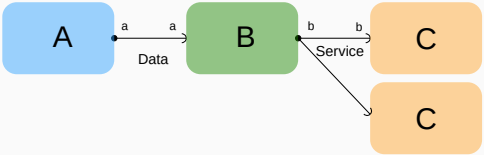
Efficient and Safe Distributed Software Commissioning and Reconfiguration

Hélène Coullon, STACK reasearch group, Assistant professor IMT Atlantique, Inria chair
Maverick Chardet, Christian Perez, Dimitri Pertin (Madeus, Concerto)
Claude Jard, Didier Lime (MADA)
Simon Robillard, Charlène Servantie (VeRDi)

2019-12-09 Tromsø

Distributed software



- Modules, services or components
  - running on distributed machines interconnected by a network
- Connections, communications
  - cooperating to get a result
- Examples: micro-services- and service-oriented software systems, MPI applications, CORBA applications, systems of systems etc.

Component management

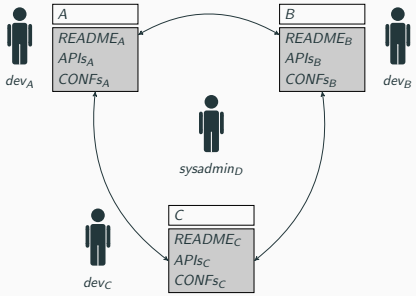
Code (development) ✓

- already existing code and communications

DevOps (management) ✗

- Control APIs on the component (start, backup etc.)
- Configuration files to get infra parameters (sysadmins)
- README file (or webpage)
  - uses both control APIs and configuration files
  - description of procedures (install, stop, update etc.)
  - commissioning procedure: requirements (libraries, packages etc.), configurations, order of execution, testing the commissioning sucess

Distributed software commissioning and reconfiguration



- dev$_A$, dev$_B$ and dev$_C$ write the APIs, the configuration files and the README for their components
- sysadmin$_D$ has to
  - coordinate the READMEs of all components
  - webpages/scripts to explain this kind of complex commissionings (example)

Challenges

- Languages and models for commissioning and reconfiguration
  - Software Engineering (SE) properties: composition, code-reuse, separation of concerns etc.
- Safety of commissionings and reconfigurations
- Efficiency of commissionings and reconfigurations

Limitations of DevOps tools

- no formal model, no guarantees
- limited composition and separation of concerns
- limited efficiency

Limitations of academic related work

- limited separation of concerns
- limited efficiency

Contributions

1. Madeus: formal model to adress the efficient coordination of commissioning procedures
   - *Maverick Chardet, Hélène Coullon, Christian Perez and Dimitri Pertin. Madeus: A formal deployment model. In 4PAD 2018 (hosted at HPCS 2018), Jul. 2018, Orléans, France.*
   - Journal under submission
2. MADA: Study the use of model checking to help in the design of safe and efficient distributed software commissioning with Madeus
   - *Hélène Coullon, Claude Jard and Didier Lime. Integrated Model-checking for the Design of Safe and Efficient Distributed Software Commissioning. In iFM 2019, Dec. 2019, Bergen, Norway.*
3. Concerto: A generalization of Madeus to (re)configuration of distributed software systems
   - Under submission

Table of contents

Distributed software commissioning: Madeus

Madeus principles

- separation of concerns: $\text{dev}_{\text{assembly}}$ does not need to know details about each component
- more parallelism in the commissioning language
- automatic coordination, correct order of execution, automatic parallelism

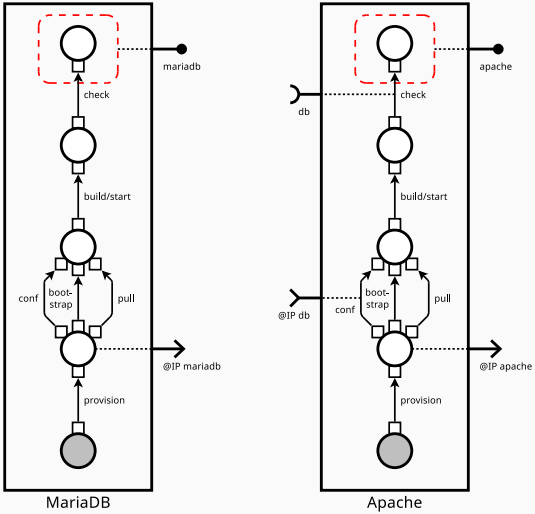
Why efficiency in distributed software commissioning?

- scripts executed many times by sysadmins
- continuous integration of big projects
- first step toward efficient reconfiguration

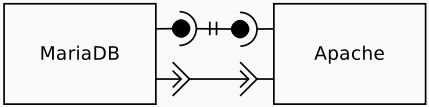
Madeus contributions

- a formal model and its operational semantics
- a theoretical performance model
- an open-source prototype (`https://gitlab.inria.fr/VeRDi-project/concerto/tree/master/concerto/madeus`)
- an evaluation on a real use-case OpenStack
  - reproducible/repeatable research
  - `https://gitlab.inria.fr/VeRDi-project/madeus-openstack`
  - `https://gitlab.inria.fr/VeRDi-project/madeus-openstack-benchmarks`

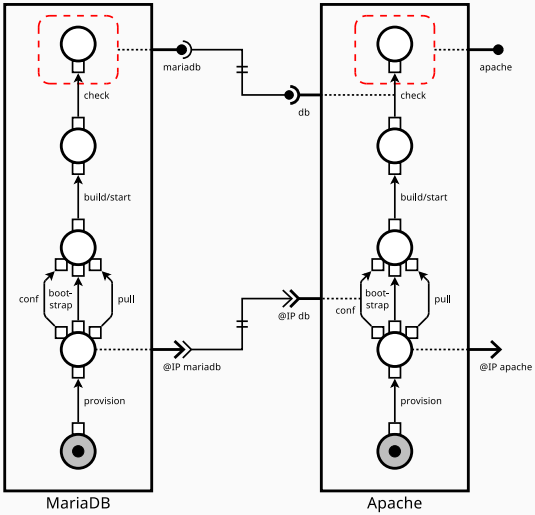
Madeus - control component (1/2)



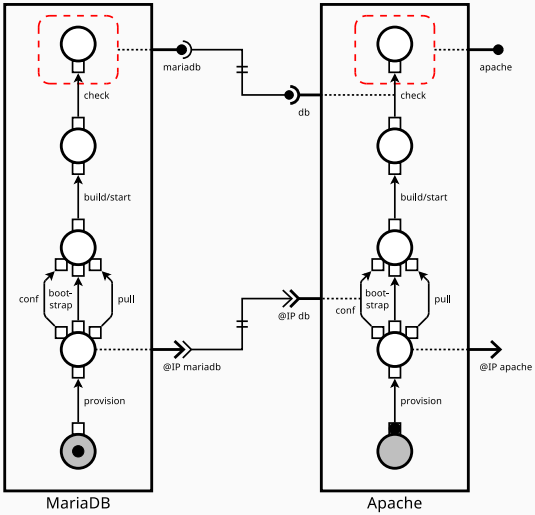
Madeus - composition and assembly (2/2)



