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

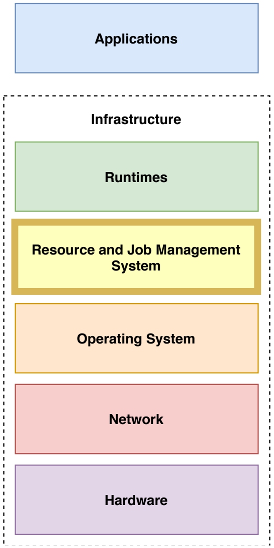
Introduction

Resource and Jobs Management System

The RJMS is at the core of the cluster.

Examples of RJMS

- OAR
- SLURM
- HadoopYARN
- Apache Mesos

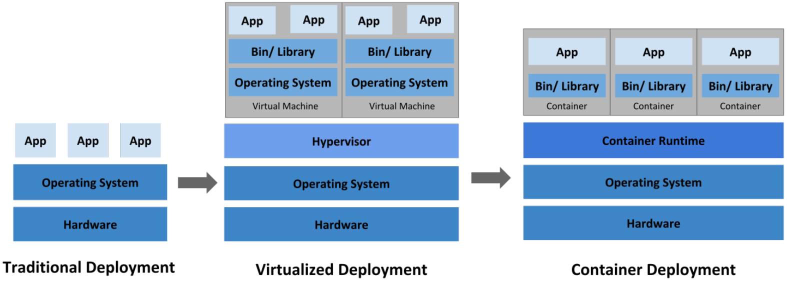


Kubernetes

Kubernetes in a nutshell

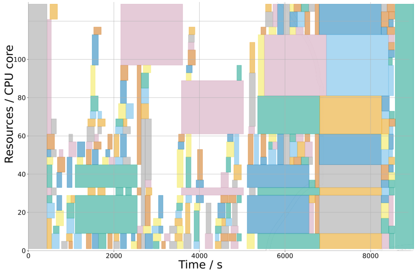
- Open source resource manager for containerized applications
- About 2M lines of code
- 2.8k contributors

kubernetes



source: `https://kubernetes.io/docs/`

A component of the RJMS: the scheduler



Scheduling is the act of allocating tasks to resources.

Numerous factors

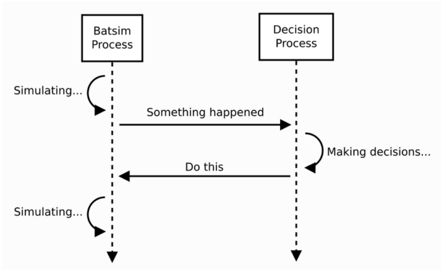
- Workloads
- Applications
- System size
- Network topology
- Energy consumption
- Scheduling policies

Complex implementations: Kubernetes default scheduler weighs **47k lines of code**.

Studying schedulers

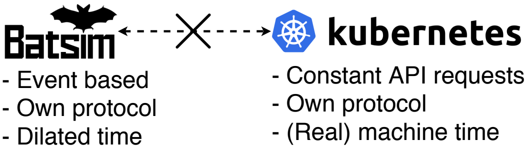
Different approaches

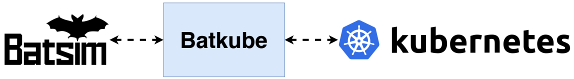
- Analytical study
- Real experiments
- Emulation
- **Simulation**



Batsim, an infrastructure simulator aimed at studying RJMS.

Contribution





Batkube supports

- Any Go scheduler
- Any cluster size
- Resource requests
- Non parallel tasks

Literature review

Infrastructure simulators I

TODO: what to do with this slide?

