Development and evaluation of a Kubernetes cluster simulator based on Batsim

Presented by: Théo Larue
Supervised by: Olivier Richard & Michael Mercier

Université Grenoble Alpes

August 31, 2020

Table of contents

Introduction

Computer infrastructures

Distributed systems, many domains: Grid, Edge, HPC, Cloud, P2P, Volunteer, Cluster.
These systems are very complex. (some graph to illustrate the increase in size of supercomputers)

Studying distributed systems

Why studying these infra? To test a system performances under varying loads, applications, scheduling policies, system size and topology. Or to develop new RJMS or research new scheduling algorithms.

Studying distributed systems

How to study these infra? Too many elements and interactions to consider, so no theoretical study.
Real experiments are too costly (both in time and resources) and not reproducible.

Studying distributed systems

Emulation resolves the issue of reproducibility.
Simulation resolves both reproducibility and scalability issues.

Literature review

Domain specific simulators

refs on domain specific simulators, summed up in a table.
Explain briefly the concept behind some of them.

Software specific simulators

YARNSim, SLURM simulator

Publication specific simulators

"Publish and perish" - Milian Poquet

SimGrid

SimGrid: Versatile, scalable, accurate.
Cpu = a computation speed.
Storage = a seek time and a data transfert rate.
Network = a flow model, modeling bandwith sharing behaviors of actual protocol.
Simple models but thoroughly validated.

Batsim

Aimed at studying RJMS.
Strong decoupling decision process / simulator.

Batsim - related work

Alea: modular, extensible.
Accasim: supports additional information (temperature, power consuption). Very efficient in terms of simulation time and memory usage.
Both outperform Batsim in terms of scalability. However it is not fair to compare them on this point because Batsim relies on well thought models, when these two only implement delay jobs.

Kubernetes

Explain containers real quick.
Container orchestration software, description based.

Kubernetes cluster simulation

k8s-cluster-simulator: open source, student project, delay jobs. Schedulers provided via a Go interface.
joySim: closed-source, fully fledged kubernetes cluster simulator, service oriented (mock nodes).

Integrating Kubernetes schedulers to Batsim

Technical challenges

1. Integration with Kubernetes.

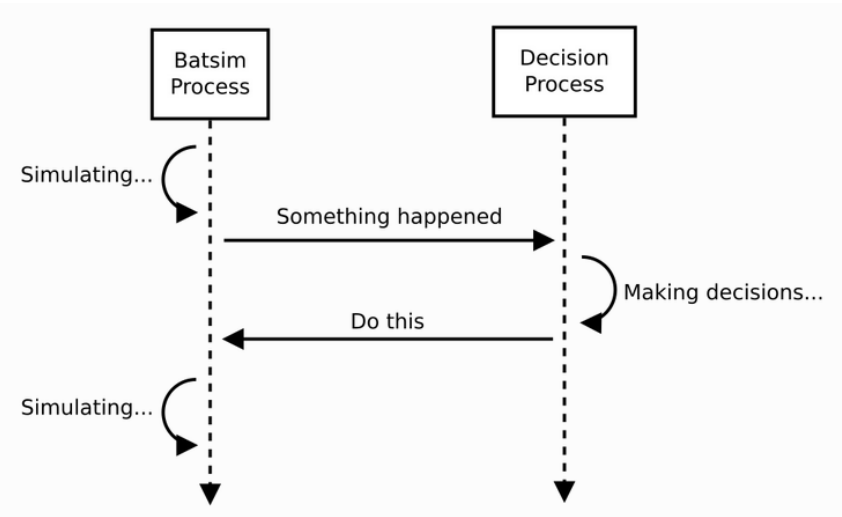
Technical challenges

1. Integration with Kubernetes.
2. Intercepting scheduler time.

Technical challenges

1. Integration with Kubernetes.
2. Intercepting scheduler time.
3. Time synchronization between Batsim and the scheduler.

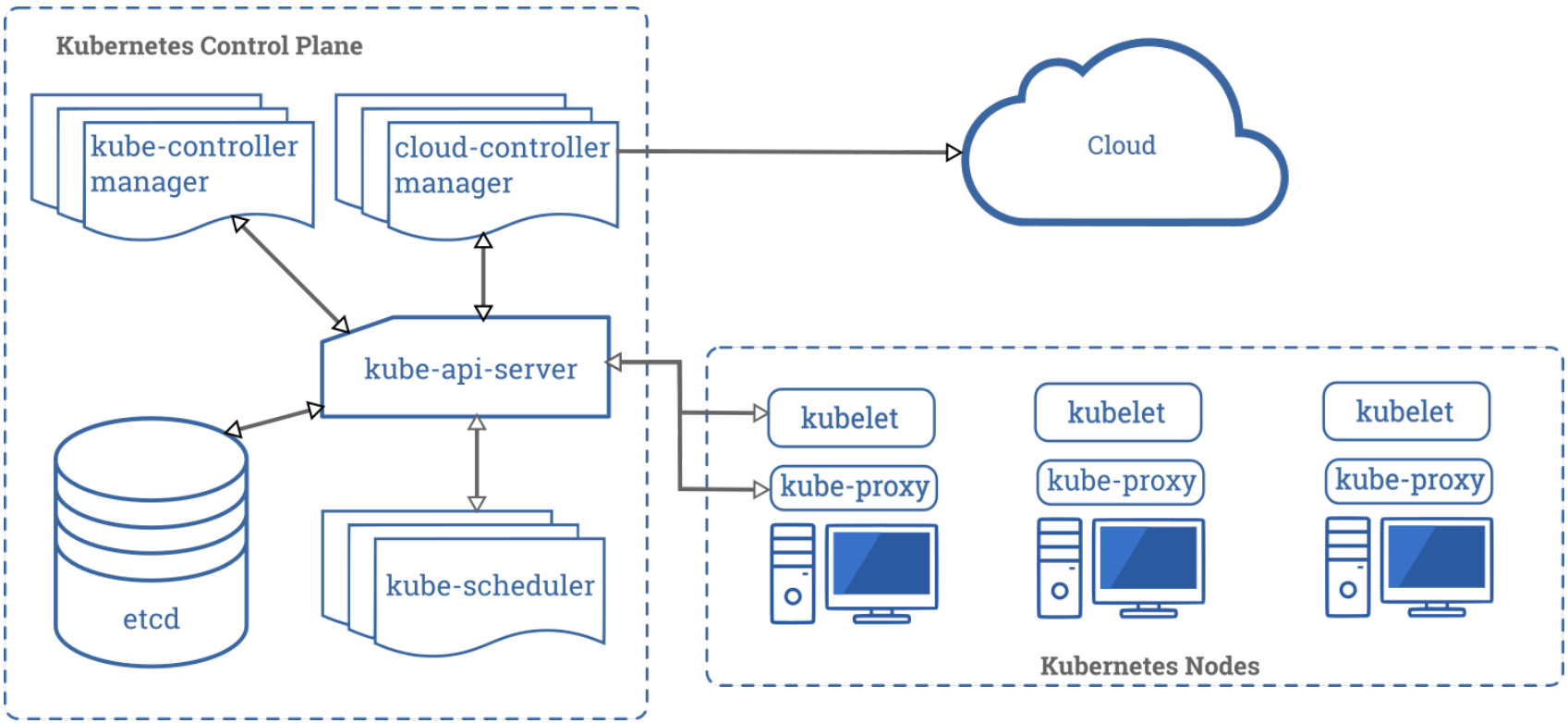
Batsim concepts



source `https://batsim.readthedocs.io`

Batsim events and protocol.
User defined workloads.
(insert json examples?)

