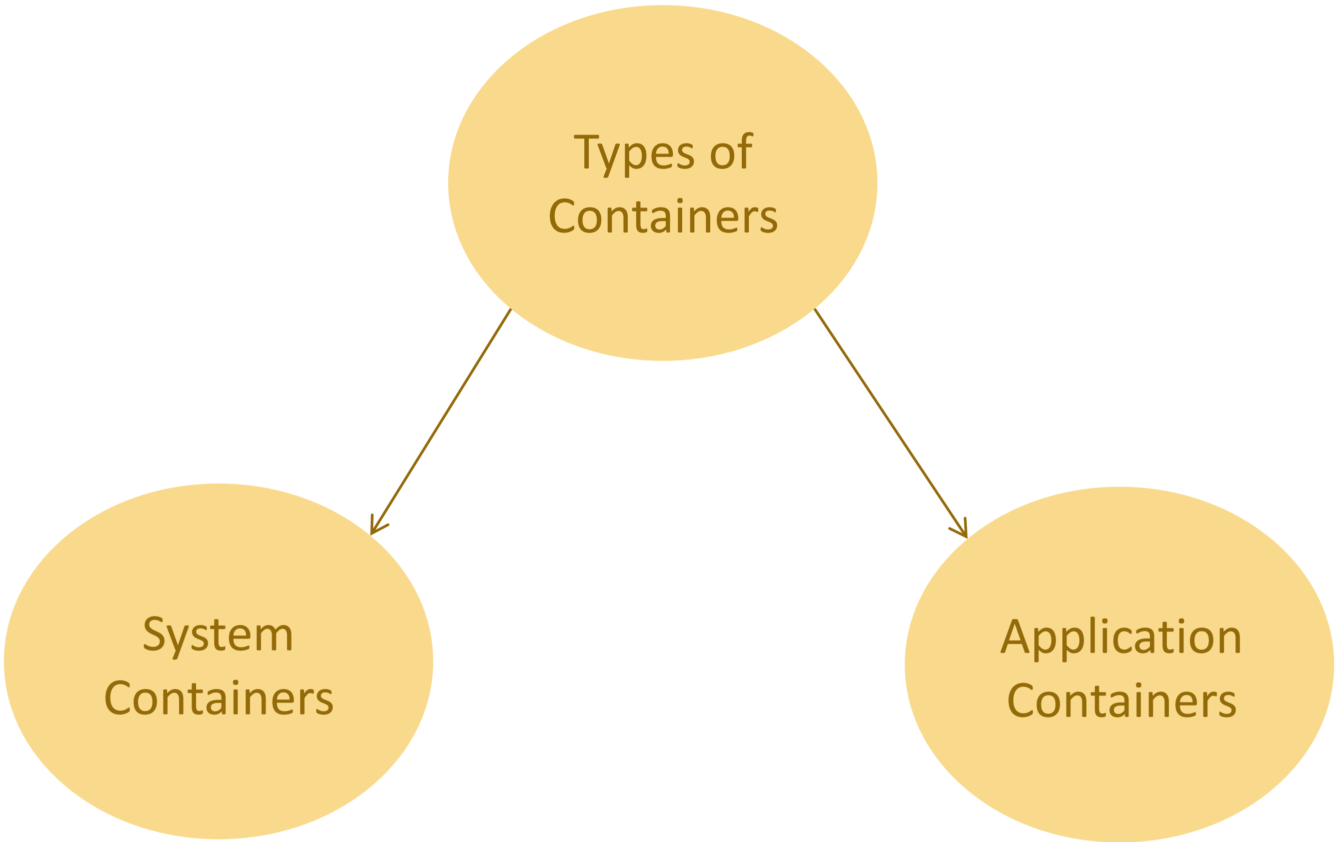


Fig: mount namespace used to mount a new container root


Types of Containers
System Containers
Application Containers

System Containers

- Environment similar to native machine
- Install, configure, run – apps, libraries, demons
- Used by cloud providers
- Have been used for a while
- Examples
    1. Linux Containers (LXC)
    2. Parallels virtuizzo
    3. Solaris zones
    4. Google lmctfy

Reference: [7], [8]

Linux Containers (LXC)

- API to deploy system containers
- Configured via CLI
- Image fetched from online repository – first time
- There after – local cache
- New container – image copied

Application containers

- Develop, build, test, ship and even run apps
- Recent – 2013
- Multiple apps – 1 container for each
- Cloud-native apps
- Examples
    1. Docker
    2. Rocket