	Grid	HPC	Cloud	Peer to Peer	Volunteer Computing
SimGrid	✓				
GridSim	✓				
LogGOPSim		✓			
BigSim		✓			
CloudSim			✓		
GroudSim	✓		✓		
PeerSim				✓	
OverSim				✓	
SimBA					✓
SimBOINC					✓

Domain specific simulators

Infrastructure simulators II

Software specific simulators

- YARNSim
- SLURM simulator

Publication specific simulators

“Publish and perish” - Milian Poquet

Simulators for the study of RJMS

- Batsim
- Alea
- Accasim

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

Challenges to tackle

1. Integration with Kubernetes.

Technical challenges

Challenges to tackle

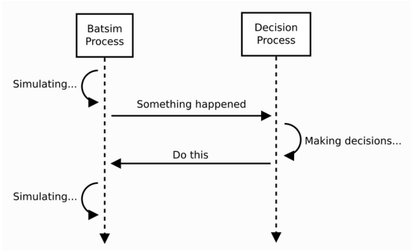
1. Integration with Kubernetes.
2. Intercepting scheduler time.

Technical challenges

Challenges to tackle

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

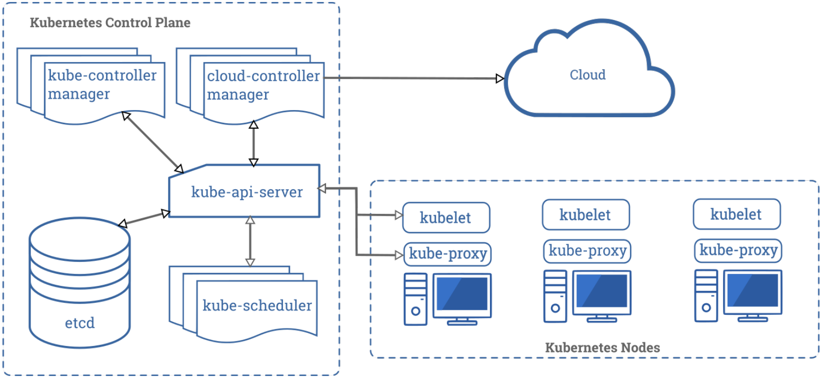
Batsim concepts



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

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

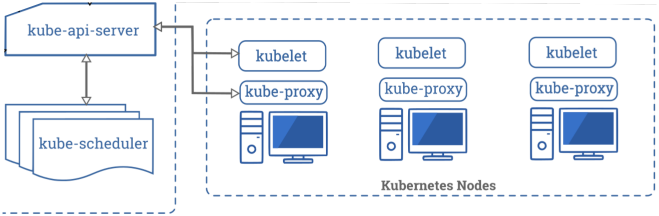
Kubernetes concepts



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

Kubernetes components.

Kubernetes concepts



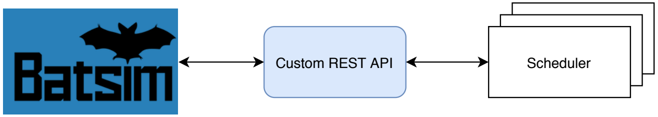
source: `https://kubernetes.io/docs/concepts/overview/components/`

Kubernetes components.

Different paradigms

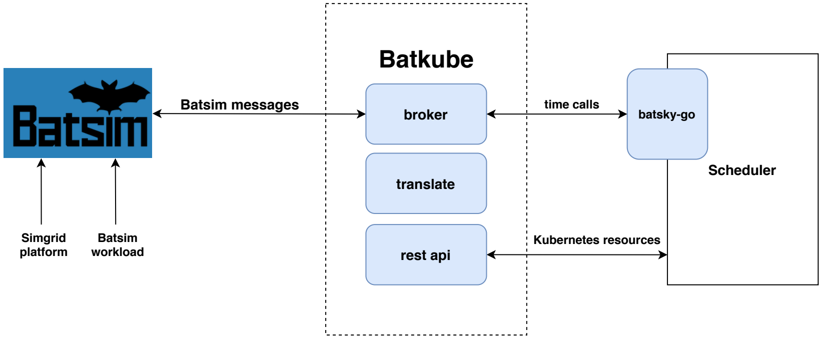
Batsim: event based, simulation time.
Kubernetes scheduler: asynchronous calls to the API, machine time.
The goal is to make the scheduler event based and relying on simulation time for Batsim, and make Batsim a kube-api-server to the scheduler.