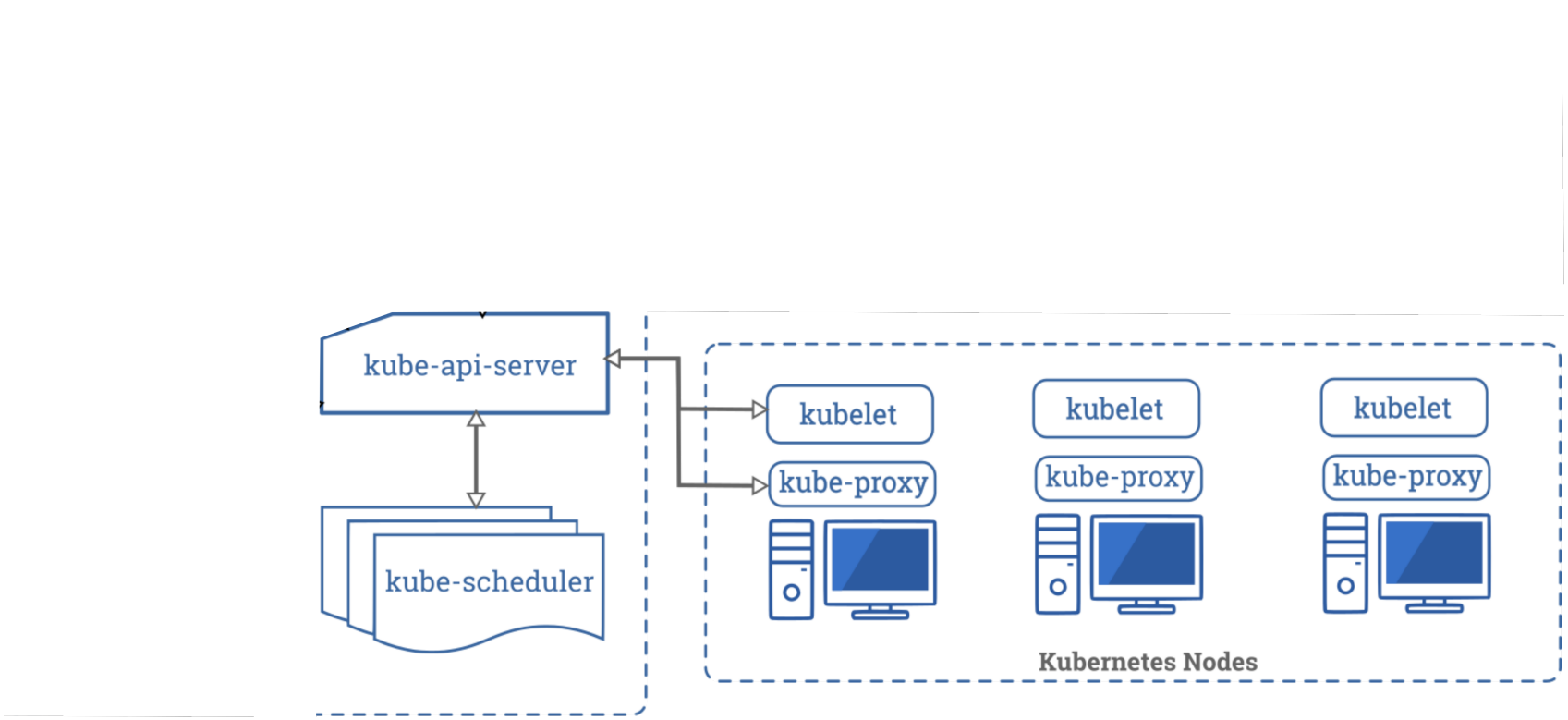
Kubernetes concepts



source: `https://kubernetes.io/docs/concepts/overview/components/`

Kubernetes components.

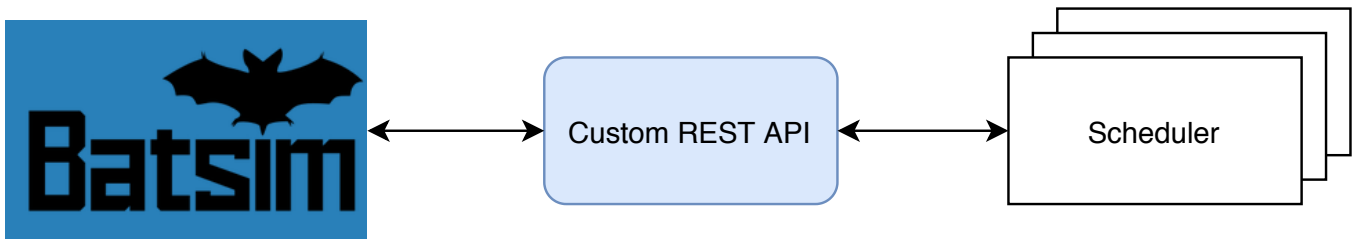
Kubernetes concepts



source: `https://kubernetes.io/docs/concepts/overview/components/`

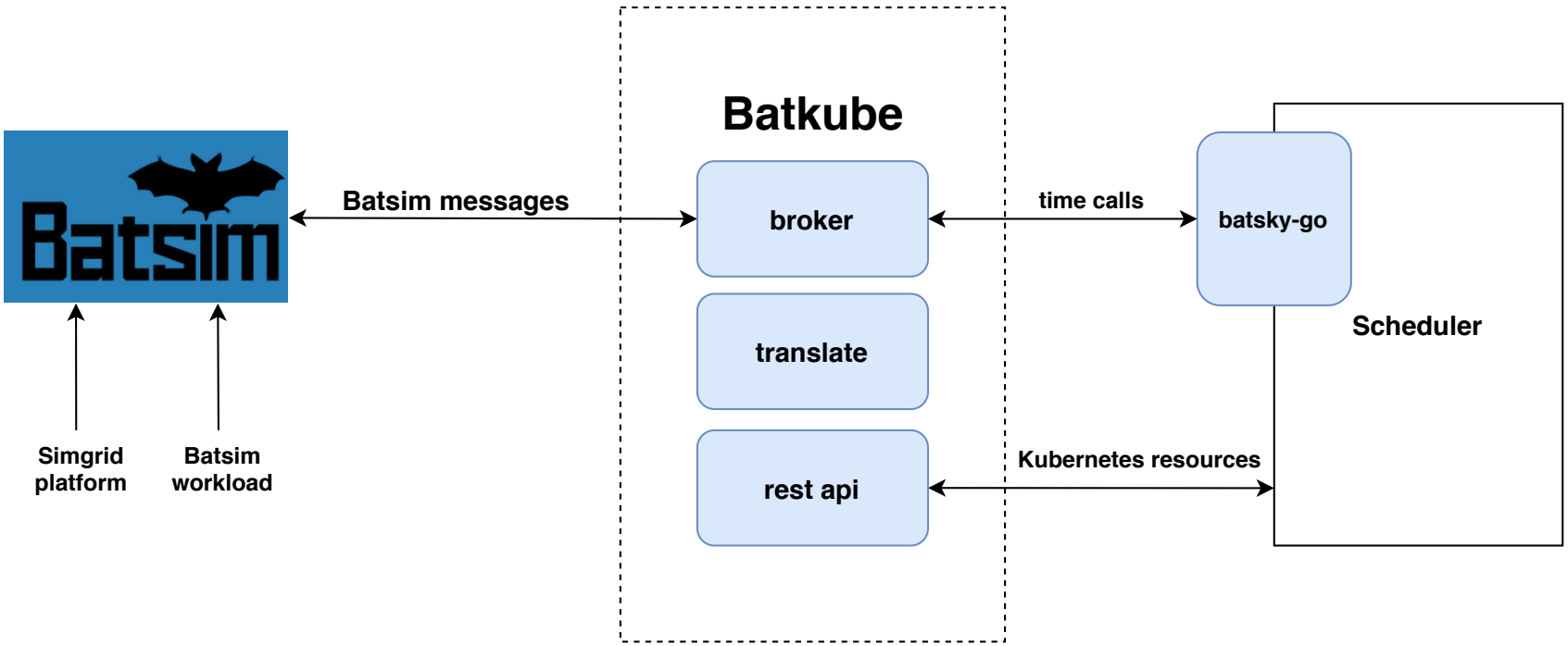
Kubernetes components.

