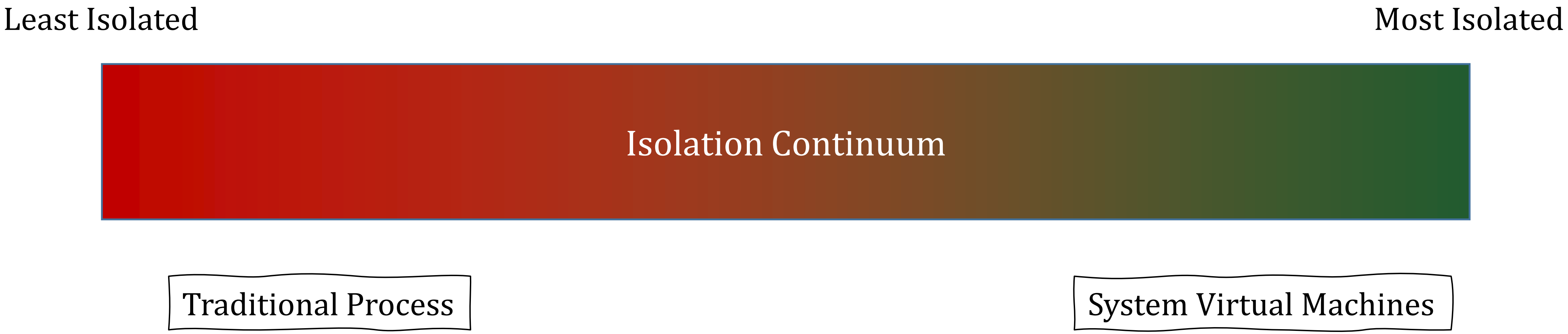
OS-Level Virtualization

Containers

Isolation

- Limiting what and who a process or application can see
- Limiting who can see a process or application