Batkube integration with Kubernetes



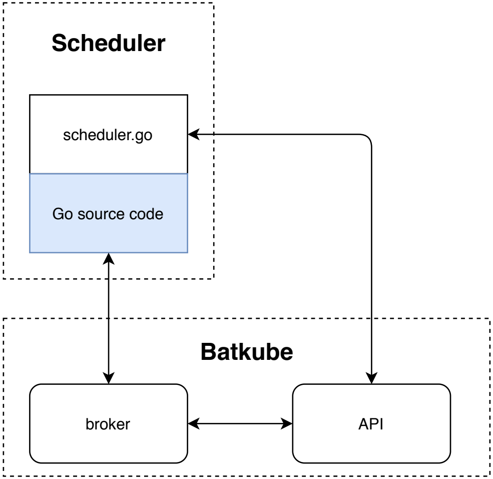
Reimplementation of a custom API.

Architeture of Batkube



Global architecture of Batkube.

Time interception

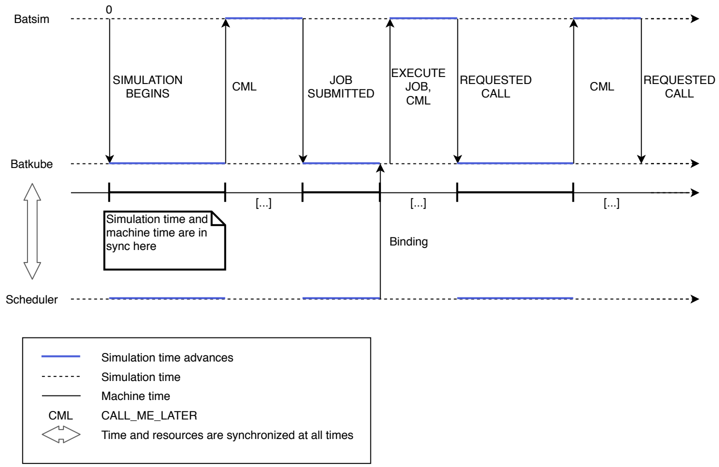


Schedulers are patched to redirect their time.

Time synchronization I

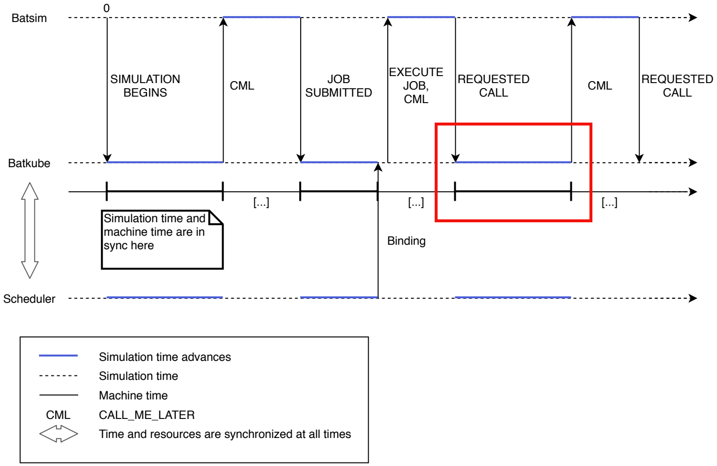
TODO: explain CML

Time synchronization II



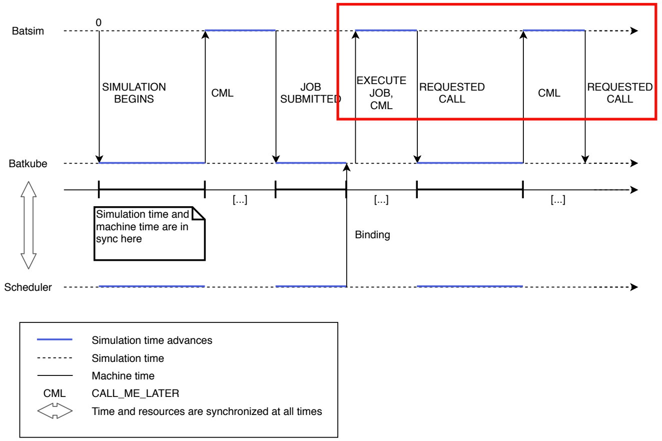
Time synchronization between Batsim and the scheduler