Batkube integration with Kubernetes



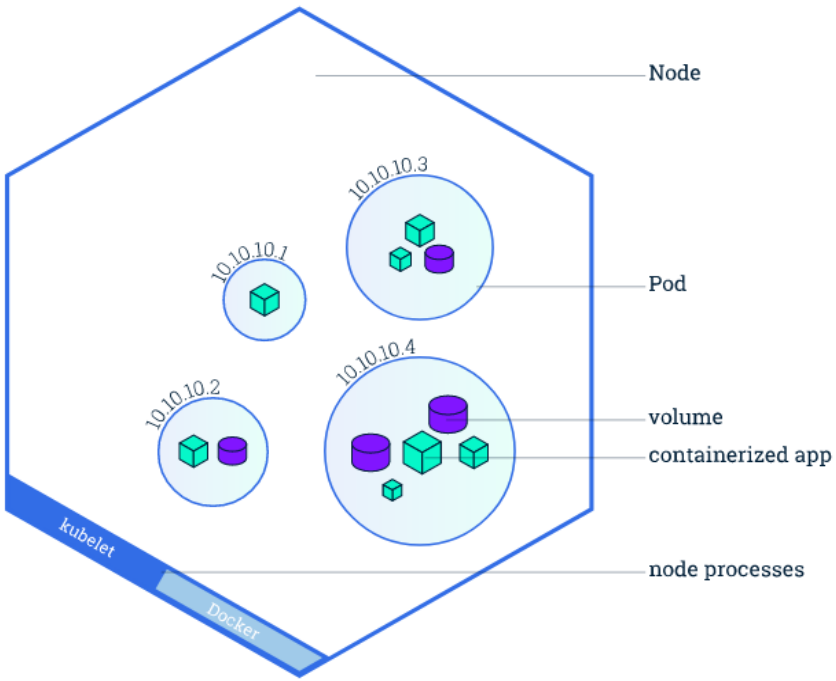
Reimplementation of a custom API.

Architeture of Batkube



Global architecture of Batkube.

Similar resources

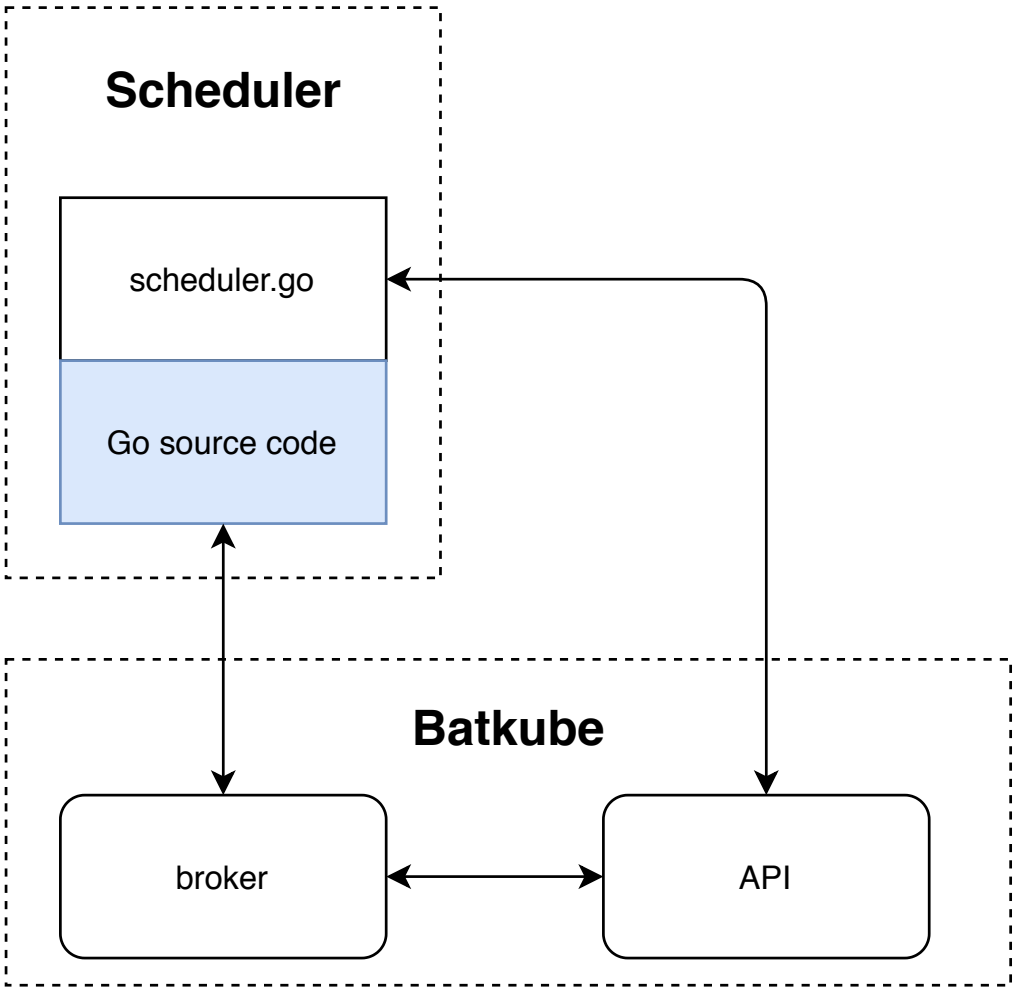


source: `https://kubernetes.io/docs/tutorials/kubernetes-basics/explore/explore-intro/`

