Container tracing: challenges and mechanisms

Progress Report Meeting
December 7, 2017

Loïc Gelle    Michel Dagenais

DORSAL Lab
École polytechnique de Montréal
Context

- Containers tend to partially replace virtual machines
  - Runtime level: Docker, LXC
  - Orchestration level: Kubernetes, Amazon ECS, Docker Swarm
Context

- Containers tend to partially replace virtual machines
- Runtime level: Docker, LXC
- Orchestration level: Kubernetes, Amazon ECS, Docker Swarm
Context

- Containers tend to partially replace virtual machines
- Runtime level: Docker, LXC
- Orchestration level: Kubernetes, Amazon ECS, Docker Swarm
Container analysis landscape

Monitoring at the cluster level by querying

Kubernetes cluster monitoring with Grafana and Prometheus

Source: blog.lwolf.org
Container analysis landscape

Monitoring at the cluster level by querying

Kubernetes node monitoring with Grafana and Prometheus

Source: blog.lwolf.org
Container analysis landscape

Monitoring at the host level by sampling

System containers monitoring with Sysdig and its UI Csysdig

Source: sysdig.org
Containers are not lightweight virtual machines

Handy shortcut to view containers as lightweight virtual machines...

- Similar isolation features from a user level
- Can be easily deployed, backed up, frozen or migrated

...yet their architecture is very different:

- A container shares the same OS kernel as the rest of the system
- A process running in a container is handled just as any other process in the system
- Containers are basically a combination of two Linux kernel features: control groups and namespaces
Containers are not lightweight virtual machines

Handy shortcut to view containers as lightweight virtual machines...

- Similar isolation features from a user level
  - Can be easily deployed, backed up, frozen or migrated

...yet their architecture is very different:

- A container shares the same OS kernel as the rest of the system
- A process running in a container is handled just as any other process in the system
- Containers are basically a combination of two Linux kernel features: control groups and namespaces
Containers are not lightweight virtual machines

Handy shortcut to view containers as lightweight virtual machines...

- Similar isolation features from a user level
- Can be easily deployed, backed up, frozen or migrated

...yet their architecture is very different:

- A container shares the same OS kernel as the rest of the system
- A process running in a container is handled just as any other process in the system
- Containers are basically a combination of two Linux kernel features: control groups and namespaces
Containers are not lightweight virtual machines

Handy shortcut to view containers as lightweight virtual machines...

- Similar isolation features from a user level
- Can be easily deployed, backed up, frozen or migrated

...yet their architecture is very different:

- A container shares the same OS kernel as the rest of the system
- A process running in a container is handled just as any other process in the system
- Containers are basically a combination of two Linux kernel features: control groups and namespaces
Containers are not lightweight virtual machines

Handy shortcut to view containers as lightweight virtual machines...

- Similar isolation features from a user level
- Can be easily deployed, backed up, frozen or migrated

...yet their architecture is very different:

- A container shares the same OS kernel as the rest of the system
- A process running in a container is handled just as any other process in the system
- Containers are basically a combination of two Linux kernel features: control groups and namespaces
Containers are not lightweight virtual machines

Handy shortcut to view containers as lightweight virtual machines...

- Similar isolation features from a user level
- Can be easily deployed, backed up, frozen or migrated

...yet their architecture is very different:

- A container shares the same OS kernel as the rest of the system
- A process running in a container is handled just as any other process in the system
- Containers are basically a combination of two Linux kernel features: control groups and namespaces
Containers are not lightweight virtual machines

Handy shortcut to view containers as lightweight virtual machines...

- Similar isolation features from a user level
- Can be easily deployed, backed up, frozen or migrated

...yet their architecture is very different:

- A container shares the same OS kernel as the rest of the system
- A process running in a container is handled just as any other process in the system
- Containers are basically a combination of two Linux kernel features: control groups and namespaces
Linux cgroups and namespaces

- **cgroups**: Allows to account for or limit resources (cpu, memory, ...) usage for user-defined sets of processes → *how much I can use*
- **namespaces**: Allows to isolate resources (PID, filesystem, ...) for user-defined sets of processes → *what I can use / see*

Running a container is about running normal processes that belong to given cgroups and namespaces...

... which has the ability to isolate these processes and limit their resource consumption.
Linux cgroups and namespaces

- **cgroups**: Allows to account for or limit resources (cpu, memory, ...) usage for user-defined sets of processes
  → *how much I can use*

- **namespaces**: Allows to isolate resources (PIDs, filesystems, ...) for user-defined sets of processes
  → *what I can use / see*

Running a container is about running normal processes that belong to given cgroups and namespaces...

... which has the ability to isolate these processes and limit their resource consumption.
Linux cgroups and namespaces

- **cgroups**: Allows to account for or limit resources (cpu, memory, ...) usage for user-defined sets of processes
  → *how much I can use*

- **namespaces**: Allows to isolate resources (PIDs, filesystems, ...) for user-defined sets of processes
  → *what I can use / see*

Running a container is about running normal processes that **belong to given cgroups and namespaces**...

... which has the ability to isolate these processes and limit their resource consumption.
Linux cgroups and namespaces

- **cgroups**: Allows to account for or limit resources (cpu, memory, ...) usage for user-defined sets of processes
  → *how much I can use*

- **namespaces**: Allows to isolate resources (PIDs, filesystems, ...) for user-defined sets of processes
  → *what I can use / see*

Running a container is about running normal processes that belong to given cgroups and namespaces...

... which has the ability to isolate these processes and limit their resource consumption.
**Project roadmap**

- Instrument the kernel to get **cgroups and namespaces** trace information (ongoing!)
- Use this information to build **container-specific views** of a system
- Design **useful analyses** for containers at the runtime level
- **Scale up** to the orchestration level
Project roadmap

- Instrument the kernel to get **cgroups and namespaces** trace information (ongoing!)
- Use this information to build **container-specific views** of a system
- Design **useful analyses** for containers at the runtime level
- **Scale up** to the orchestration level
Project roadmap

- Instrument the kernel to get **cgroups and namespaces** trace information (ongoing!)
- Use this information to build **container-specific views** of a system
- Design **useful analyses** for containers at the runtime level
- **Scale up** to the orchestration level
Project roadmap

- Instrument the kernel to get **cgroups and namespaces** trace information (ongoing!)
- Use this information to build **container-specific views** of a system
- Design **useful analyses** for containers at the runtime level
- **Scale up** to the orchestration level
Analyses ideas

- Understand why a process in a container is being throttled back (cgroup limitations or system overload?)
- Understand why access to some resources from a container fail (namespace isolation?)
- "What if"-like analysis: what if I change my container resources? What impact on the critical path length?

Suggestions and use cases are welcome throughout the project!
Analyses ideas

- Understand why a process in a container is being throttled back (cgroup limitations or system overload?)
- Understand why access to some resources from a container fail (namespace isolation?)
- "What if"-like analysis: what if I change my container resources? What impact on the critical path length?

Suggestions and use cases are welcome throughout the project!
Analyses ideas

- Understand why a process in a container is being throttled back (cgroup limitations or system overload?)
- Understand why access to some resources from a container fail (namespace isolation?)
- "What if"-like analysis: what if I change my container resources? What impact on the critical path length?

Suggestions and use cases are welcome throughout the project!
Analyses ideas

- Understand why a process in a container is being throttled back (cgroup limitations or system overload?)
- Understand why access to some resources from a container fail (namespace isolation?)
- "What if"-like analysis: what if I change my container resources? What impact on the critical path length?

Suggestions and use cases are welcome throughout the project!
Thank you!

Questions?

loic.gelle@polymtl.ca