CONTAINER-BASED ARCHITECTURE PERFORMANCE ANALYSIS

Pierre-Frédérick DENYS

Monday 9 December 2019

Agenda

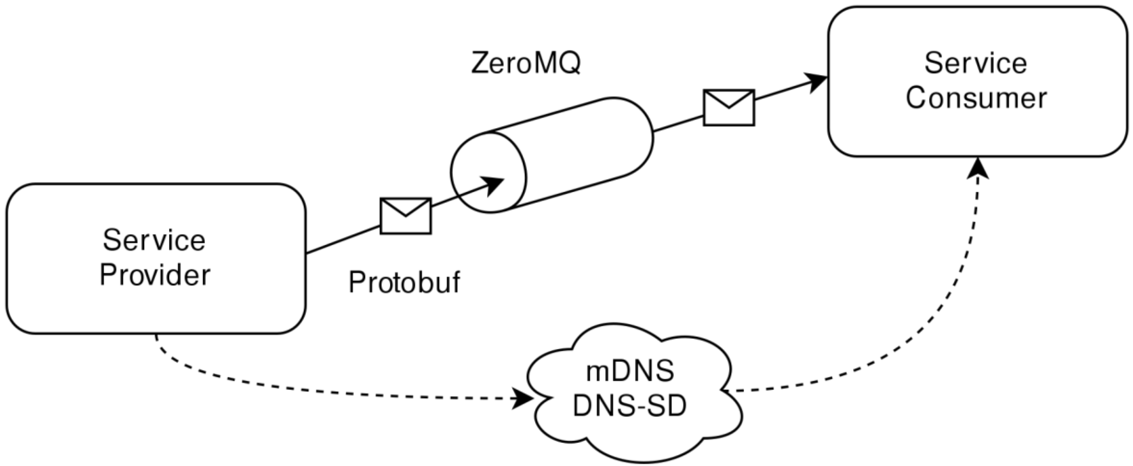
- Monitoring messaging in a container-based environnement
- Container architecture for HPC and monitoring

“

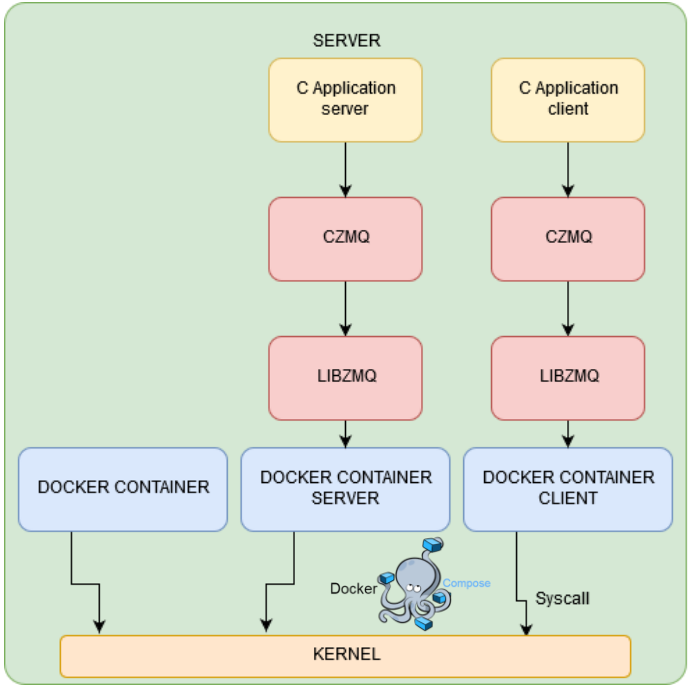
Part 1 : containers messaging analysis

“ How to monitor messaging between containers and prevent data loss ?