Translation between Kubernetes and Batsim

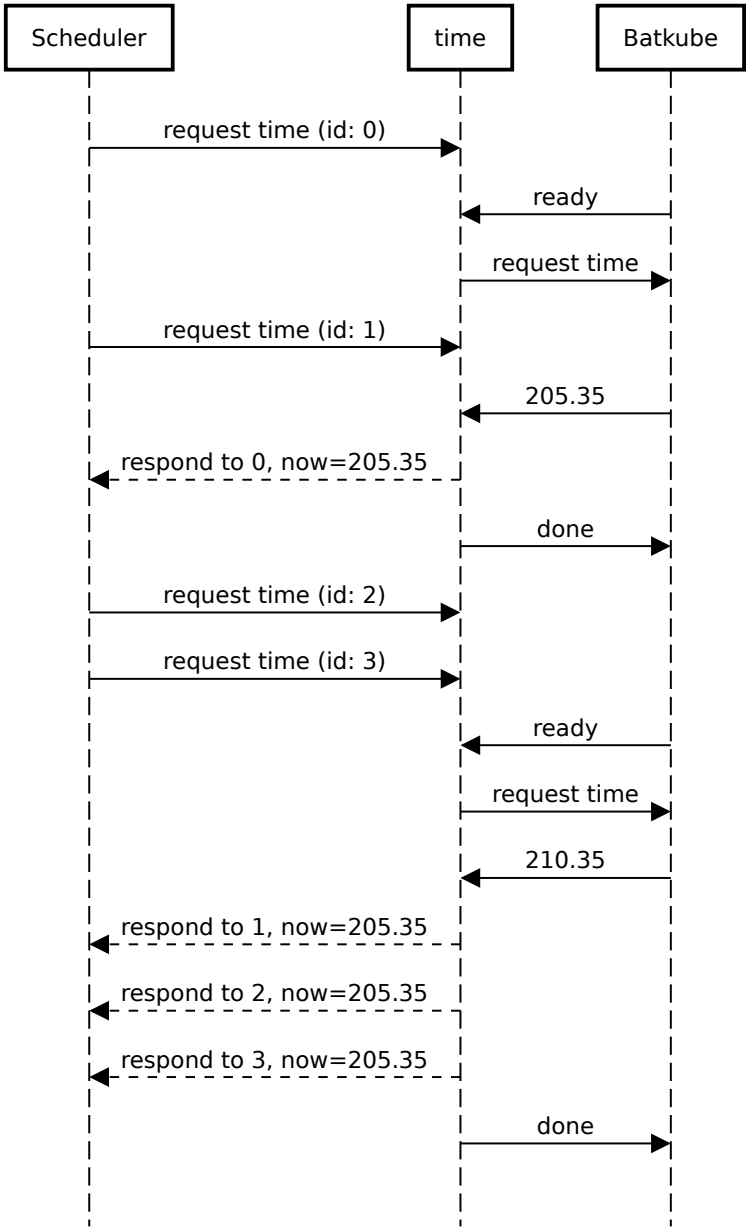
- A Pod = a job.
- A Node = a compute resource.

Time interception



Schedulers are patched to redirect their time.

batsky-go

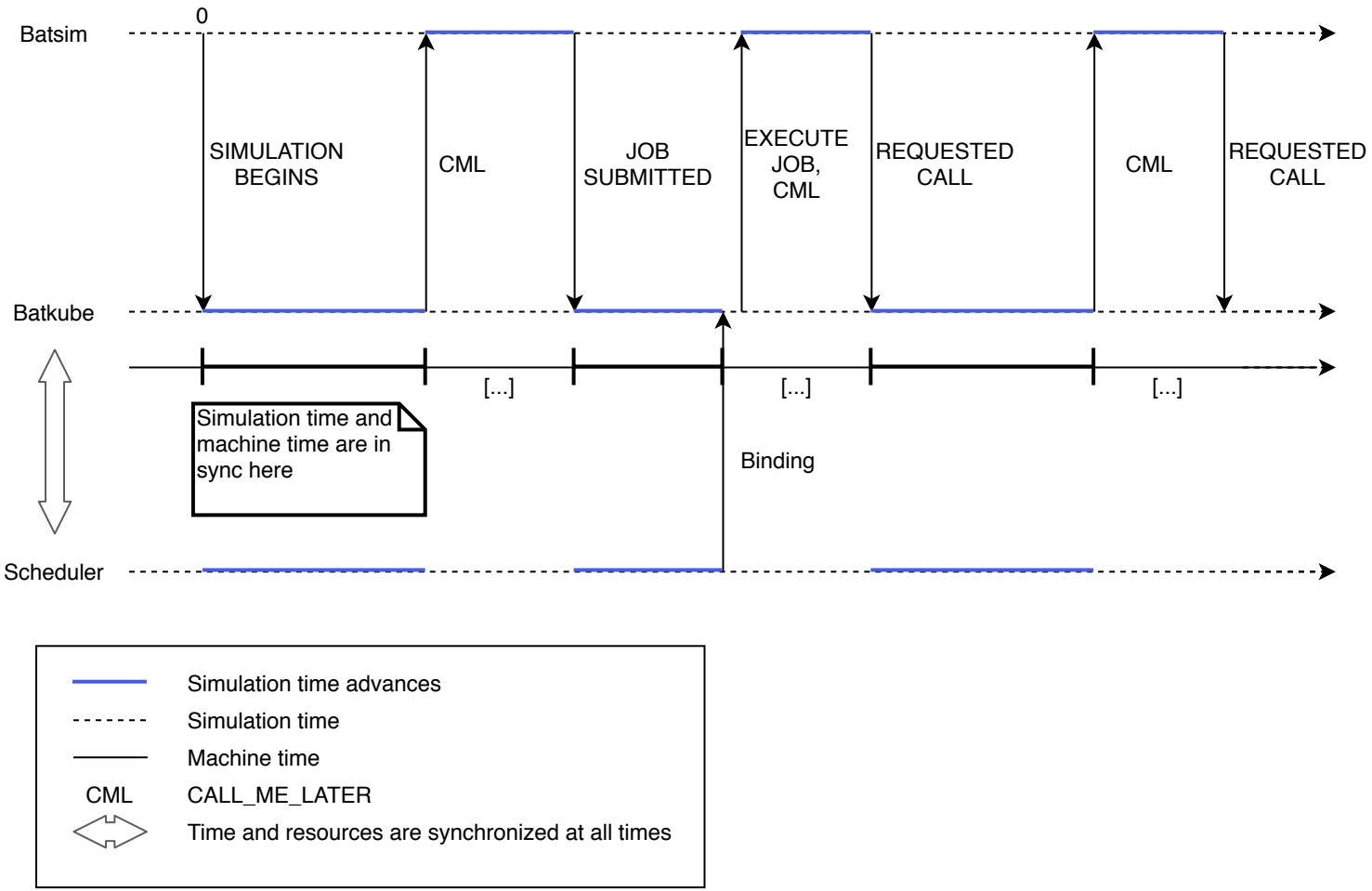


Exchanges between the scheduler, batsky-go (“time”) and Batsim

Time synchronization I

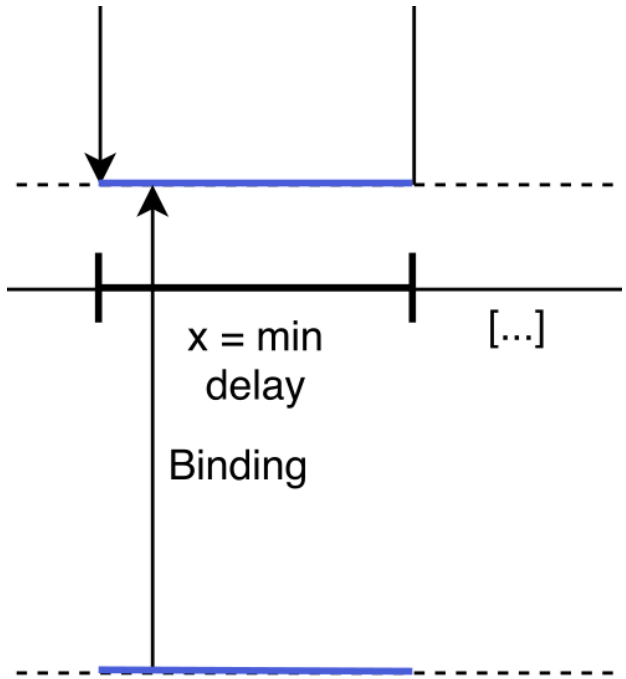
TODO: explain CML

Time synchronization II



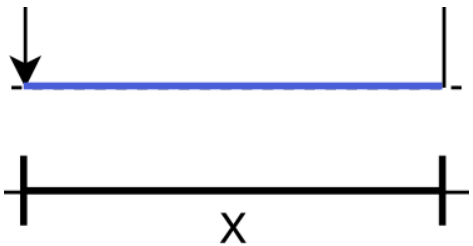
Time synchronization between Batsim and the scheduler