Least Isolated

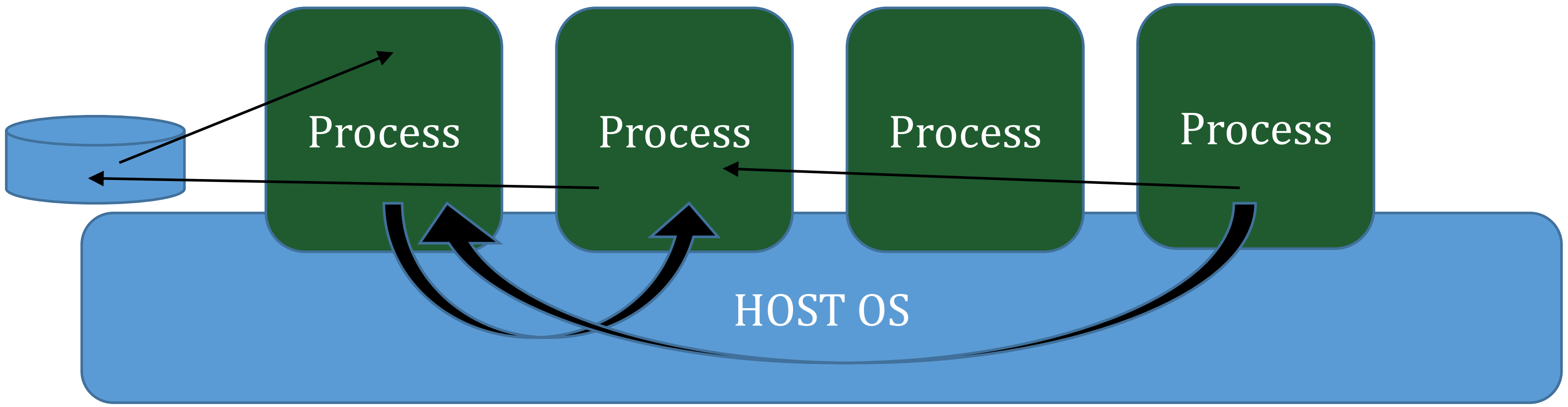
Most Isolated

Isolation Continuum

Traditional Process

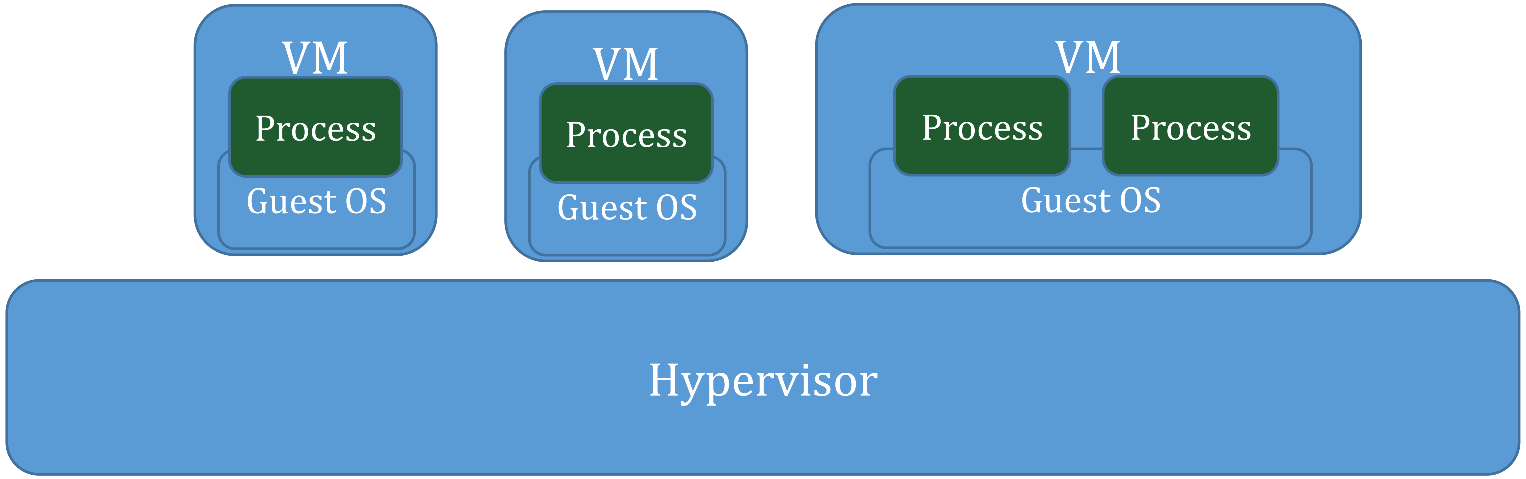
System Virtual Machines

Traditional processes



- Each process gets its own
  - virtual memory
  - One or more virtual CPUs (threads)
  - Access to OS services w/ system calls
- Processes see/share a lot (in an OS-controlled manner)
  - File system, storage, network, I/O
  - Other processes (with IPC)