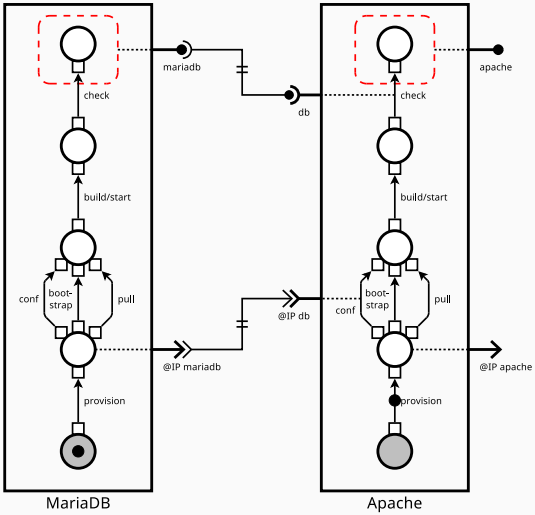
Madeus execution



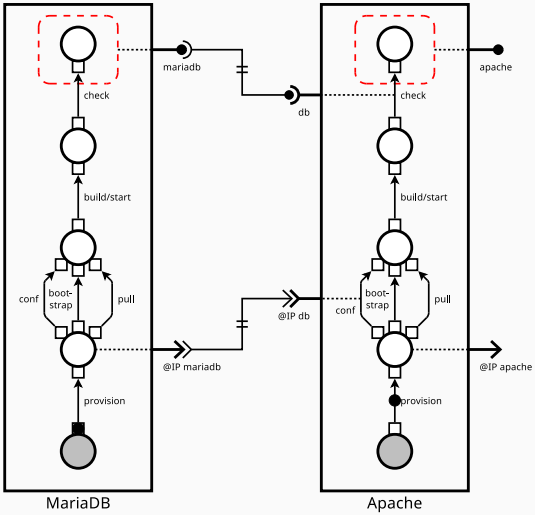
Madeus execution



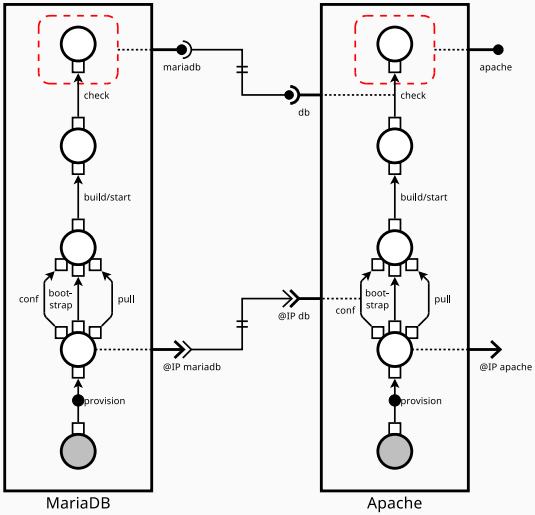
Madeus execution



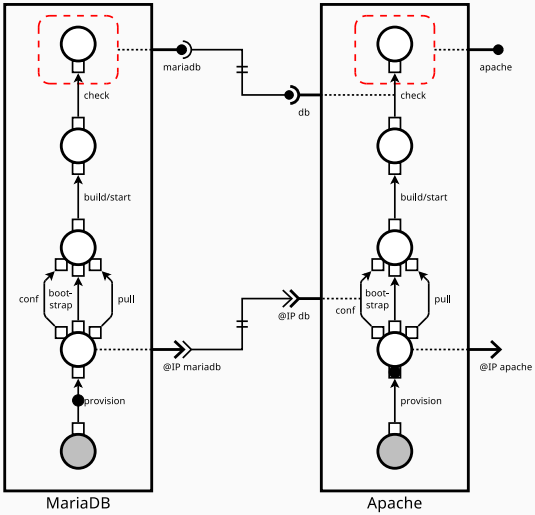
Madeus execution



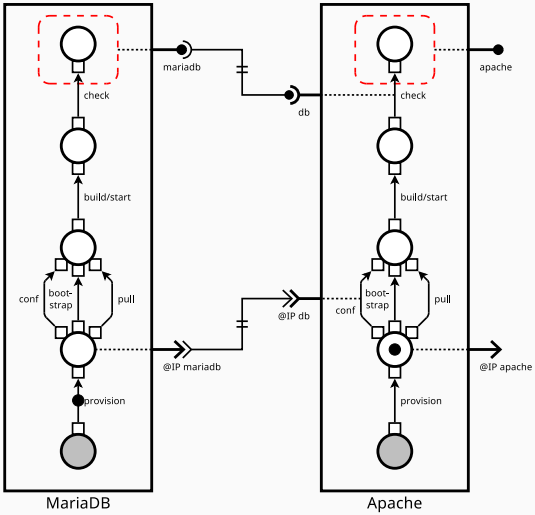
Madeus execution



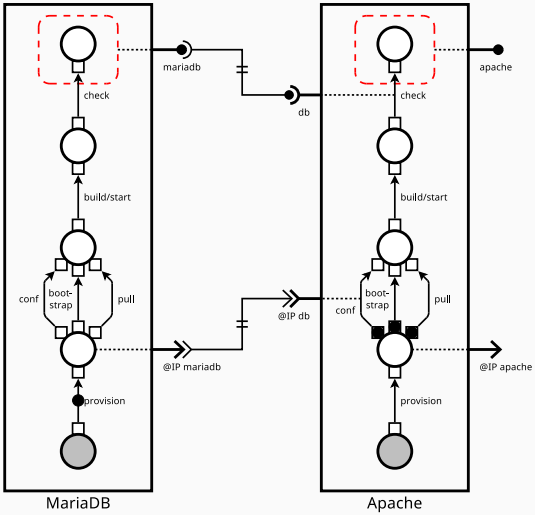
Madeus execution



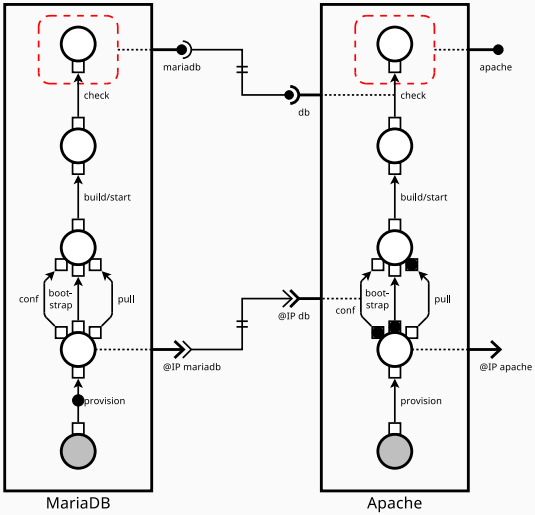
Madeus execution



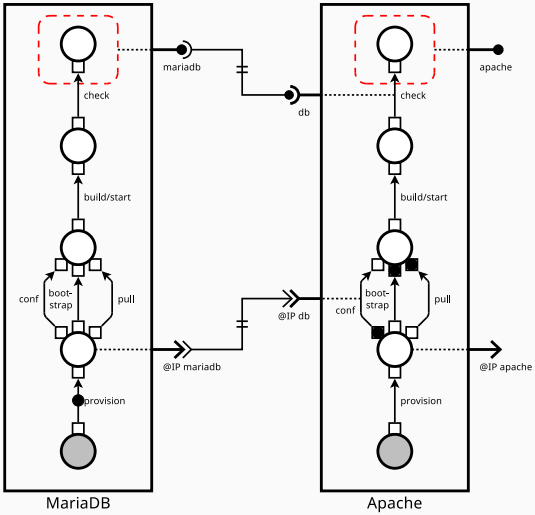
Madeus execution



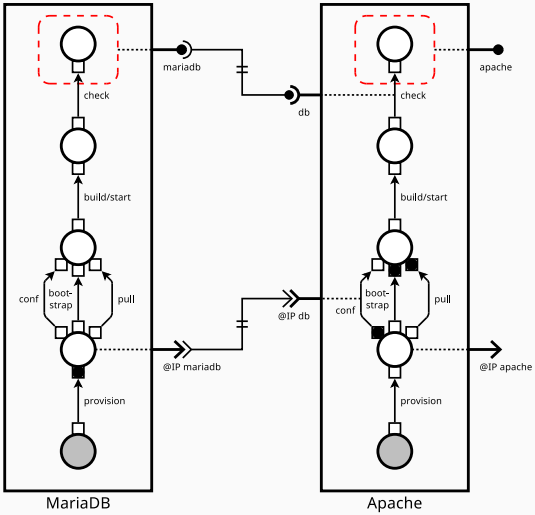
Madeus execution



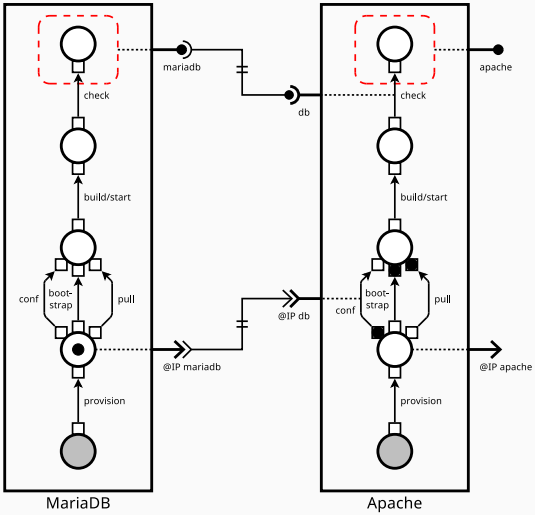
Madeus execution



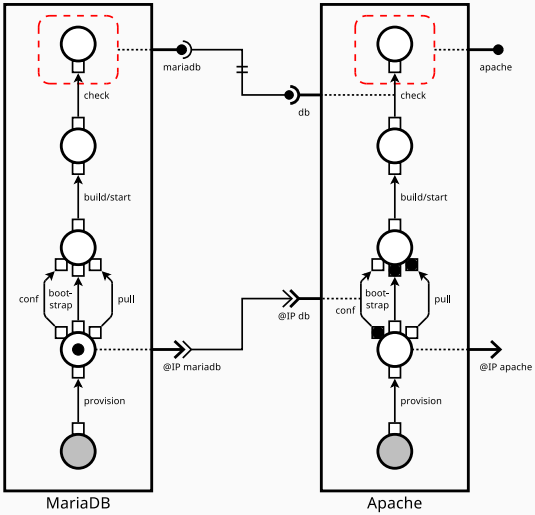