Parameters of the synchronization



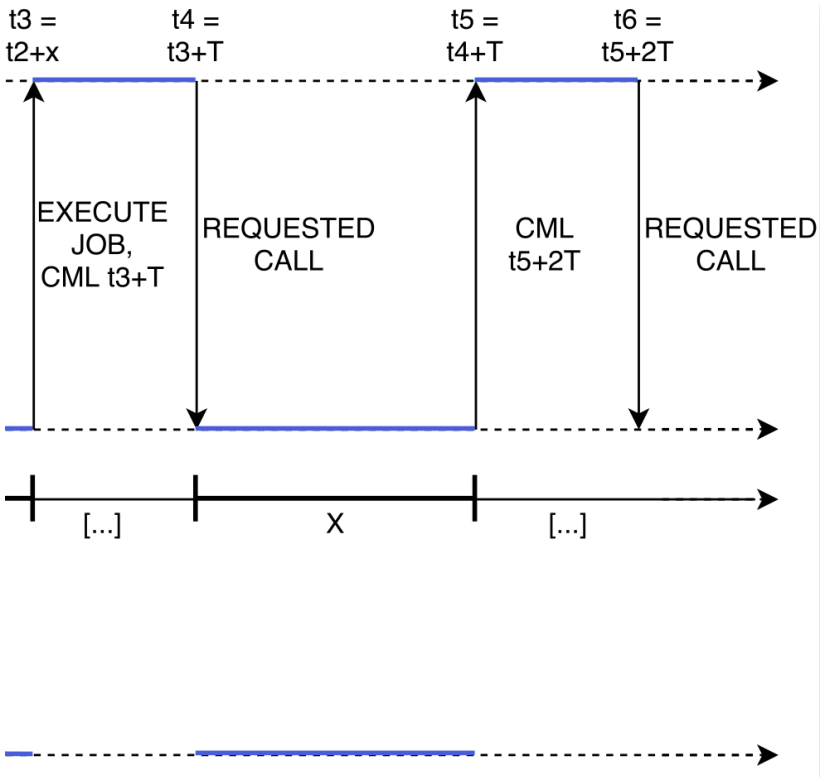
Minimum delay

Parameters of the synchronization



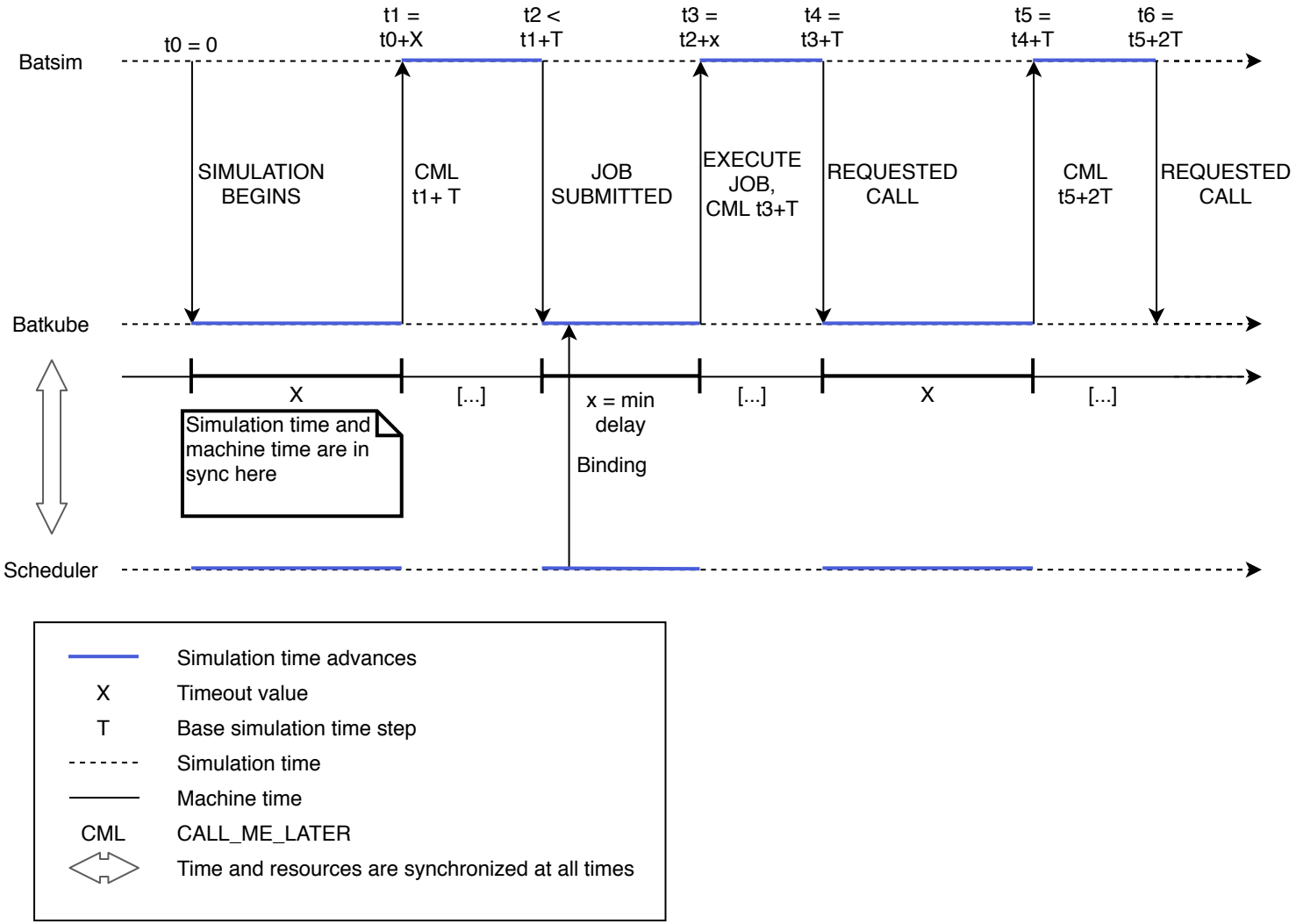
Timeout value

Parameters of the synchronization



Simulation time step $\in$ [`base-simulation-timestep`, `max-simulation-timestep`]