Parameters of the synchronization I



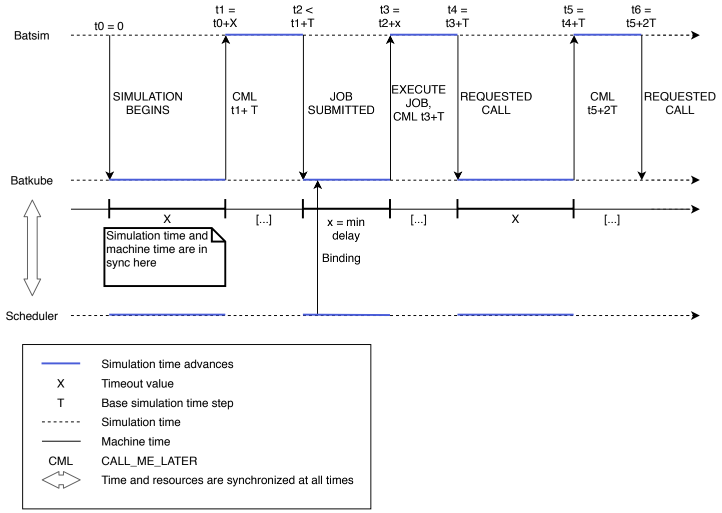
Timeout value

Parameters of the synchronization II



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

Time synchronization breakdown



—	Simulation time advances
X	Timeout value
T	Base simulation time step
- - - - - -	Simulation time
———	Machine time
CML	CALL_ME_LATER
⇔	Time and resources are synchronized at all times

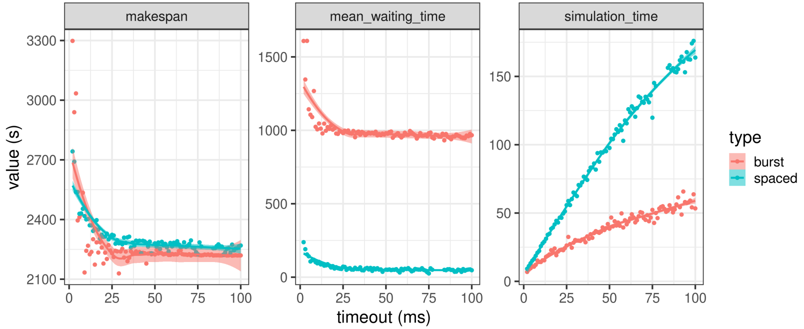
Time synchronization between Batsim and the scheduler

Study of the simulator

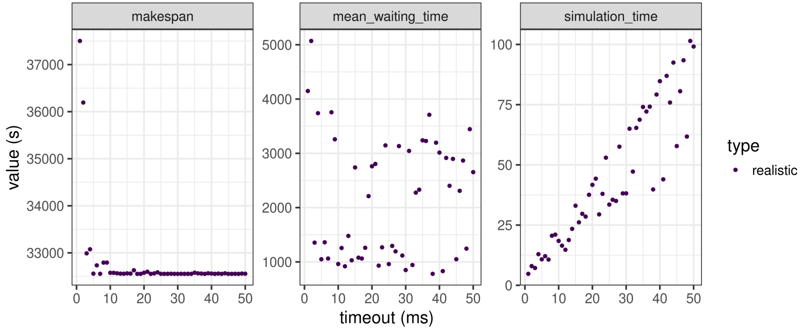
Experimental design

TODO: Scheduler used, platforms and workloads tested, what experiments (parameters, metrics studied, repetitions)