Madeus execution



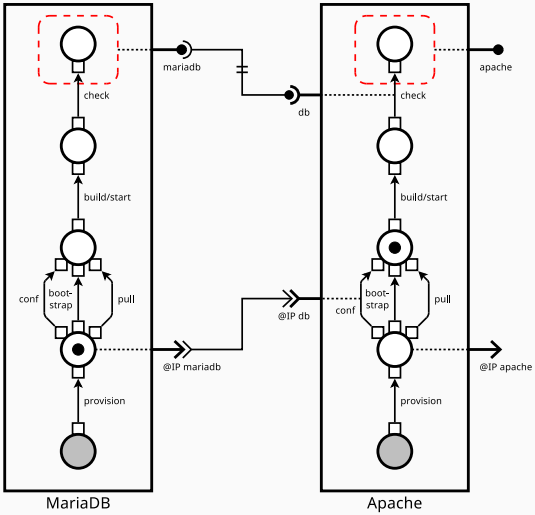
Madeus execution



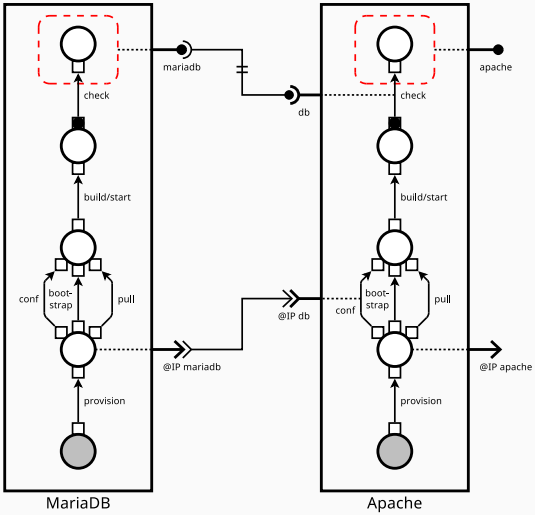
Madeus execution



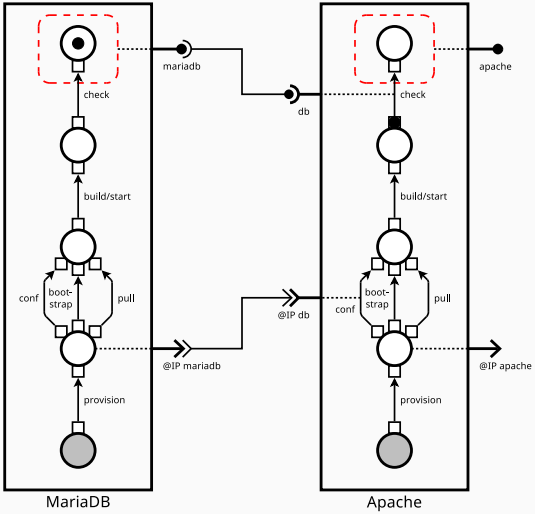
Madeus execution



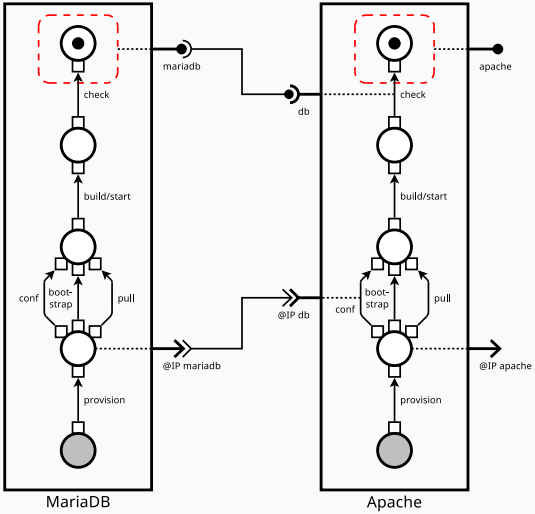
Madeus execution



Madeus execution

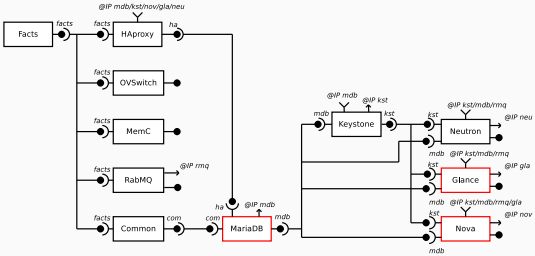


Madeus execution



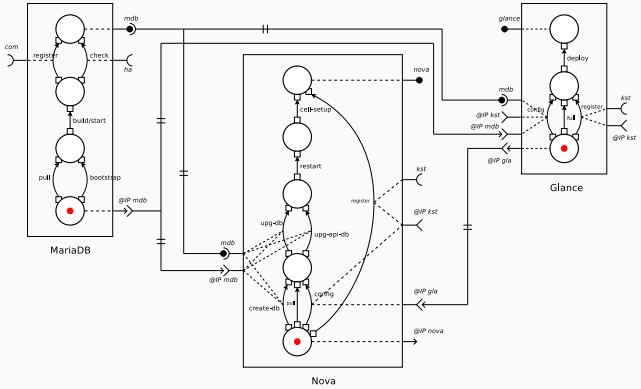
Experiments (1/4)

- "OpenStack is a cloud operating system that controls large pools of compute, storage, and networking resources throughout a datacenter"
- Kolla-ansible commissioning of OpenSatck (36 services gathered in 11 components, deployed on three nodes).