Reference: [6]

Docker Architecture

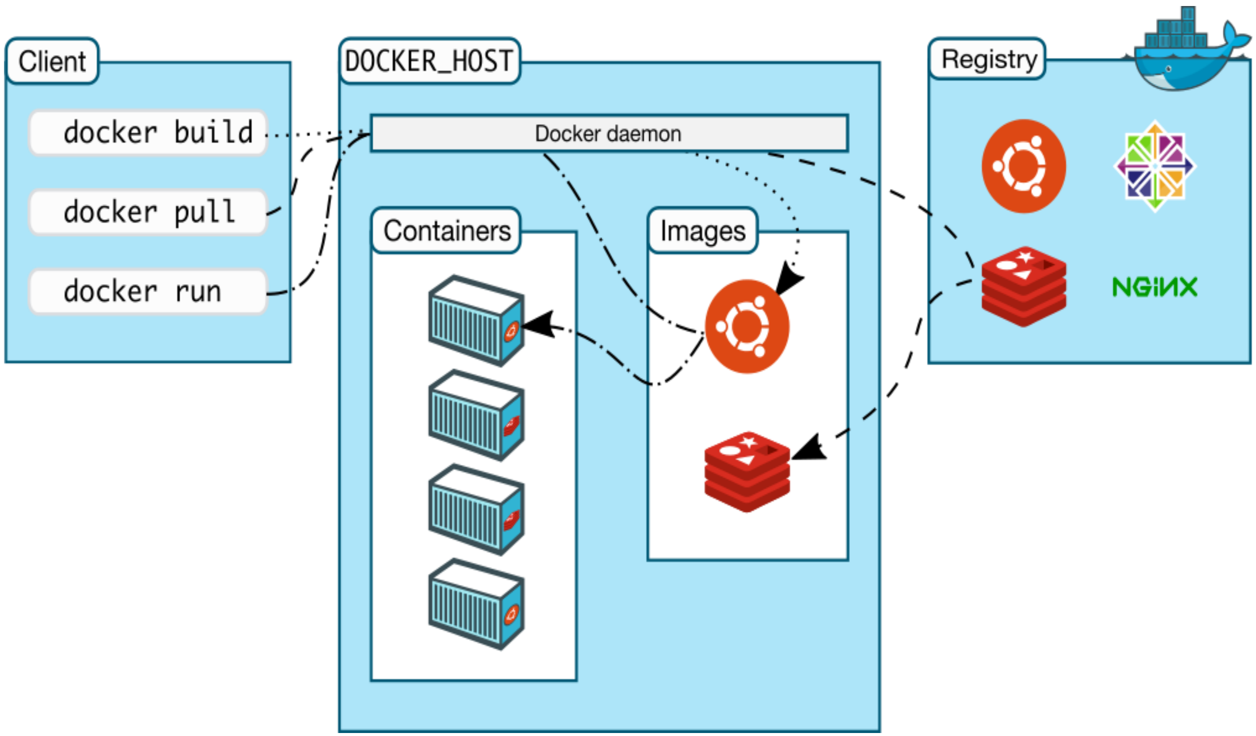


Fig: Docker Architecture, source: [6]

COMPONENTS

1. **Client:** UI to manage containers
2. **Host:** Build & Run containers
3. **Registry:** Image store
4. **Images:** Read-only template
5. **Containers:** Created from image