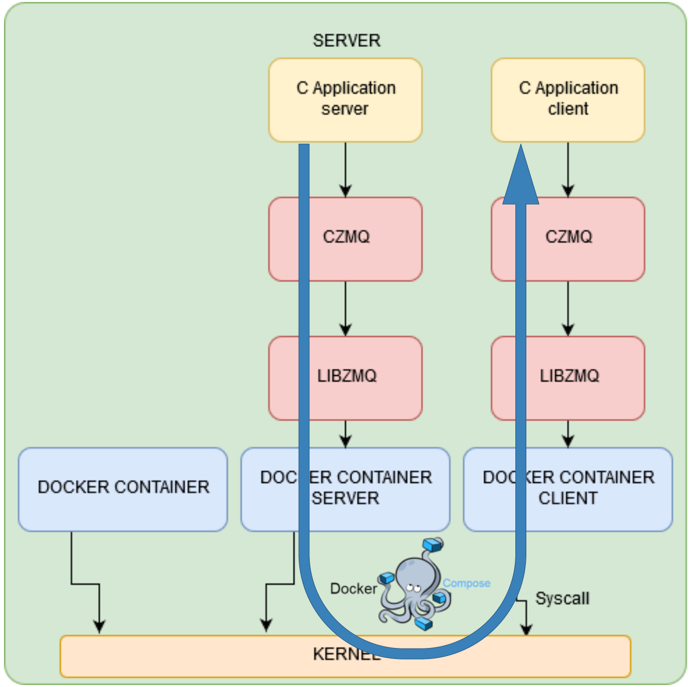
CASE STUDY : zeroMQ as message broker



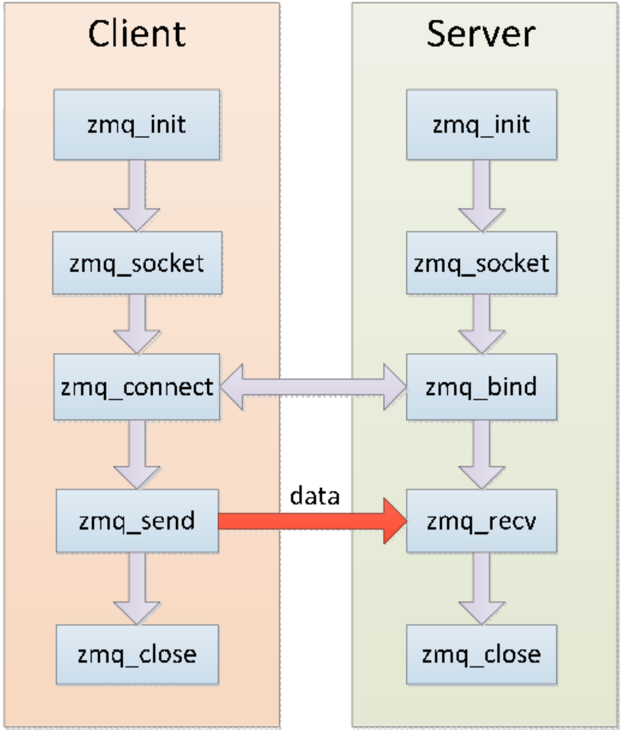
CASE STUDY : zeroMQ as message broker



CASE STUDY : zeroMQ as message broker



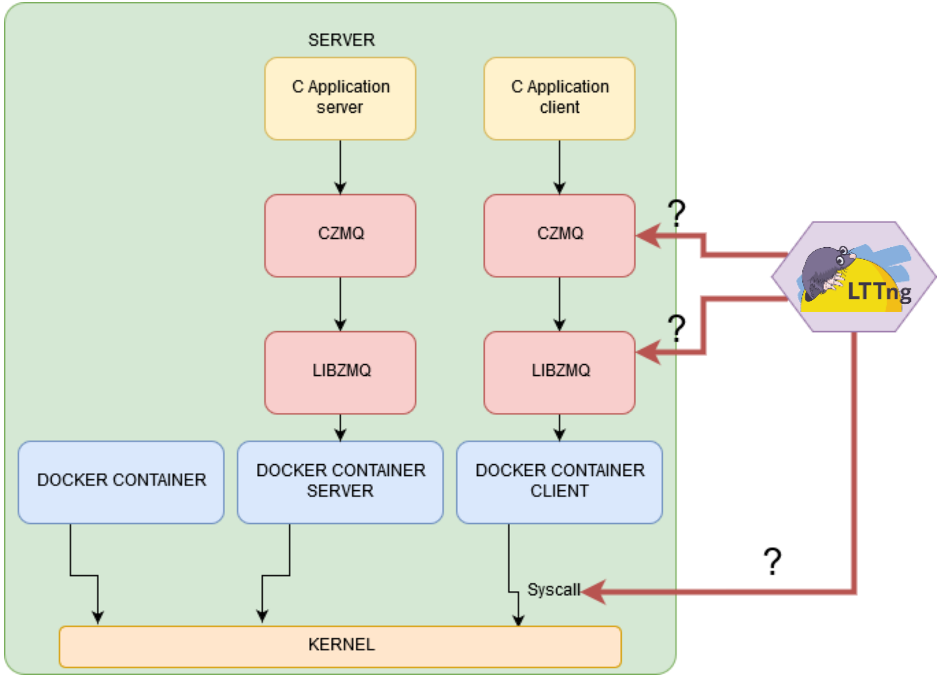
CASE STUDY : zeroMQ as message broker



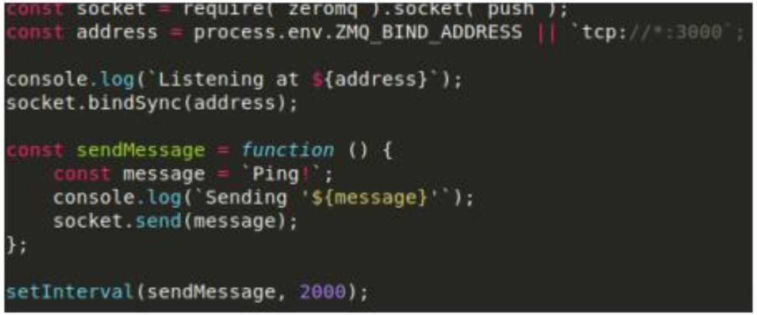
“

What is the best place to instrument the messaging system ?

CASE STUDY : zeroMQ as message broker

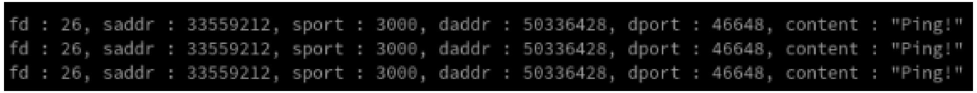


CASE STUDY : zeroMQ as message broker



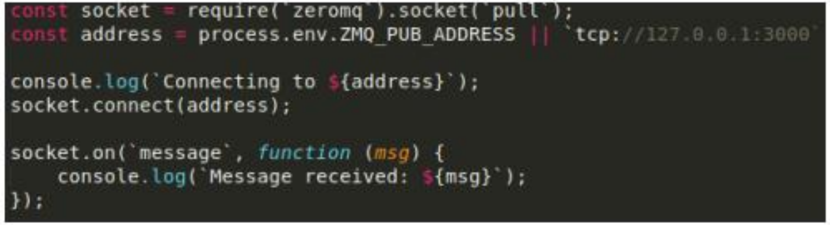
```
const socket = require(`zeromq`).socket(`push`);
const address = process.env.ZMQ_BIND_ADDRESS || `tcp://*:3000`;

console.log(`Listening at ${address}`);
socket.bindSync(address);

const sendMessage = function () {
    const message = `Ping!`;
    console.log(`Sending '${message}'`);
    socket.send(message);
};

setInterval(sendMessage, 2000);
```

FIGURE 7 – Code node.js exécuté sur le container serveur



```
fd : 26, saddr : 33559212, sport : 3000, daddr : 50336428, dport : 46648, content : "Ping!"
fd : 26, saddr : 33559212, sport : 3000, daddr : 50336428, dport : 46648, content : "Ping!"
fd : 26, saddr : 33559212, sport : 3000, daddr : 50336428, dport : 46648, content : "Ping!"
```