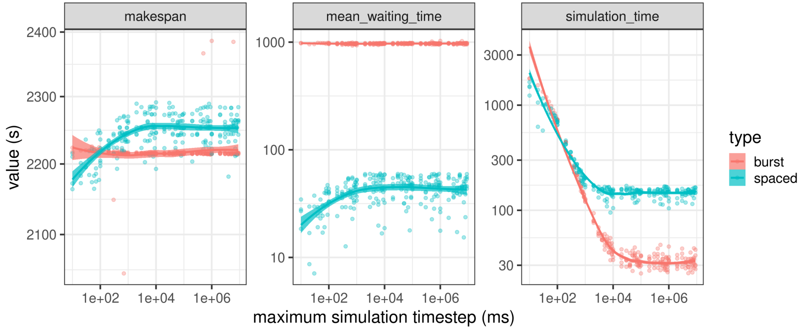
Timeout I


makespan
mean_waiting_time
simulation_time
value (s)
3300
3000
2700
2400
2100
1500
1000
500
0
150
100
50
0
0
25
50
75
100
timeout (ms)
type
burst
spaced

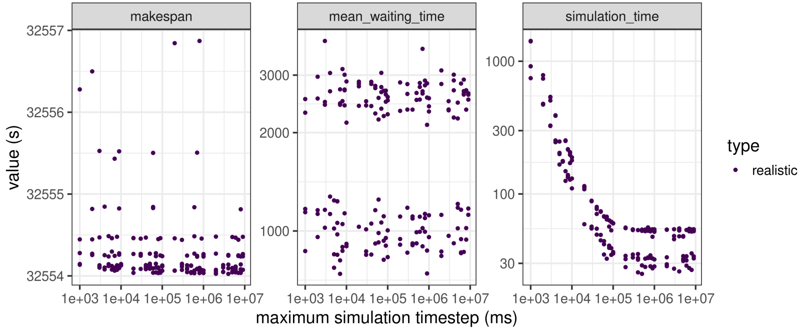
Timeout II


makespan
mean_waiting_time
simulation_time
value (s)
timeout (ms)
type
realistic
37000
36000
35000
34000
33000
5000
4000
3000
2000
1000
100
75
50
25
0
0
10
20
30
40
50

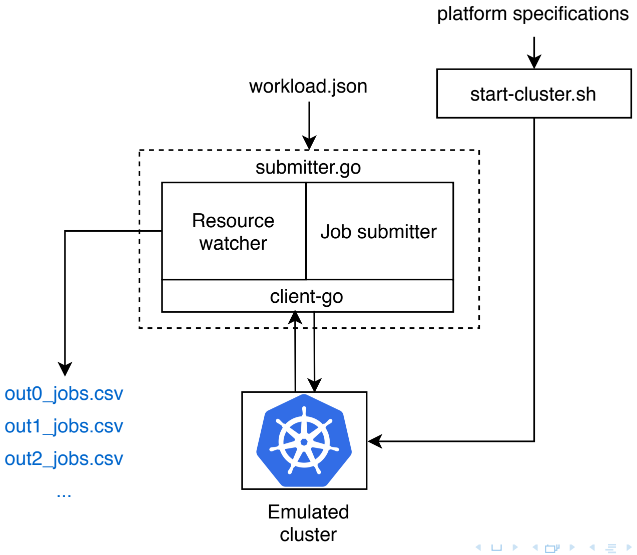
Maximum simulation timestep I



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

Conclusion

Deviation with reality: can be fixed with some work on the api. Need experiments to measur and quantify this deviation.
max timestep: studying max timestep alone is not enough, need to study it with backoff multiplier.
base time step: need an experiment on it. Too much importance was credited to max timestep, the base timestep might have importance.

Discussion and future work

Capabilities and limitations of Batkube

WIP

Capabilities

- 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