Docker Image layers

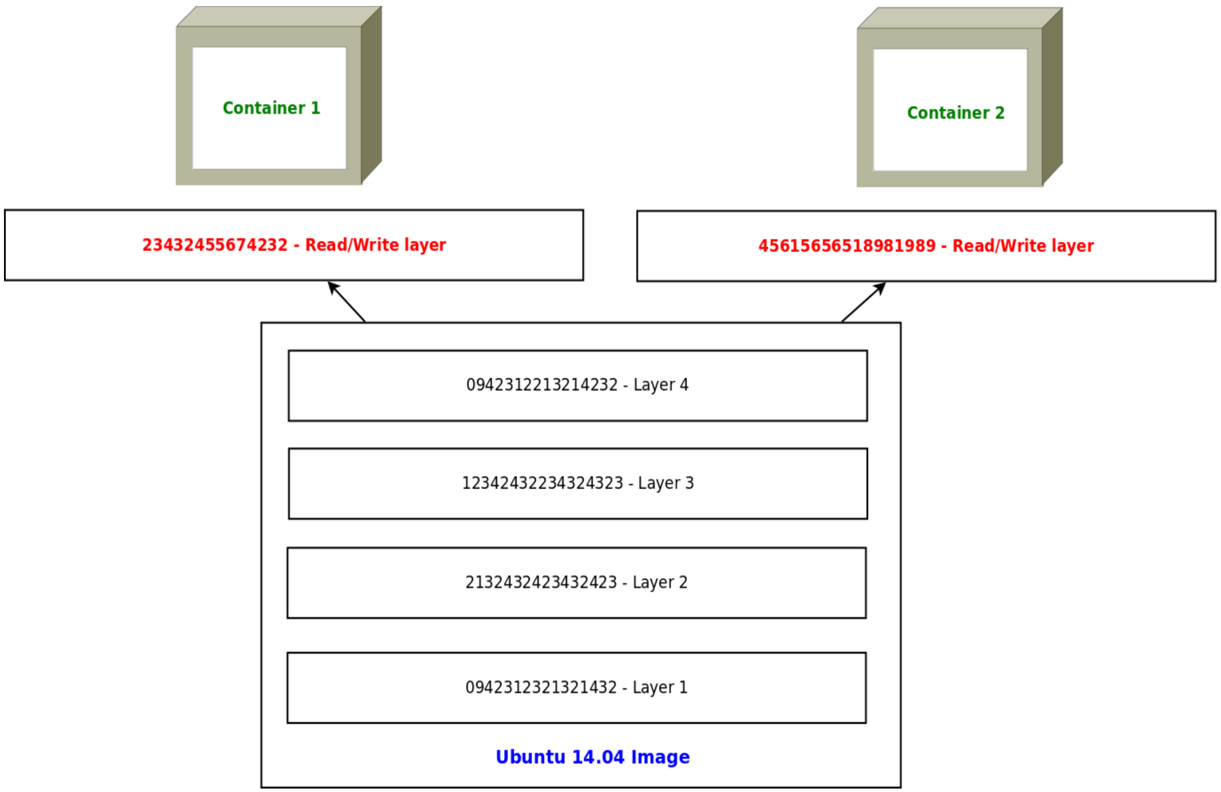


Fig: Docker image layers

POINTS

- Stackable image layers
- Reuse layers
- Copy-On-Write (CoW)
- Container adds Read-Write layer on image
- Commit makes layer read only

Application of containers

- System containers
    1. Cloud providers (IAAS/PAAS)
    2. Data centers
    3. Potentially anywhere instead of VM
- Application containers
    1. HPC clusters
    2. Large companies
    3. App developers

Reference: [5], [10]

Google Borg

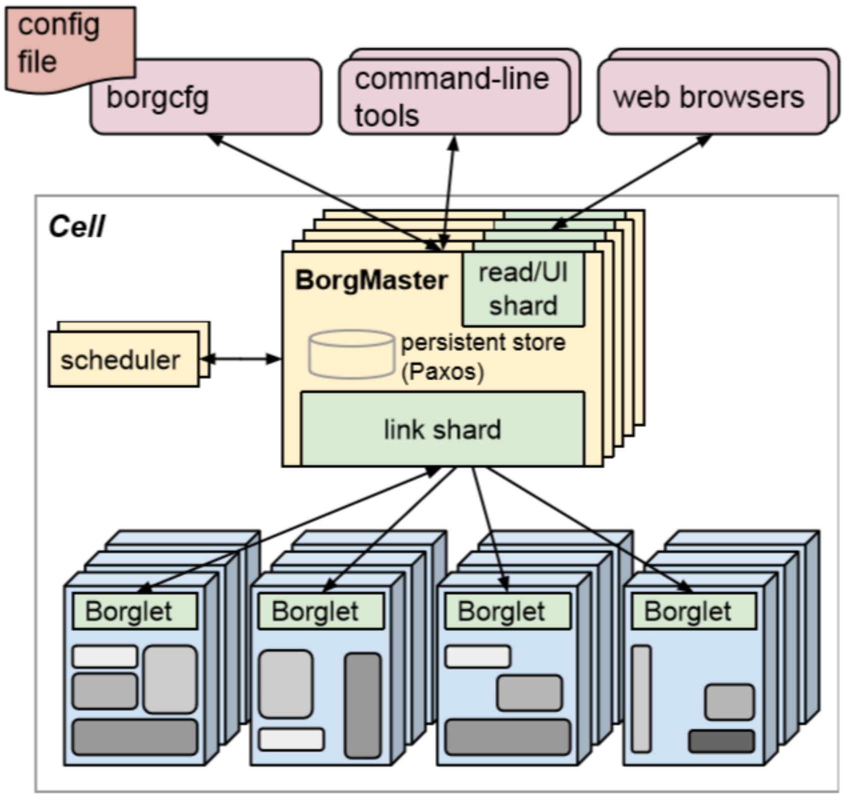


Fig: Cluster management at Google with Borg, source [5]