Time synchronization breakdown



Time synchronization between Batsim and the scheduler

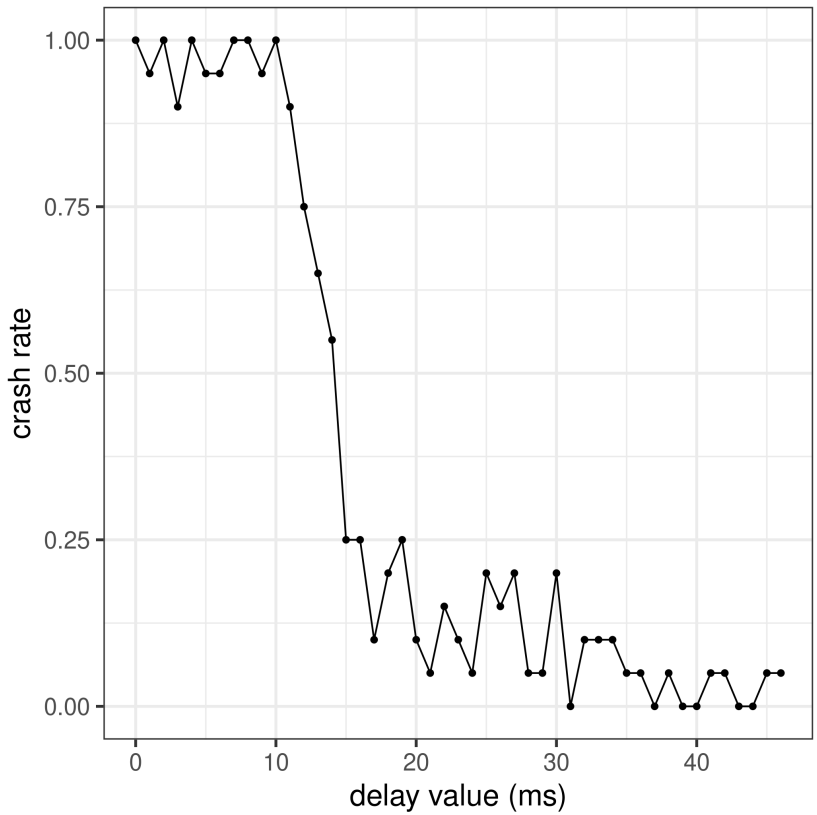
Study of the simulator

Studied workloads and platforms

TODO

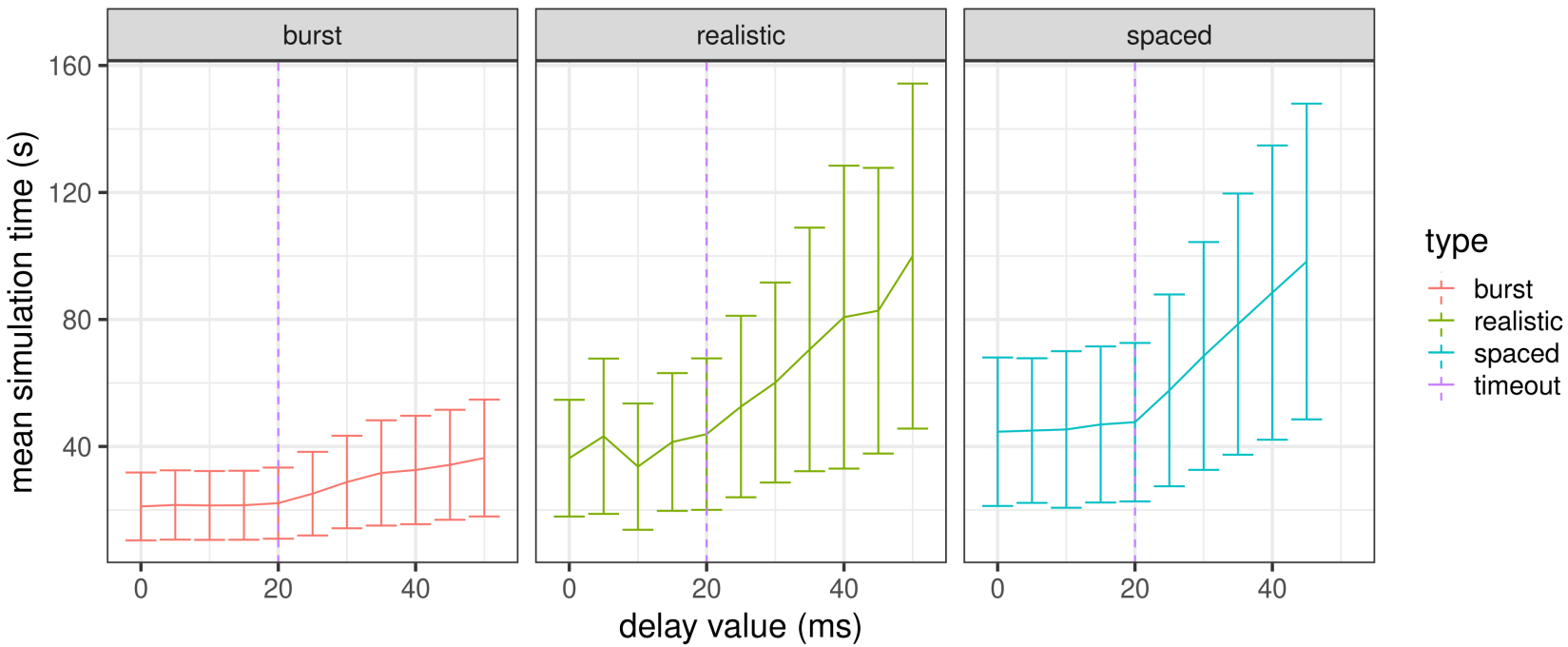
name	number of jobs	jobs type	sub time	resour
burst	200	delay (170s)	at time zero	
spaced	200	delay (170s)	every ten seconds	

Minimum delay I

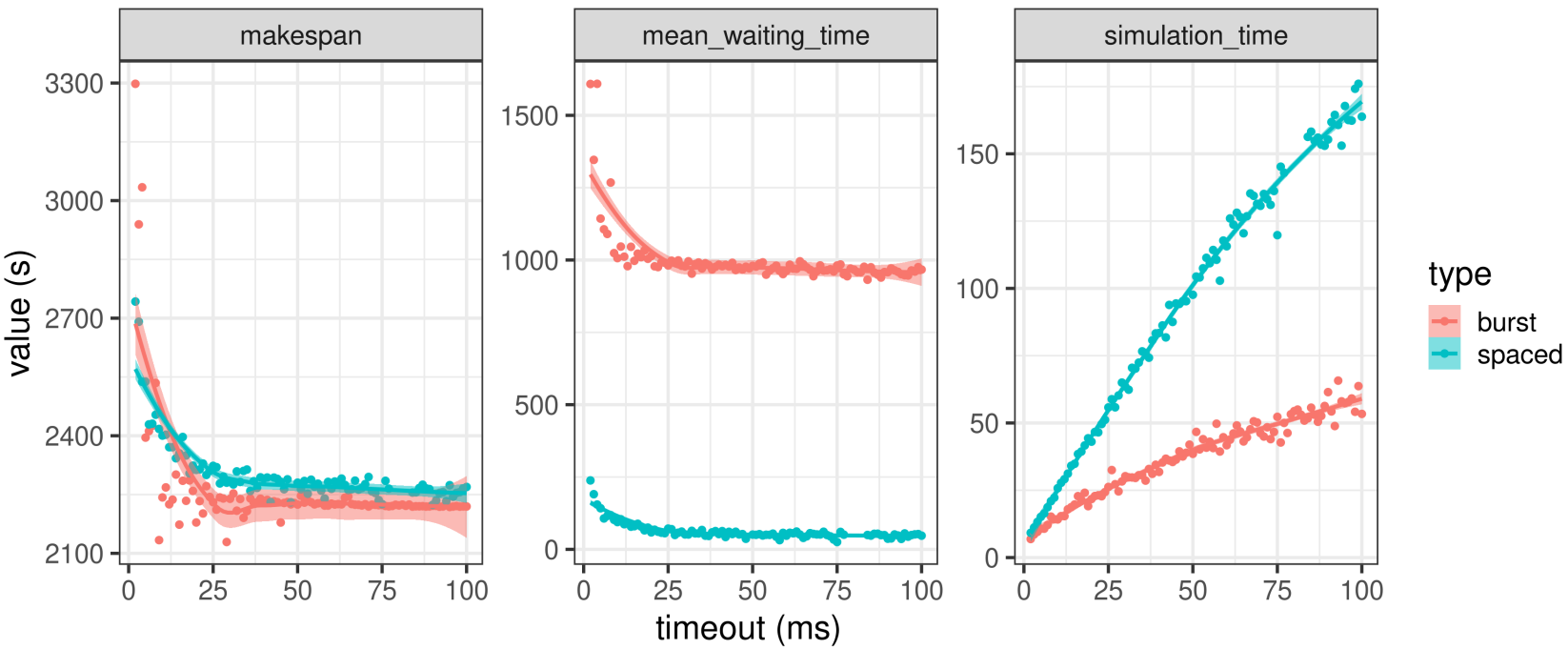


Note: inclure ce graphe?