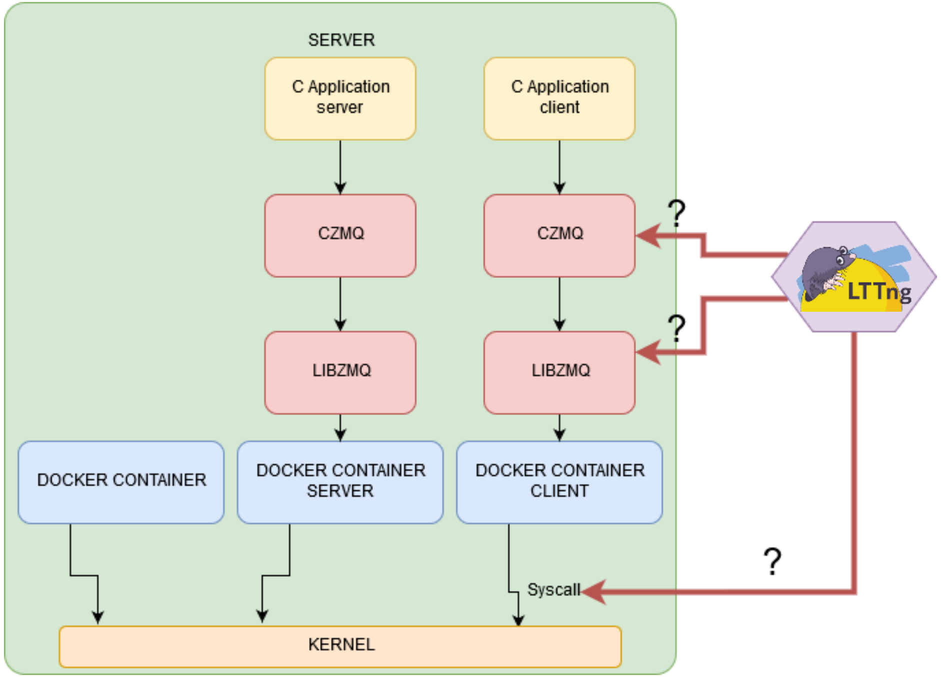
(ip en décimal et inversée (notation cpu) dans le contexte 33559212 -> 172.18.0.2)



```
const socket = require(`zeromq`).socket(`pull`);
const address = process.env.ZMQ_PUB_ADDRESS || `tcp://127.0.0.1:3000`

console.log(`Connecting to ${address}`);
socket.connect(address);

socket.on(`message`, function (msg) {
    console.log(`Message received: ${msg}`);
});
```

FIGURE 8 – Code node.js exécuté sur le container client

CASE STUDY : zeroMQ as message broker



CASE STUDY : zeroMQ as message broker

What is done actually :

- Data collection at kernel level, libzmq level, czmq level
- Patch for zeroMQ (ZeroMQ instrumentation and LTTng data collection)

What I will work on :

- Data aggregation between containers
- Data vizualisation with trace compass
- Generalization to other asynchronous messaging systems

“

Part 2 : containers for high performance computing

“

What is HPC and its needs ?

HPC

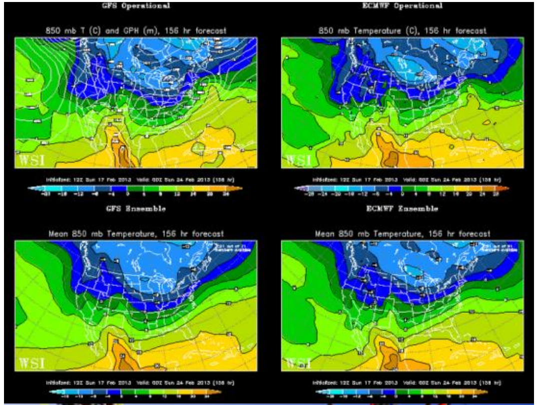
Supercomputers are designed to run large workloads — such as weather forecasting or molecular modeling — as a single application across hundreds or even thousands of servers. Despite the initial similarities to cloud computing, some fundamental differences do exist between the two architectures

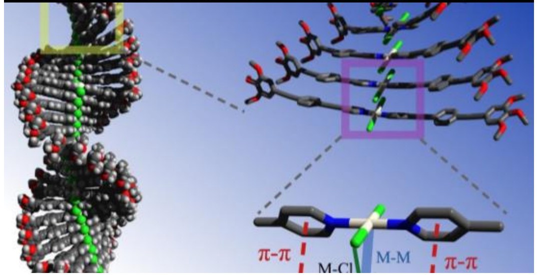
HPC programs tend to be tightly-coupled. They may have multiple components, but they all need access to a shared memory space

Unlike most Docker setups today, in an HPC setup, multiple nodes will be reading and writing to a single shared file system.