COMPONENTS

- **BorgMaster:** Central entity used to manage jobs in a cell
- **Scheduler:** Schedules jobs into nodes
- **Borglet:** Local agent that manages jobs in a physical machine
- **Paxos:** Checkpoints state of system to recover in case of Borgmaster crash

Merits and Demerit of containers

MERITS

- Startup latency minimal
- No hardware emulation
- No multiple OS copies
- Overheads - close to native

DEMERITS

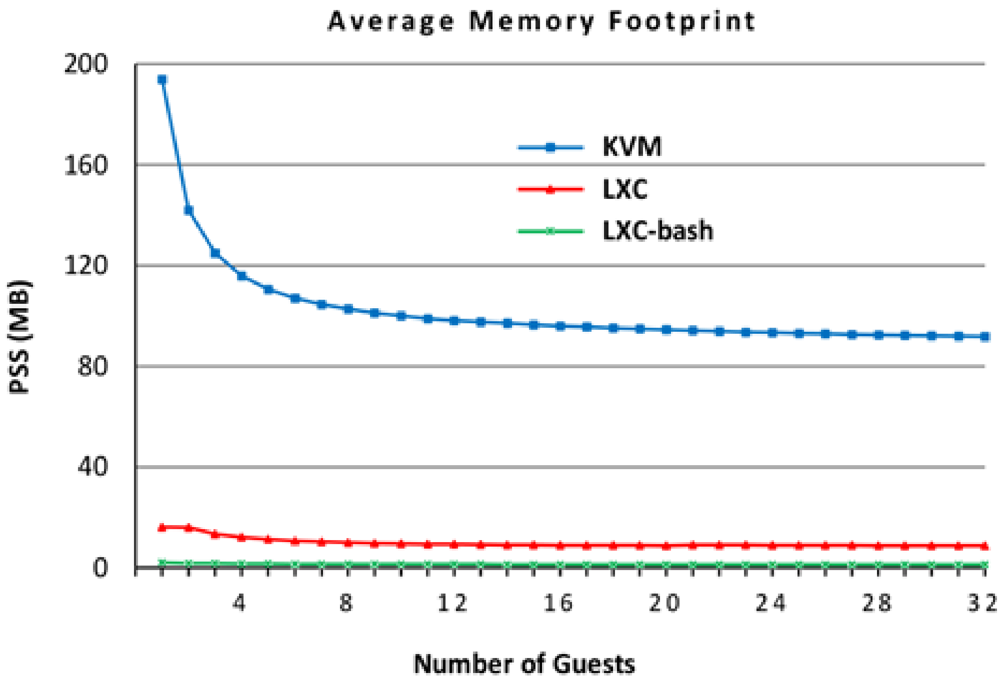
- Only base kernel type containers
- Security

Comparing Containers to VMs

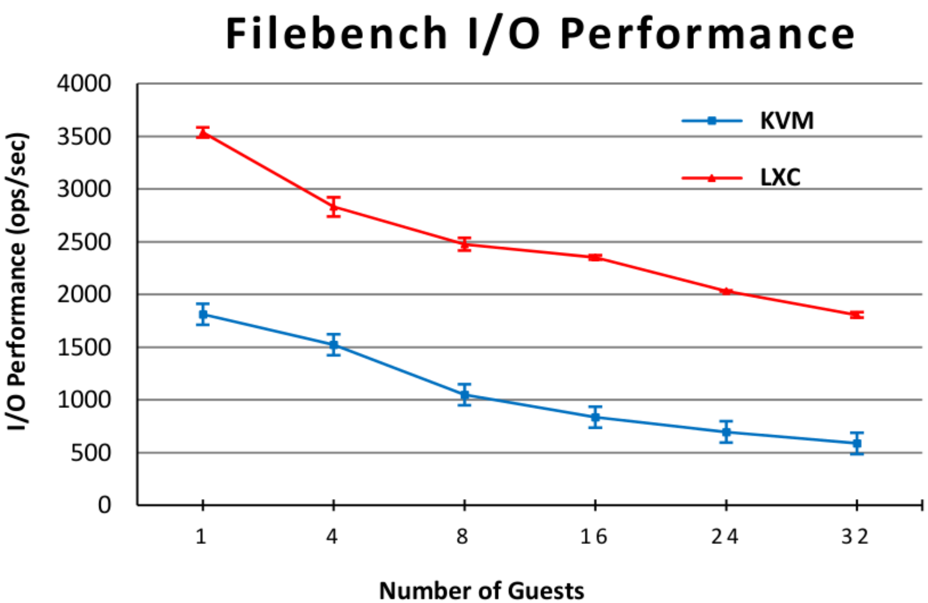
- Claim – Both similar features, containers lesser overheads
- Setup – 1 example from each, compared in same setup
- Performance metrics compared
    1. Start up latency
    2. Memory size
    3. Disk I/O throughput
    4. Network latency
    5. CPU throughput