Experiments (2/4)

	Places	Trans.	Ports
Facts	2	1	1
Common	3	2	2
HaProxy	2	1	7
MemCached	2	1	2
MariaDB	4	5	4
RabbitMQ	2	1	3
Keystone	3	2	4
Glance	3	4	7
Nova	5	8	8
OpenVSwitch	3	1	2
Neutron	3	4	7
Total	32	30	47



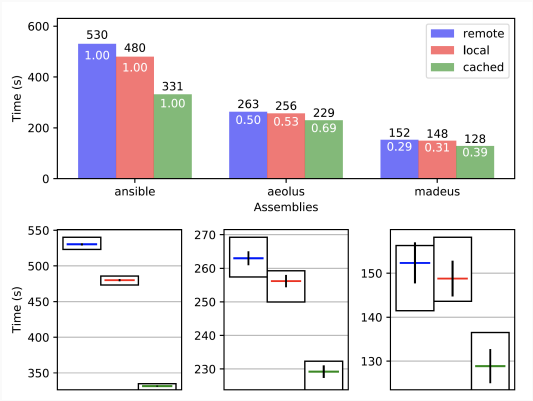
Experiments (3/4)

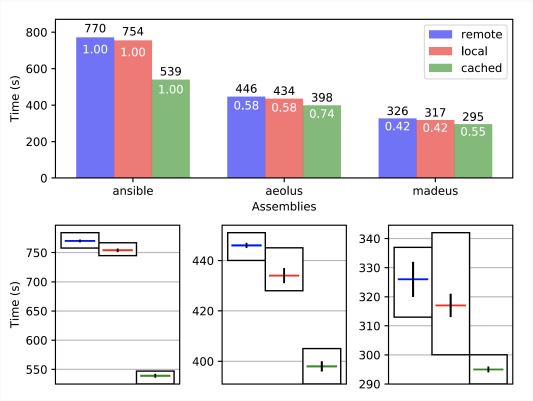
- Kolla-Ansible = Docker + Ansible

	Compute	Network	Control
# images	9	11	16
Total size (MB)	2767	2705	4916

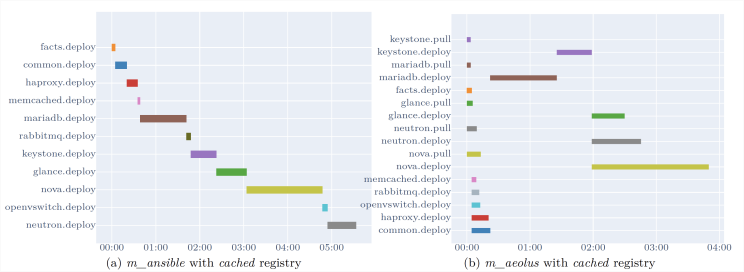
- 3 types of Docker registries: remote, local, cached
- benchmarks on two different clusters of Grid'5000: ecotype (Nantes), nova (Lyon)

results - performance (1/2)



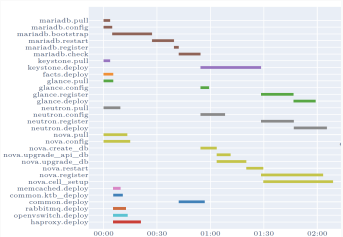


results - tasks analysis (2/2)



(a) *m_ansible* with *cached* registry

(b) *m_aeolus* with *cached* registry



(c) *madeus* with *cached* registry

Distributed software reconfiguration: Concerto

Why reconfiguration?

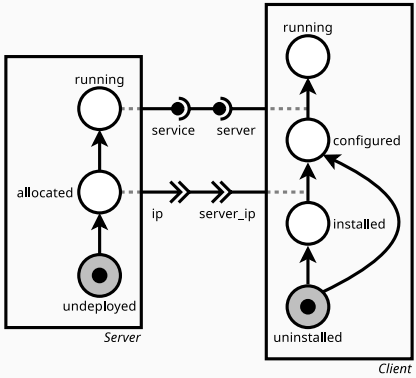
- Resources, placement and network changes
  - optimization
  - mobility
  - faults
- Software topology changes
  - optimization
  - energy, security, sensors etc.
  - external events
- Software update
- etc.

State of the Art

Autonomic reconfiguration: MAPE-K as a general model

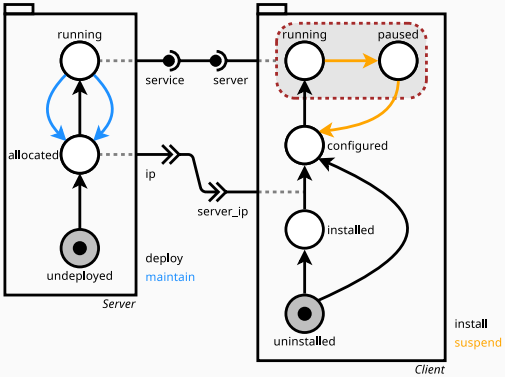
- (M) monitoring (monitor the software system and its environment)
- (A) analysze (choose a new configuration)
- (P) plan how to move to the new configuration
- (E) execute the plan
- (K) the set of models shared by MAPE
- State of the art mainly on (A) and (P)
  - considering simple (K) models for (E)
  - no parallelism, no distribution etc.
- We focus on building new (E) models
  - Aeolus is the closest in terms of modeling
  - Ansible is the better production tool for expressivity freedom (but hand-coded)

Concerto



Concerto = Madeus + Behaviors + ScoreL language

Concerto

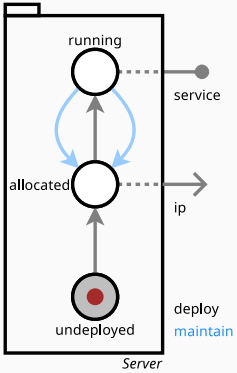


Concerto = Madeus + Behaviors + ScoreL language

Concerto - Deployment example

Example (Deployment)