“HPC needs to run on shared resources” Kniep, HPC advisory council 2018

HPC workflow doesn't benefit much from process isolation

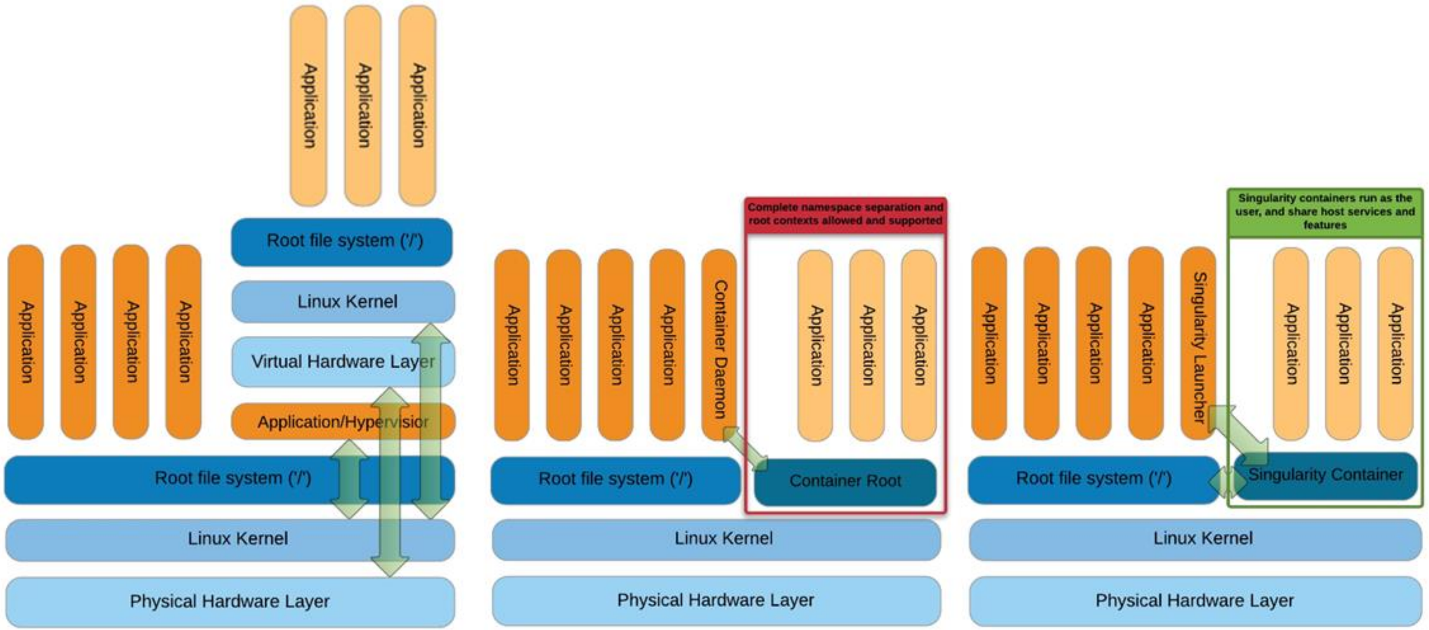

GFS Operational
ECMWF Operational
850 mb T (C) and GPH (m), 156 hr forecast
850 mb Temperature (C), 156 hr forecast
WSI
GFS Ensemble
ECMWF Ensemble
Mean 850 mb Temperature, 156 hr forecast
Mean 850 mb Temperature, 156 hr forecast



Singularity vs docker

	Docker	Singularity
Main problem being addressed	DevOps, microservices. Enterprise applications	Application portability (single image file, contain all dependencies) Reproducibility, run cross platform, provide support for legacy OS and apps.
Interaction with docker		Work completely independent of Docker Ability to import docker images, convert them to singularity images, or run docker container directly
Group of users	Developers/DevOps	
	Scientific Application Users	
Scheduler	Swarm, kubernetes	Agnostic, run singularity as a job. Works well with Slurm,
Namespace isolation paradigm	Process, Network, user space are isolated by default, but can be shared via cli arg	process namespace isolation can easily be enabled. Default share Process Namespace, username space