- Startup Latency – VM typically takes about 50-100x

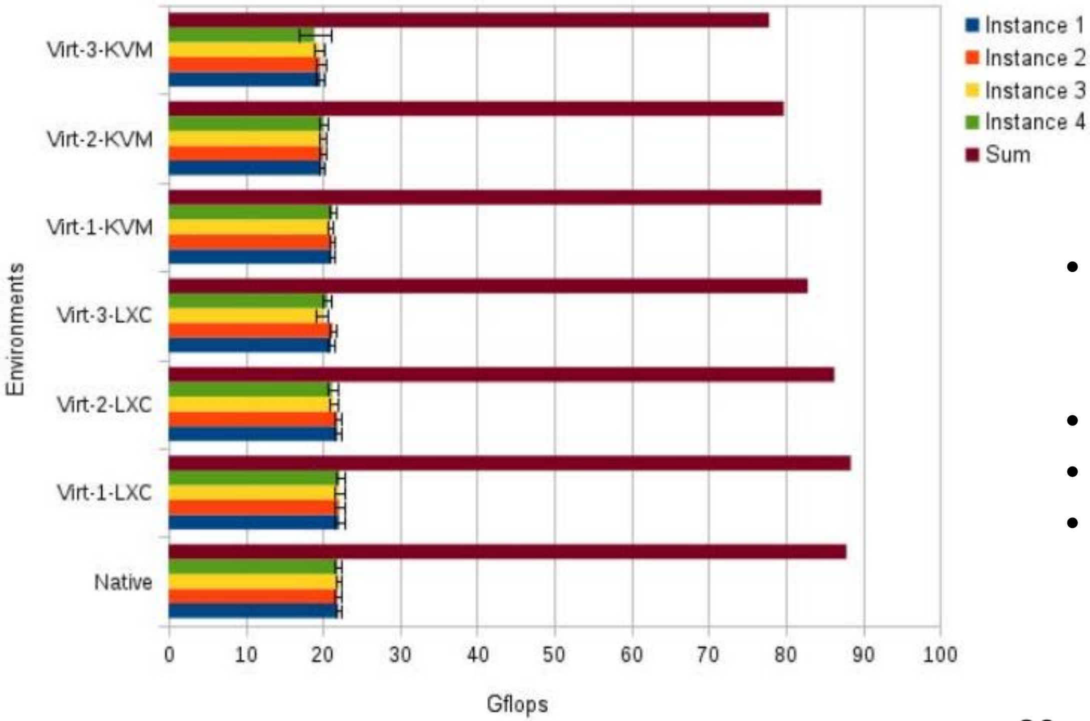
Reference: [9], [10], [11], [12], [13]



- Increasing number of guests and how it effects memory size
- lower the better
- 11-60x better in containers
- Source [9]