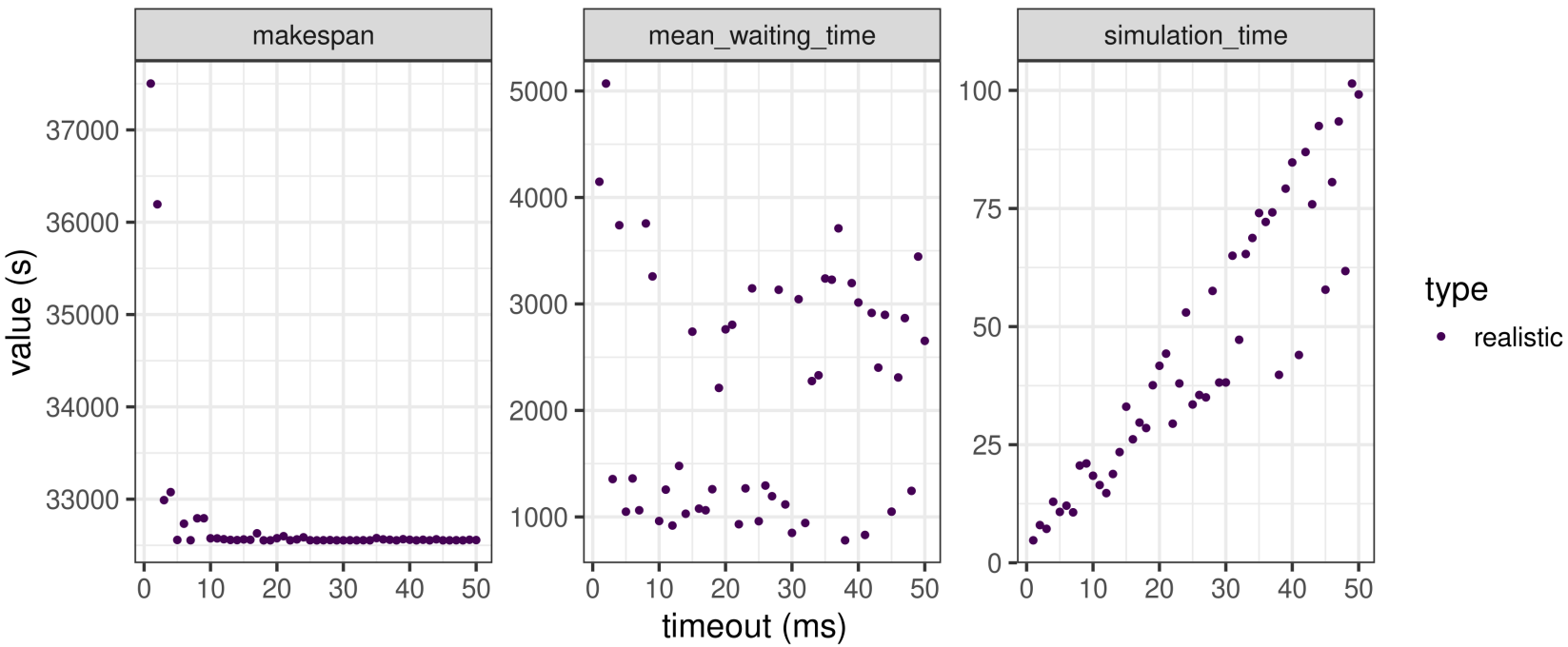
Minimum delay II



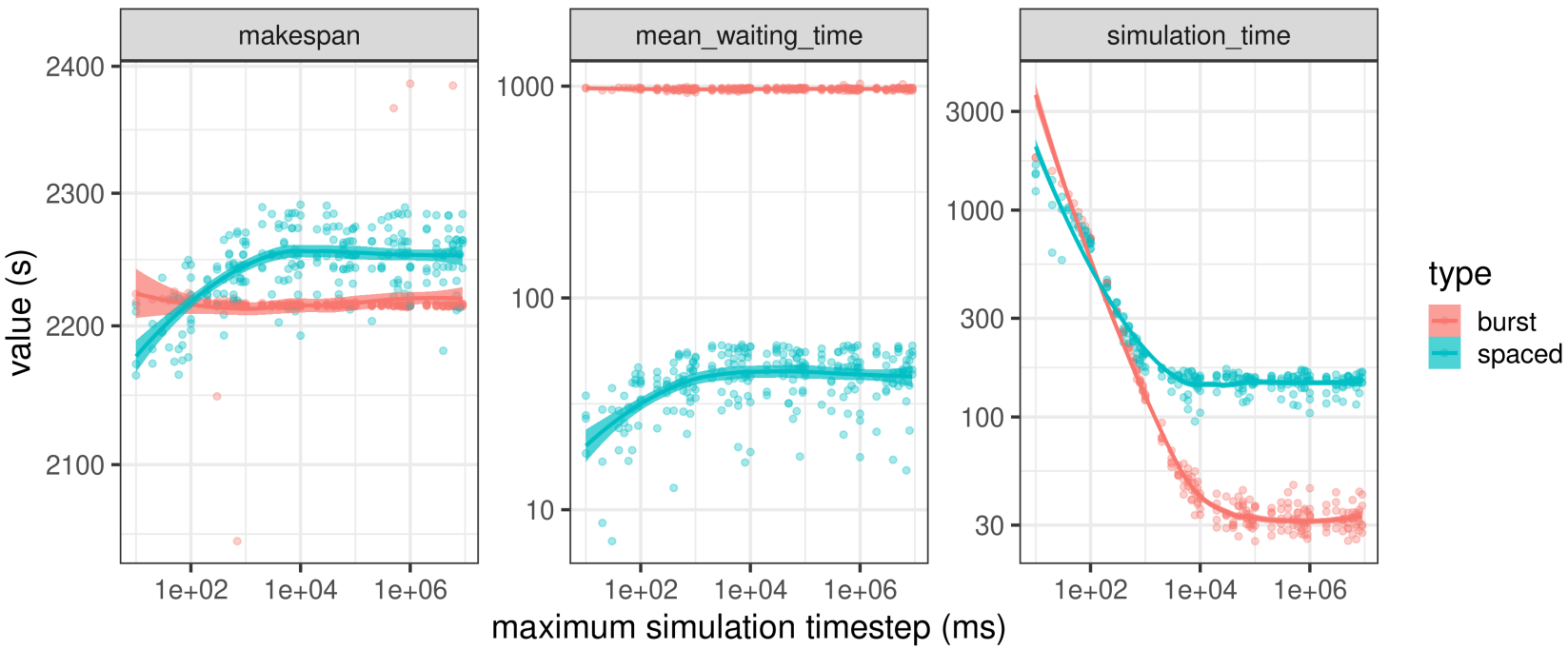
Timeout I


makespan
mean_waiting_time
simulation_time
value (s)
timeout (ms)
type
burst
spaced