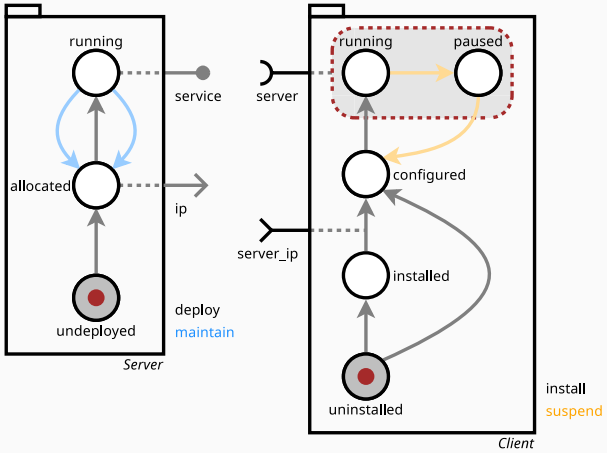
```
add(s : Server)
add(c : Client)
con(s.ip, c.server_ip)
con(s.service, c.server)
pushB(s, deploy)
pushB(c, install)
wait(c)
```



Concerto - Deployment example

Example (Deployment)

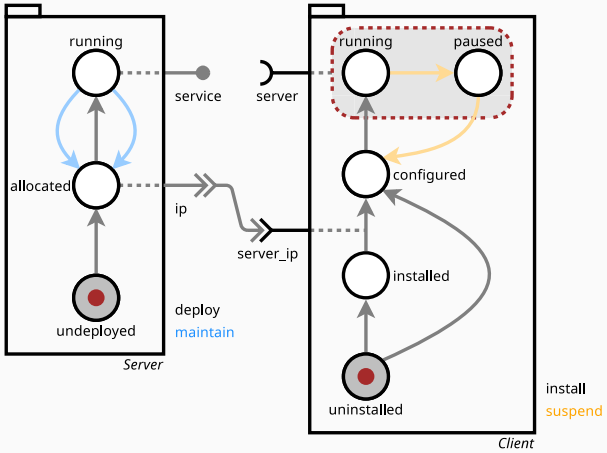
```
add(s : Server)
add(c : Client)
con(s.ip, c.server_ip)
con(s.service, c.server)
pushB(s, deploy)
pushB(c, install)
wait(c)
```



Concerto - Deployment example

Example (Deployment)

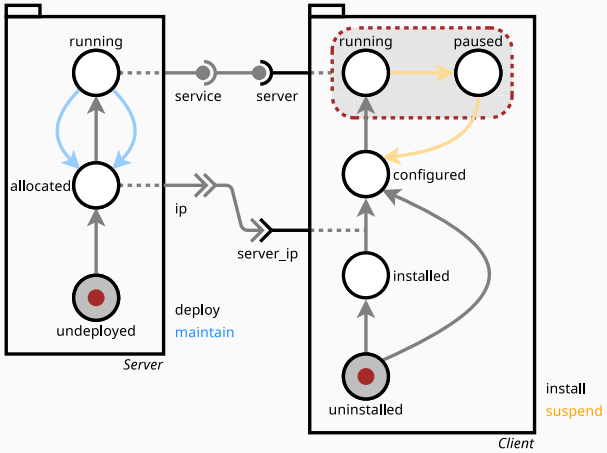
```
add(s : Server)
add(c : Client)
con(s.ip, c.server_ip)
con(s.service, c.server)
pushB(s, deploy)
pushB(c, install)
wait(c)
```



Concerto - Deployment example

Example (Deployment)

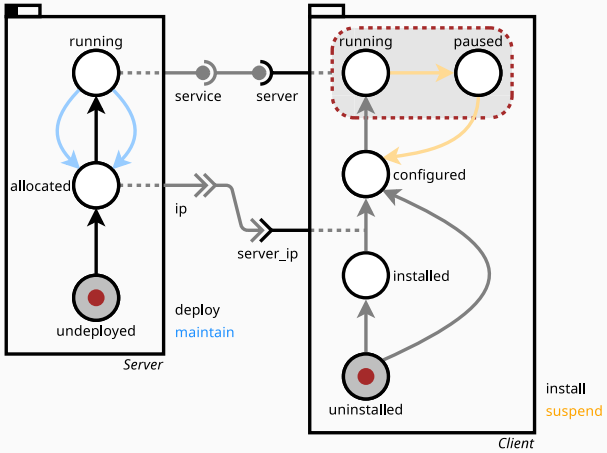
```
add(s : Server)
add(c : Client)
con(s.ip, c.server_ip)
con(s.service, c.server)
pushB(s, deploy)
pushB(c, install)
wait(c)
```



Concerto - Deployment example

Example (Deployment)

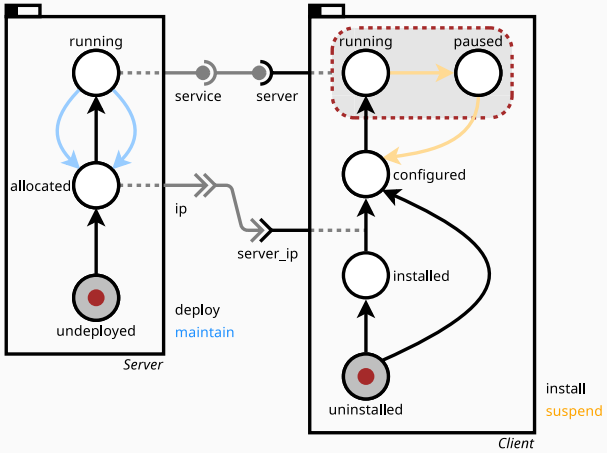
```
add(s : Server)
add(c : Client)
con(s.ip, c.server_ip)
con(s.service, c.server)
pushB(s, deploy)
pushB(c, install)
wait(c)
```



Concerto - Deployment example

Example (Deployment)

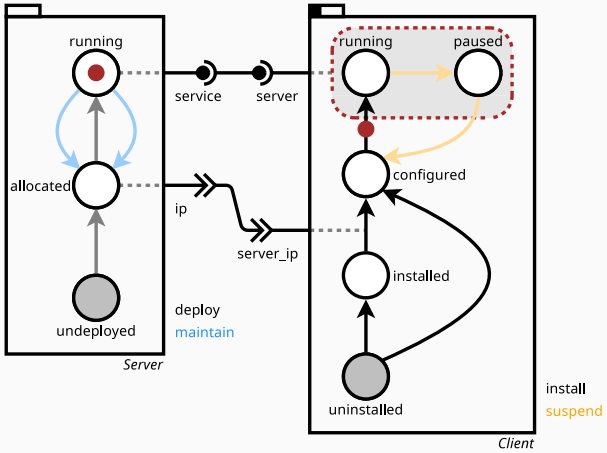
```
add(s : Server)
add(c : Client)
con(s.ip, c.server_ip)
con(s.service, c.server)
pushB(s, deploy)
pushB(c, install)
wait(c)
```



Concerto - Deployment example

Example (Deployment)

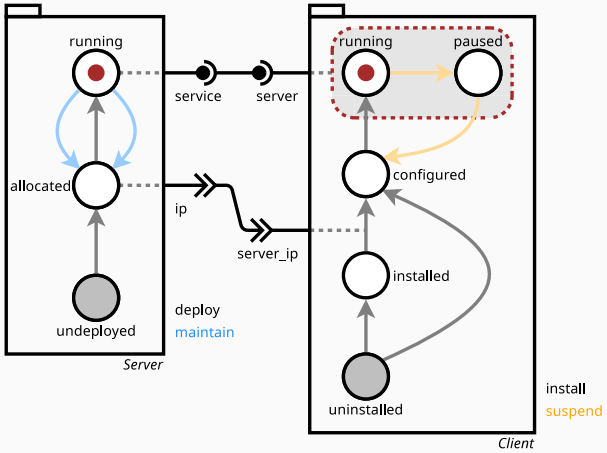
```
add(s : Server)
add(c : Client)
con(s.ip, c.server_ip)
con(s.service, c.server)
pushB(s, deploy)
pushB(c, install)
wait(c)
```