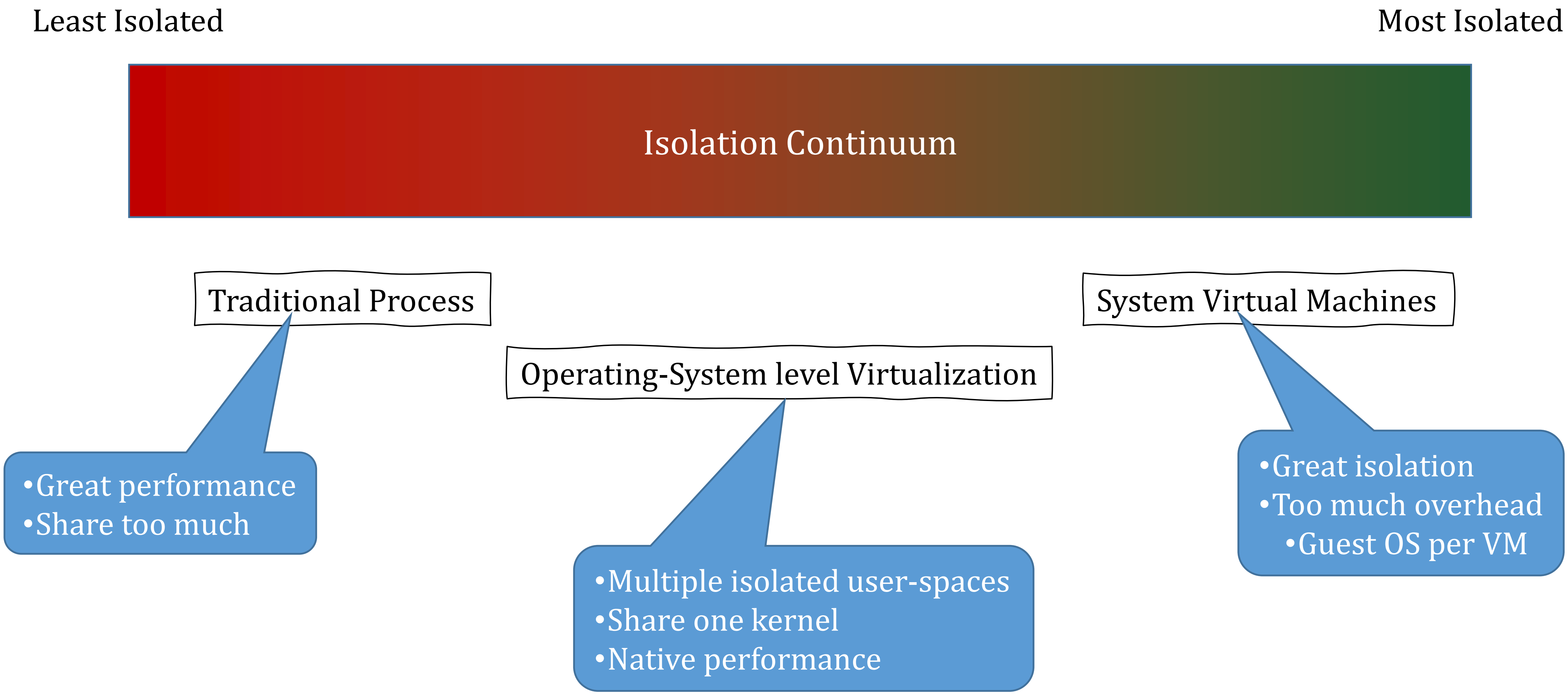
System Virtual Machines



- Co-related processes grouped into VM's
- Each VM has its own:
  - Guest OS
  - Guest Physical Memory
  - One or more virtual CPUs
  - Virtual I/O devices (disk, network, etc.)

- Ideally, co-located VM's don't see or share ANYTHING!

What level to Isolate?



Least Isolated

Most Isolated

Isolation Continuum

Traditional Process

- Great performance
- Share too much

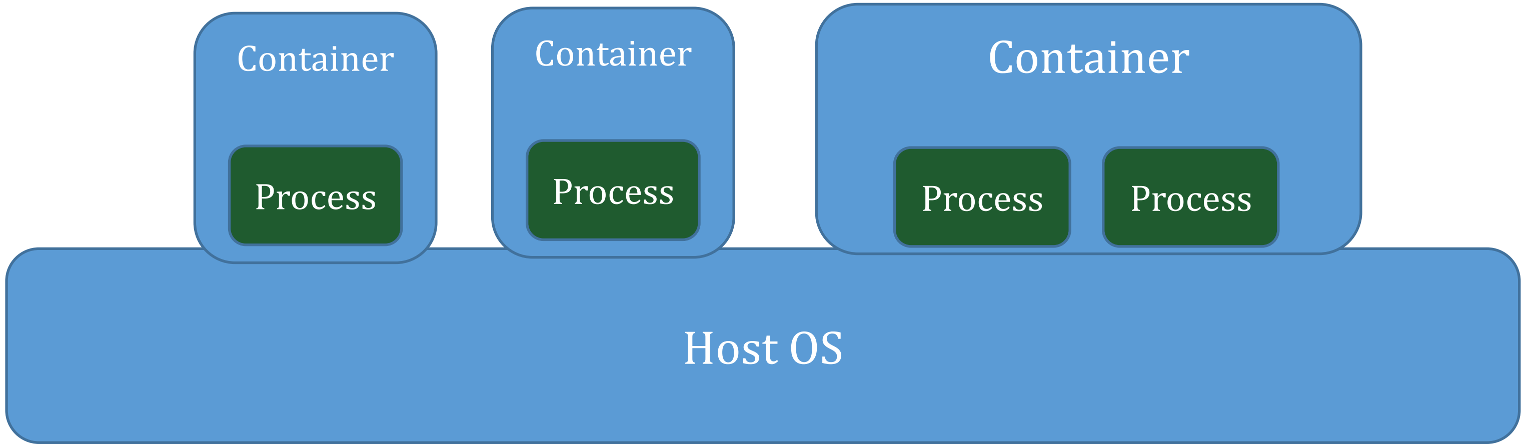
Operating-System level Virtualization

- Multiple isolated user-spaces
- Share one kernel
- Native performance

System Virtual Machines

- Great isolation
- Too much overhead
  - Guest OS per VM

Operating System Level Virtualization



- Containers – a new Virtual level provided by OS
  - Isolate a container from other containers
  - Group traditional processes together and restrict resources they can see
- In Linux: Namespaces and Control Groups