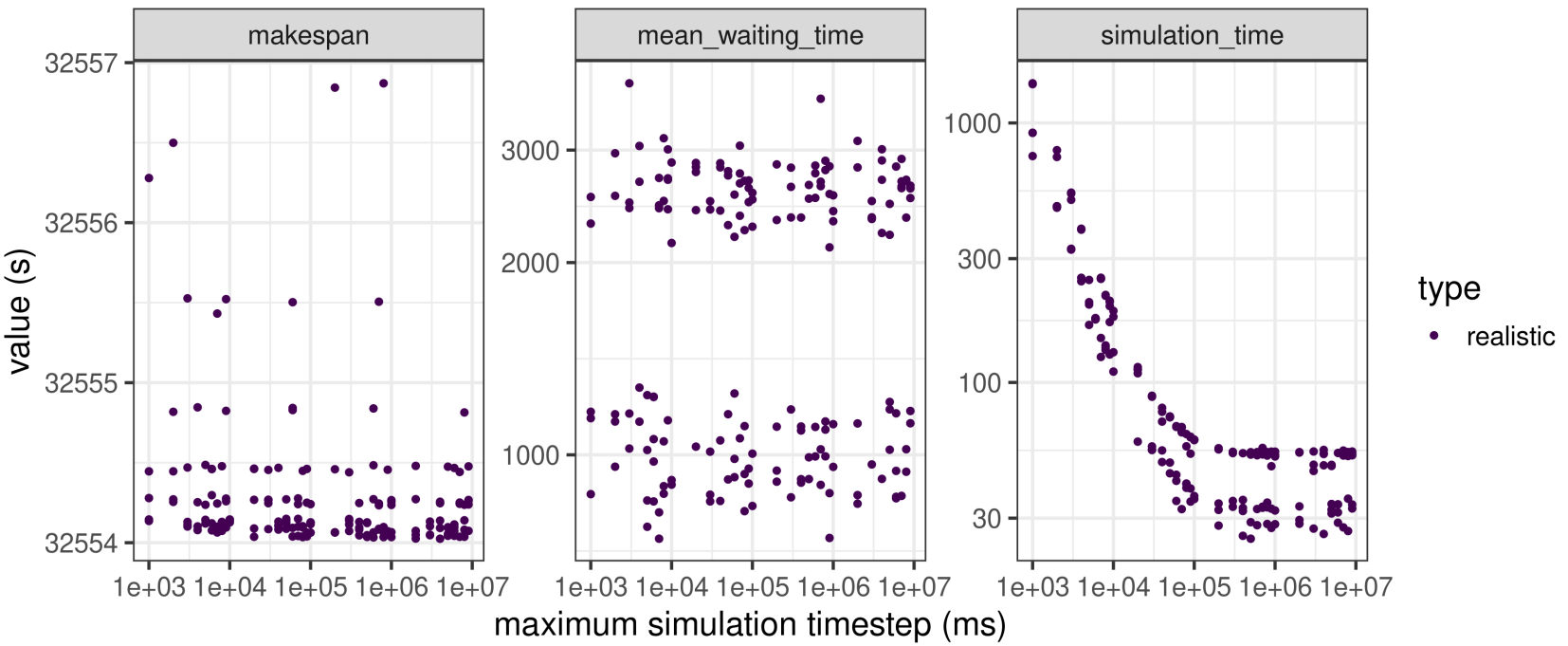
Timeout II


makespan
mean_waiting_time
simulation_time
value (s)
timeout (ms)
type
realistic

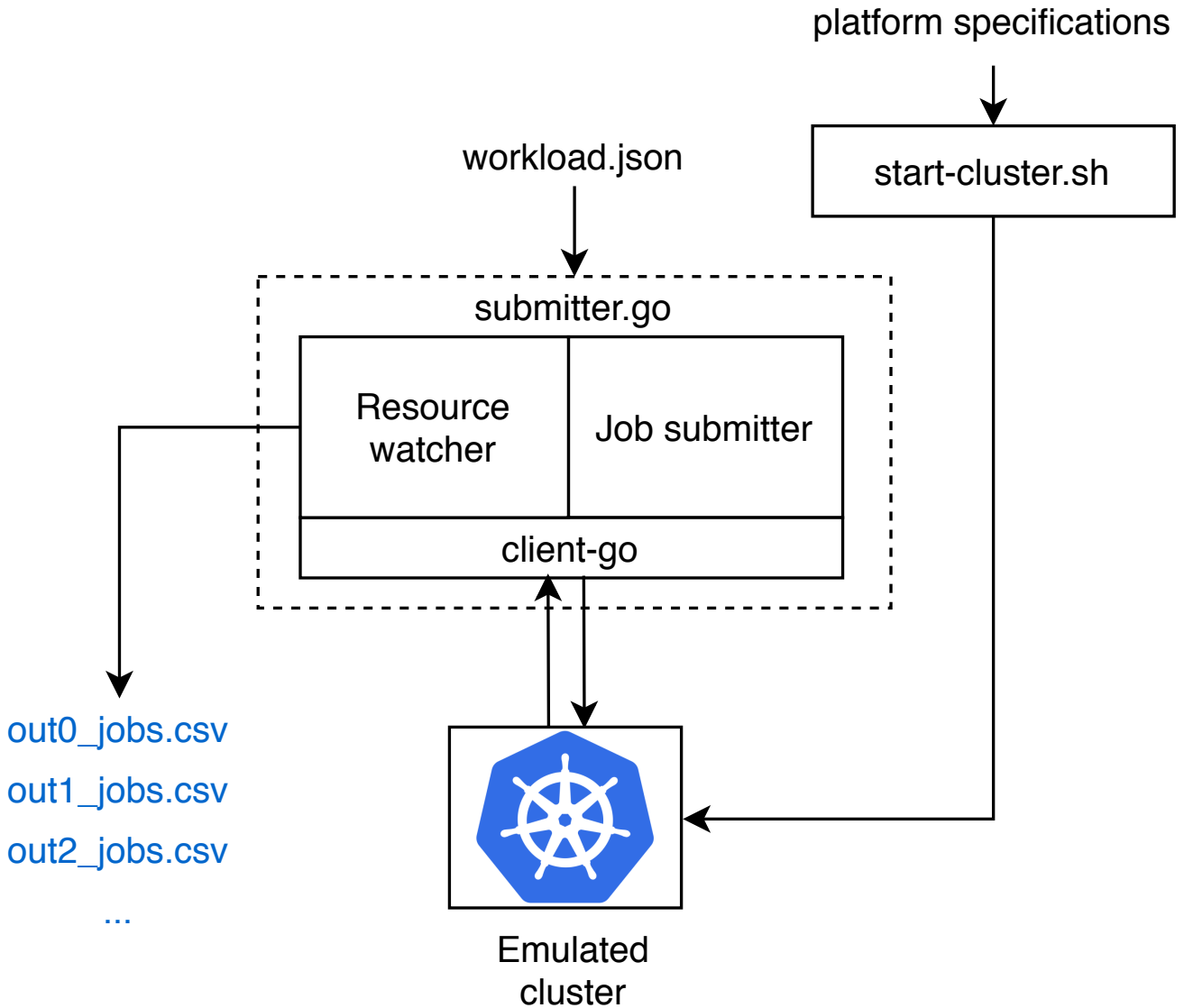
Maximum simulation timestep I



Maximum simulation timestep II



Experimentation on a real cluster



Deviation with reality

workload	**makespan**				**mean waiting time**			
	emulated		simulated		emulated		simulated	
	μ	σ	μ	σ	μ	σ	μ	σ
burst	2467	28.3	2215 (-252)	0.508	1077	10.6	970 (-107)	12.6
spaced	2468	5.14	2257 (-211)	16.9	146	1.67	48.1 (-97.9)	9.44
realistic	32556	-	32555 (-1)	1.30	2884	-	2020 (-864)	950

Discussion and future work

Capabilities of Batkube

- delay jobs
- cpu and memory requests
- can patch any kubernetes scheduler written in Go
- the api only supports the default scheduler

Limitations

- memory hungry (in fact, the scheduler is memory hungry)
- some problems with the scheduler
- not scalable

Perspectives for future work

- parallel jobs
- storage
- more complete api: support for more schedulers but also tools (monitoring tools)

References I