Concerto - Deployment example

Example (Deployment)

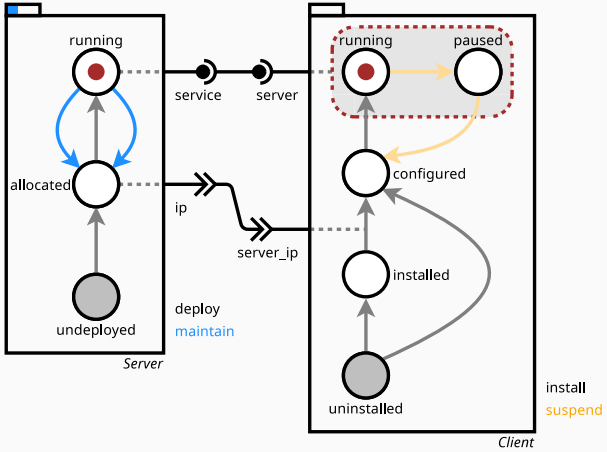
```
add(s : Server)
add(c : Client)
con(s.ip, c.server_ip)
con(s.service, c.server)
pushB(s, deploy)
pushB(c, install)
wait(c)
```



Concerto - Maintenance example

Example (Maintenance)

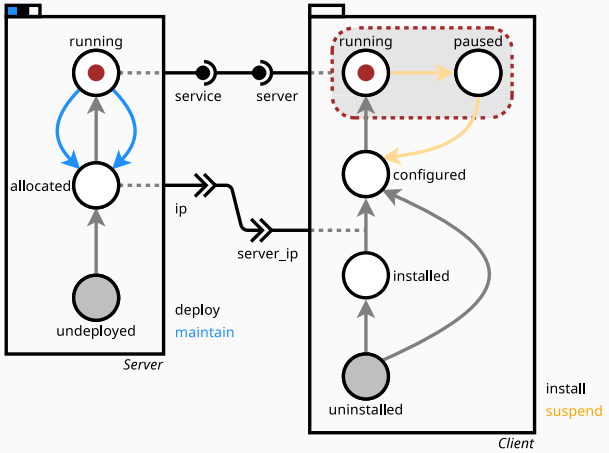
```
pushB(s, maintain)
pushB(s, deploy)
pushB(c, suspend)
wait(s)
pushB(c, install)
wait(c)
```



Concerto - Maintenance example

Example (Maintenance)

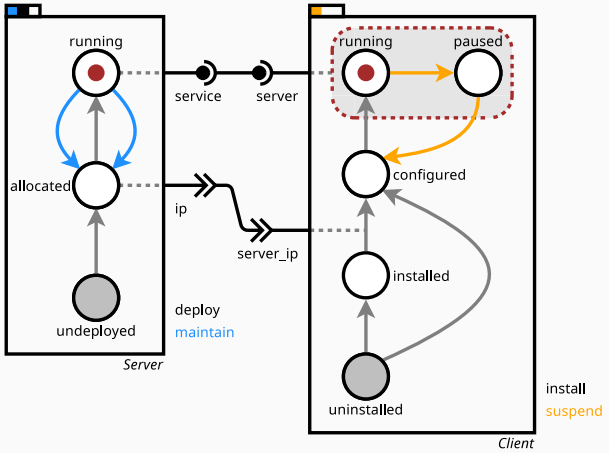
```
pushB(s, maintain)
pushB(s, deploy)
pushB(c, suspend)
wait(s)
pushB(c, install)
wait(c)
```



Concerto - Maintenance example

Example (Maintenance)

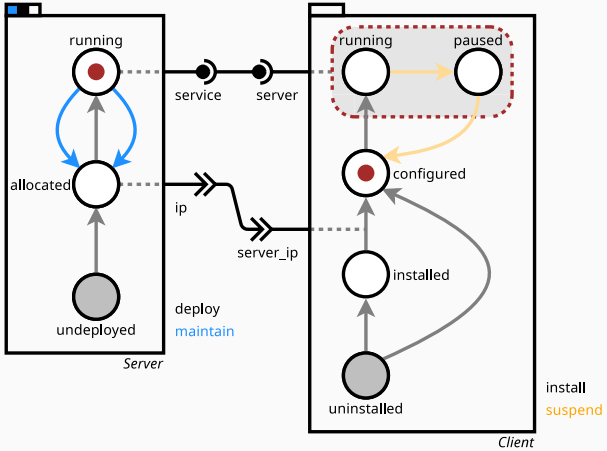
```
pushB(s, maintain)
pushB(s, deploy)
pushB(c, suspend)
wait(s)
pushB(c, install)
wait(c)
```



Concerto - Maintenance example

Example (Maintenance)

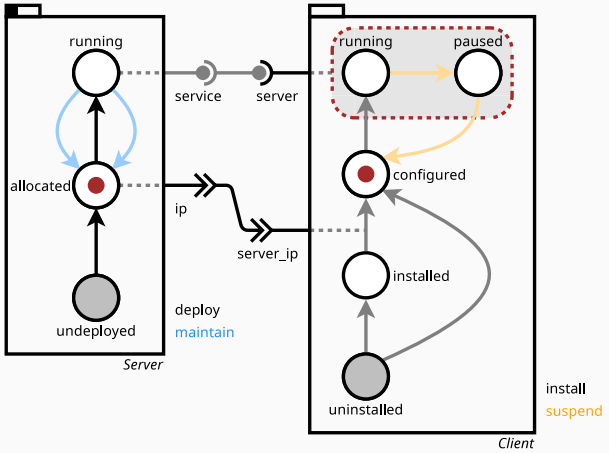
```
pushB(s, maintain)
pushB(s, deploy)
pushB(c, suspend)
wait(s)
pushB(c, install)
wait(c)
```



Concerto - Maintenance example

Example (Maintenance)

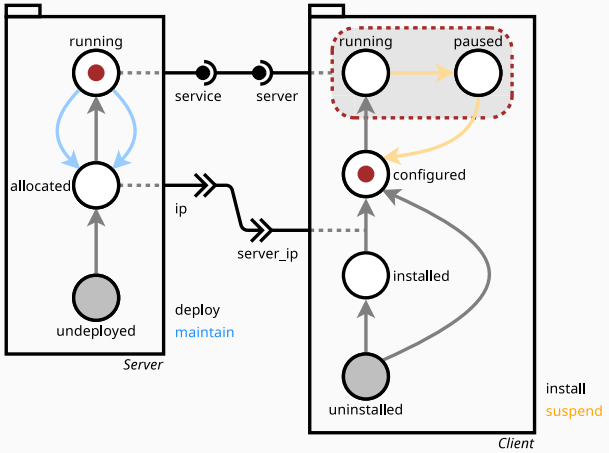
```
pushB(s, maintain)
pushB(s, deploy)
pushB(c, suspend)
wait(s)
pushB(c, install)
wait(c)
```



Concerto - Maintenance example

Example (Maintenance)