Chroot

- Early (1979) precursor to modern namespaces
- Change root directory for the calling process and it's children
- `>chroot NEWROOT` or `>chroot path`
- Per man chroot – "This call changes an ingredient in the pathname resolution process and does nothing else."
- Intention: Prevent programs from accessing files outside the NEWROOT directory tree (chroot jail)
- Not secure (Lots of ways to escape chroot jail)

FreeBSD Jails (1999)

- Extends chroot to compartmentalize files and other resources
- Jails protect the rest of the system from the jailed process
  - Does not protect the process from the rest of the system!
- Virtualized resources:
  - File system
  - Set of users (included a jail root account)
  - Networking subsystem

- Again, jail escapes were possible

Linux Namespace

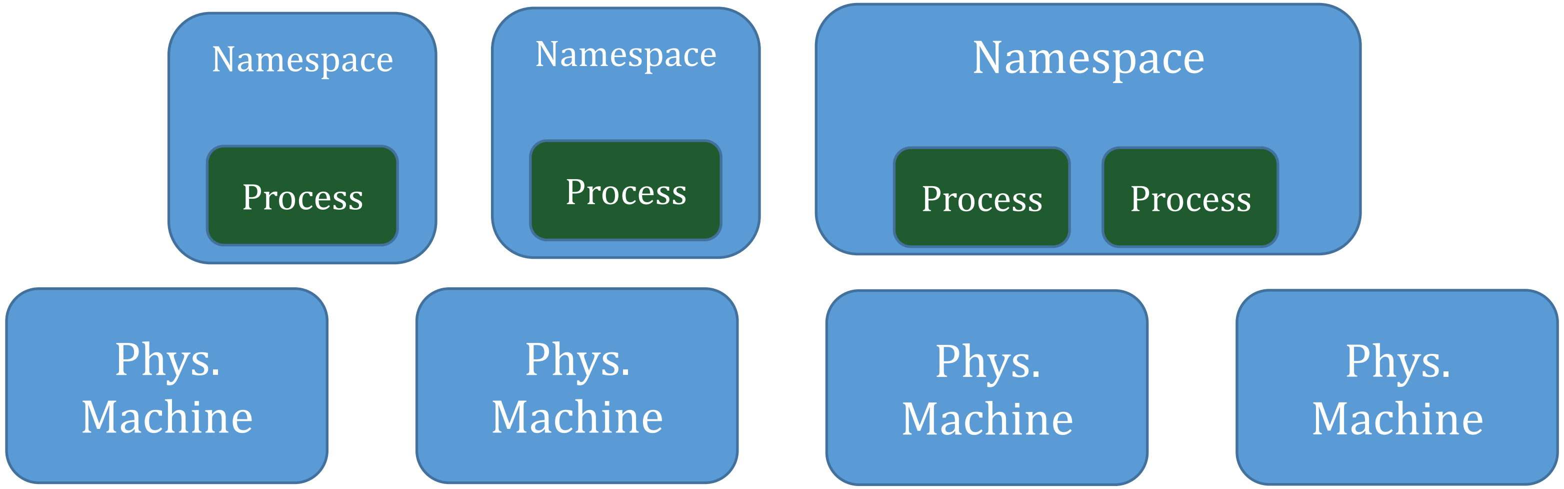
- man namespaces – "A namespace wraps a global system resource in an abstraction that makes it appear to the processes within the namespace that they have their own isolated instance of the global resource."

Namespace	Limitations
PID	limits the set of processes which can see each other
IPC	limits the set of processes which are allowed to communicate with each other
Filesystem	limits which part of the file system is seen by a process group (mount)
Network	unique IP address, host name, domain name, etc. for a process group
User	limits the user and group ID's allowed
…	

Linux Control Groups (Cgroups)

- Performs resource accounting for groups of processes
- Allows administrator to set soft/hard limits on usage of memory, network bandwidth, CPU, etc.
- Typically used with namespaces to control resources for a namespace (see man cgroup_namespaces)

Single System Image



- Extend the notion of namespaces to multiple physical machines
- Multiple machines map to one or more namespaces
  - PID, IPC, and/or Filesystem namespaces
- Process migration – move process from one machine to another without changing it's namespace
- Examples: MOSIX, OpenSSI, Kerrighed