Singularity vs docker



General VM
eg ESXi

General Container
eg Docker

HPC Container
Singularity

Solution for HPC

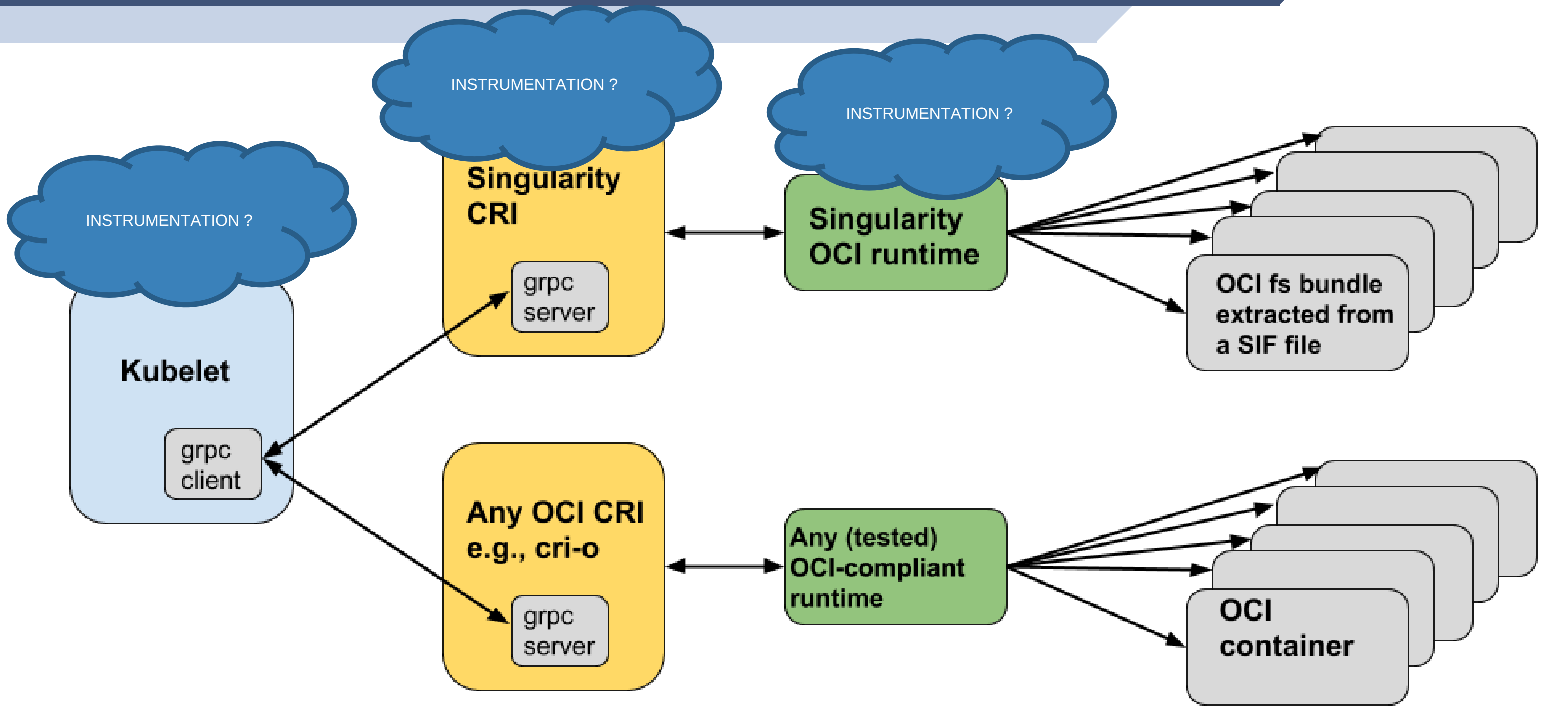
Combine both Kubernetes and HPC workload-management systems to fully meet the HPC requirements

Slurm + singularity + Kubernetes = HPC

“

How to monitor these infrastructures ?

Solution for HPC



SINGULARITY CLUSTER INSTRUMENTATION

In progress :

- Setup a small HPC cluster with kubernetes, slurm and singularity

Work to do :

- Instrument components with LTTng
- Collect data with LTTng
- Add a module to Trace Compass to see data

=> Tools for HPC monitoring (scientific computing) , and by extension for kubernetes monitoring (business computing)

Thank you for listening !