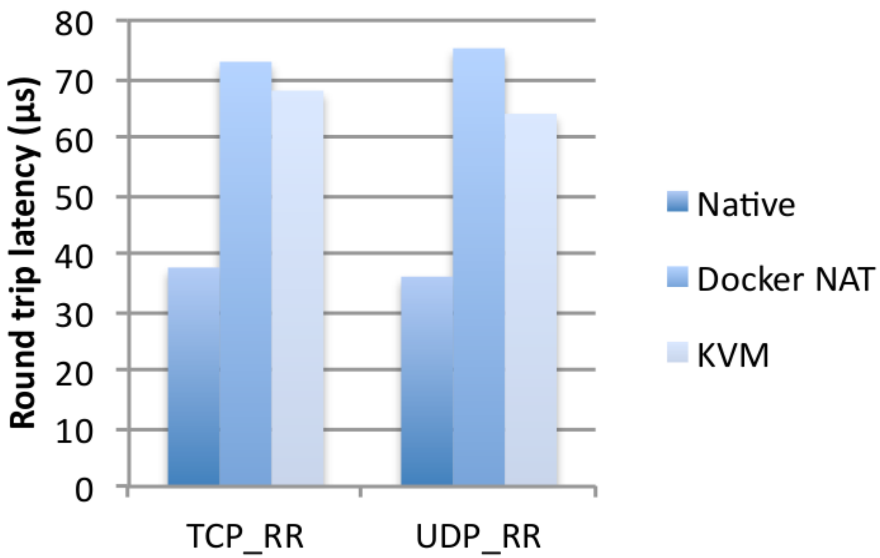


- Increasing number of guests and how it effects I/O throughput
- higher the better
- Optimization: direct map in VM
- source [9]



- Increasing number of guests in HPC environment and how it effects CPU throughput
- Higher the better
- 2-22% lesser in VM
- source [10]

- Effect on RTT – client-server
- lower the better
- VM (80%) > container (100%)
- source [11]



Conclusion

- Performance overheads - Big win
- Tremendous potential
- Limitation of a container is the ability to only run OS of host kernel type

- **Future Scope:**

  Multi subsystem cgroup hierarchies, existing bug fixes, sharing packages with host, vulnerabilities, live-migration, memory reclamation etc.

References

Components of container
[1] P. B. Menage, “Adding generic process containers to the linux kernel," in Proceedings of the Linux Symposium , vol. 2, pp. 45{57, Citeseer, 2007.
[2] M. Kerrisk, “Lwn namespaces overview," 2013.
[3] Michael Kerrisk “namespaces in operation”, https://lwn.net/Articles/531114/, 2013

Container
[4] G. Banga, P. Druschel, and J. C. Mogul, "Resource containers: A new facility for resource management in server systems," in OSDI , vol. 99, pp. 45{58, 1999.
[5] A. Verma, L. Pedrosa, M. Korupolu, D. Oppenheimer, E. Tune, and J. Wilkes, “Large-scale cluster management at google with borg," in Proceedings of the Tenth European Conference on Computer Systems. p. 18, ACM, 2015.
[6] D. Inc., “Docker offical documentation," 2016.
[7] K. Kolyshkin, “Virtualization in linux,“ White paper, OpenVZ , vol. 3, p. 39, 2006.
[8] S. Soltesz, H. Potzl, M. E. Fiuczynski, A. Bavier, and L. Peterson, “Container-based operating system virtualization: a scalable, high-performance alternative to hypervisors," in ACM SIGOPS Operating Systems Review, vol. 41, pp. 275{287, ACM, 2007.

Comparison with VMs

[9] K. Agarwal, B. Jain, and D. E. Porter, “Containing the hype," in Proceedings of the 6th Asia-Pacific Workshop on Systems , p. 8, ACM, 2015.

[10] D. Beserra, E. D. Moreno, P. Takako Endo, J. Barreto, D. Sadok, and S. Fernandes, “Performance analysis of lxc for hpc environments," in Complex, Intelligent, and Software Intensive Systems(CISIS), 2015 Ninth International Conference on, pp. 358{363, IEEE, 2015.

[11] W. Felter, A. Ferreira, R. Rajamony, and J. Rubio, “An updated performance comparison of virtual machines and linux containers," in Performance Analysis of Systems and Software (ISPASS), 2015 IEEE International Symposium On, pp. 171{172, IEEE, 2015.

[12] R. Morabito, J. Kjallman, and M. Komu, "Hypervisors vs. lightweight virtualization: a performance comparison," in Cloud Engineering (IC2E), 2015 IEEE International Conference on , pp. 386{393, IEEE, 2015.

[13] M. S. Rathore, M. Hidell, and P. Sj□odin, "Kvm vs. lxc: comparing performance and isolation of hardware-assisted virtual routers,“ American Journal of Networks and Communications , vol. 2, no. 4, pp. 88{96, 2013

Disk I/O and storage driver optimizations

[14] T. Harter, B. Salmon, R. Liu, A. C. Arpaci-Dusseau, and R. H. Arpaci-Dusseau, “Slacker: Fast distribution with lazy docker containers,"

[15] J. Kang, B. Zhang, T. Wo, C. Hu, and J. Huai, “Multilanes: providing virtualized storage for os-level virtualization on many cores," in Proceedings of the 12th USENIX Conference on File and Storage Technologies (FAST 14), pp. 317{329, 2014.