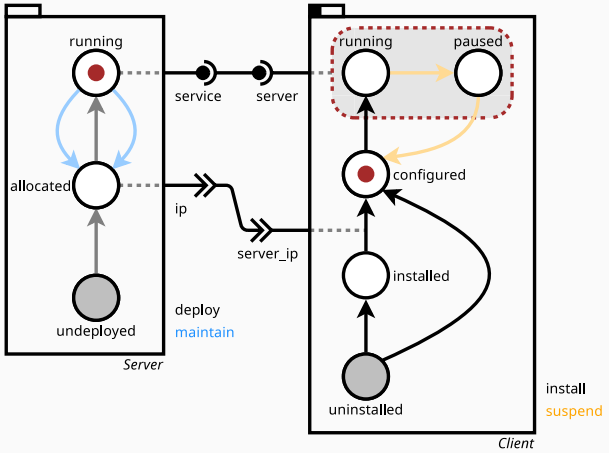
```
pushB(s, maintain)
pushB(s, deploy)
pushB(c, suspend)
wait(s)
pushB(c, install)
wait(c)
```



Concerto - Maintenance example

Example (Maintenance)

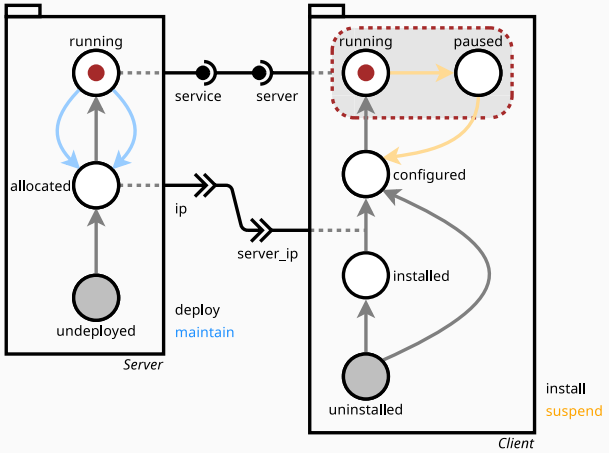
```
pushB(s, maintain)
pushB(s, deploy)
pushB(c, suspend)
wait(s)
pushB(c, install)
wait(c)
```



Concerto - Maintenance example

Example (Maintenance)

```
pushB(s, maintain)
pushB(s, deploy)
pushB(c, suspend)
wait(s)
pushB(c, install)
wait(c)
```



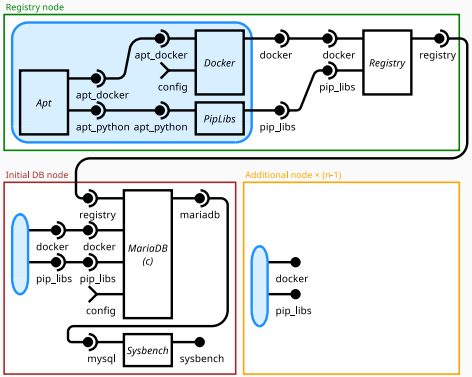
Experiments - decentralized DB (1/3)

Use-case

- From centralized MariaDB to decentralized MariaDB
- Galera cluster of MariaDBs (see the OpenStack summit)
- Reference code written with Ansible (see Juice)

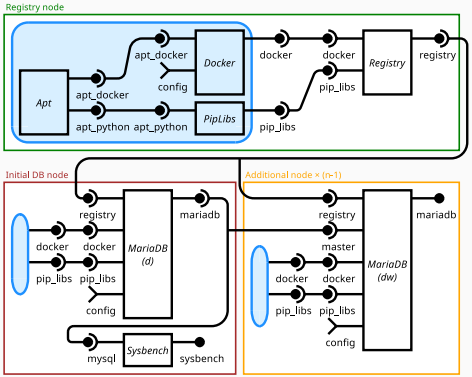
Experiments - decentralized DB (2/3)

- Deployment: Docker registry, MariaDB
- Reconf1: Docker registry, 1 MariaDB master, 1 MariaDB worker
- Reconf2: Docker registry, 1 MariaDB master, N MariaDB worker



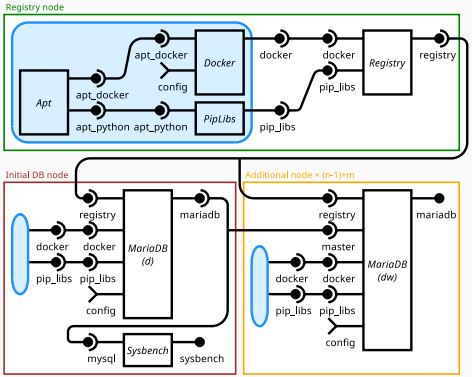
Experiments - decentralized DB (2/3)

- Deployment: Docker registry, MariaDB
- Reconf1: Docker registry, 1 MariaDB master, 1 MariaDB worker
- Reconf2: Docker registry, 1 MariaDB master, N MariaDB worker



Experiments - decentralized DB (2/3)

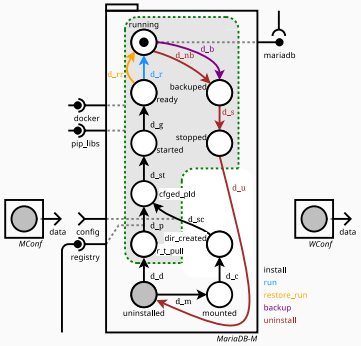
- Deployment: Docker registry, MariaDB
- Reconf1: Docker registry, 1 MariaDB master, 1 MariaDB worker
- Reconf2: Docker registry, 1 MariaDB master, N MariaDB worker



Experiments - decentralized DB (3/3)

Example (Deployment)

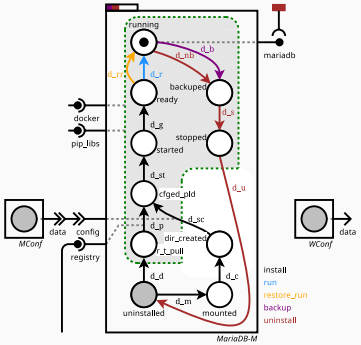
```
pushB (sysbenchm, suspend)
pushB (mariadb1, backup)
pushB (mariadb1, uninstall)
con (mconf.data, mariadb1.config)
for i in 2..n: [n+1..n+m]
  add (mariadb{i} : MariaDBw)
  con (wconf.data, mariadb{i}.config)
  con (mariadb1.mariadb, mariadb{i}.master)
  con (docker{i}.docker, mariadb{i}.docker)
  con (piplibs{i}.pip_libs,mariadb{i}.pip_libs)
  con (r_registry.registry,mariadb{i}.registry)
for i in 1..n:
  pushB (mariadb{i}, install)
pushB (mariadb1, restore_run)
pushB (sysbenchm, install)
```



Experiments - decentralized DB (3/3)

Example (Deployment)

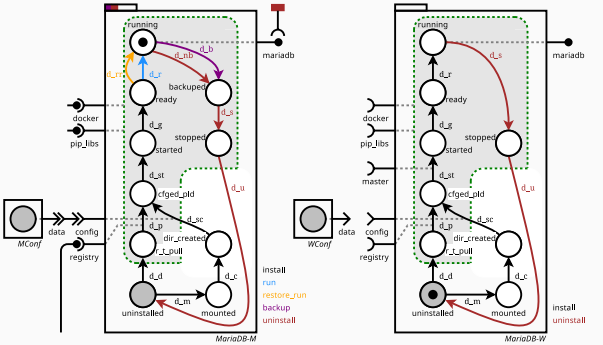
```
pushB (sysbenchm, suspend)
pushB (mariadb1, backup)
pushB (mariadb1, uninstall)
con (mconf.data, mariadb1.config)
for i in 2..n: [n+1..n+m]
  add (mariadb{i} : MariaDBw)
  con (wconf.data, mariadb{i}.config)
  con (mariadb1.mariadb, mariadb{i}.master)
  con (docker{i}.docker, mariadb{i}.docker)
  con (piplibs{i}.pip_libs,mariadb{i}.pip_libs)
  con (r_registry.registry,mariadb{i}.registry)
for i in 1..n:
  pushB (mariadb{i}, install)
pushB (mariadb1, restore_run)
pushB (sysbenchm, install)
```



Experiments - decentralized DB (3/3)

Example (Deployment)

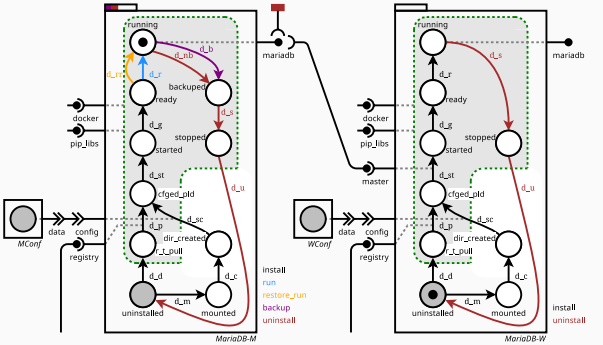
```
pushB (sysbenchm, suspend)
pushB (mariadb1, backup)
pushB (mariadb1, uninstall)
con (mconf.data, mariadb1.config)
for i in 2..n: [n+1..n+m]
  add (mariadb{i} : MariaDBw)
  con (wconf.data, mariadb{i}.config)
  con (mariadb1.mariadb, mariadb{i}.master)
  con (docker{i}.docker, mariadb{i}.docker)
  con (piplibs{i}.pip_libs,mariadb{i}.pip_libs)
  con (r_registry.registry,mariadb{i}.registry)
for i in 1..n:
  pushB (mariadb{i}, install)
pushB (mariadb1, restore_run)
pushB (sysbenchm, install)
```



Experiments - decentralized DB (3/3)

Example (Deployment)

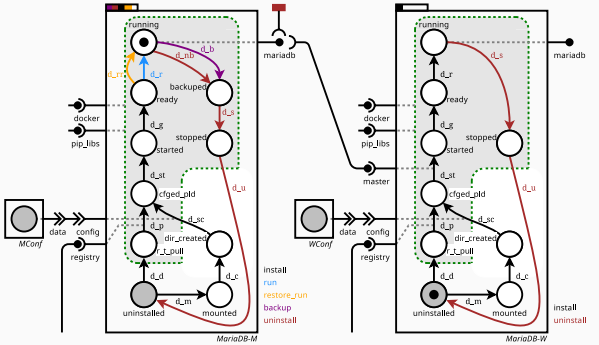
```
pushB (sysbenchm, suspend)
pushB (mariadb1, backup)
pushB (mariadb1, uninstall)
con (mconf.data, mariadb1.config)
for i in 2..n: [n+1..n+m]
  add (mariadb{i} : MariaDBw)
  con (wconf.data, mariadb{i}.config)
  con (mariadb1.mariadb, mariadb{i}.master)
  con (docker{i}.docker, mariadb{i}.docker)
  con (piplibs{i}.pip_libs,mariadb{i}.pip_libs)
  con (r_registry.registry,mariadb{i}.registry)
for i in 1..n:
  pushB (mariadb{i}, install)
pushB (mariadb1, restore_run)
pushB (sysbenchm, install)
```



Experiments - decentralized DB (3/3)

Example (Deployment)

```
pushB (sysbenchm, suspend)
pushB (mariadb1, backup)
pushB (mariadb1, uninstall)
con (mconf.data, mariadb1.config)
for i in 2..n: [n+1..n+m]
  add (mariadb{i} : MariaDBw)
  con (wconf.data, mariadb{i}.config)
  con (mariadb1.mariadb, mariadb{i}.master)
  con (docker{i}.docker, mariadb{i}.docker)
  con (piplibs{i}.pip_libs,mariadb{i}.pip_libs)
  con (r_registry.registry,mariadb{i}.registry)
for i in 1..n:
  pushB (mariadb{i}, install)
pushB (mariadb1, restore_run)
pushB (sysbenchm, install)
```



Safety

Why safety?

Getting guarantees on commissioning and reconfiguration

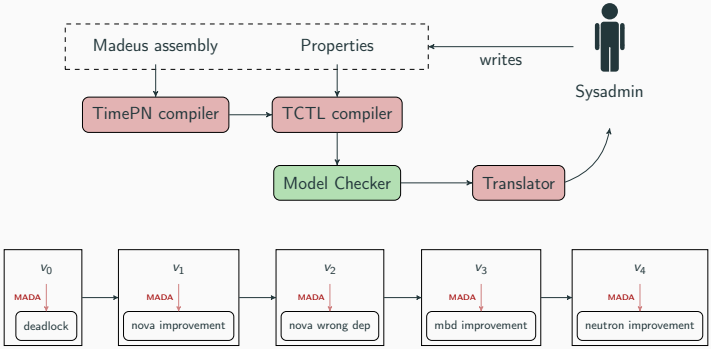
- attainability
- liveness
- invariants on the validity of the assembly
- performance analysis

Using formal methods as an helping tool to design safe and efficient procedures

- efficiency analysis
- fault injections
- uncertainty

MADA - Overview

Study the use of model checking to help in the design of safe and efficient distributed software commissioning



MADA - Properties

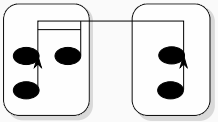
- Time Petri nets are used
  - intervals of time given for each transition representing a Madeus transition

```
def set_interval(self, component, transition, min, max)
def add_deployment(self, name, dict_componentsplaces)
```

- High Abstraction Level Properties (HALP)
  - qualitative properties
  - quantitative properties

```
def deployability(self, deployment_name, with_intervals)
def sequentiality(self, ordered_list_components_transition)
def forbidden(self, list_marked, list_unmarked)
def parallelism(self, full_assembly, list_components)
def gantt_boundaries(self, deployment_name, mini, maxi, critical)
```

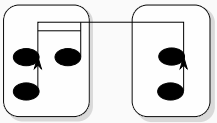
VeRDi - Project



Verified Reconfiguration Driven by execution

- Charlène Servantie - Engineer for 18 months
- Dimitri Pertin - Engineer for 2 months
- Simon Robillard - Postdoc for 18 months

VeRDi - On going work



Verified Reconfiguration Driven by execution

- model checking approach for Concerto (such as MADA)
- compositional verification of Concerto programs
- synthesis of reconfiguration scripts in Concerto

Perspectives

Perspectives

Perspectives for autonomous reconfiguration

- decentralized reconfiguration
- online (A) and (P) decisions

Perspectives for safety

- handling errors in reconfiguration
- handling uncertainty in reconfiguration
- more complex control loop including formal verifications

Perspectives

Collaboration on the DAO project?

- Madeus and Concerto offer safe and efficient coordination for reconfiguration
- could be extended to Cyber-physical systems reconfiguration

Thank you

Thank you !

Questions?