IMT Atlantique
Bretagne-Pays de la Loire
École Mines-Télécom

TOWARD EFFICIENT AND SAFE DISTRIBUTED SOFTWARE DEPLOYMENT

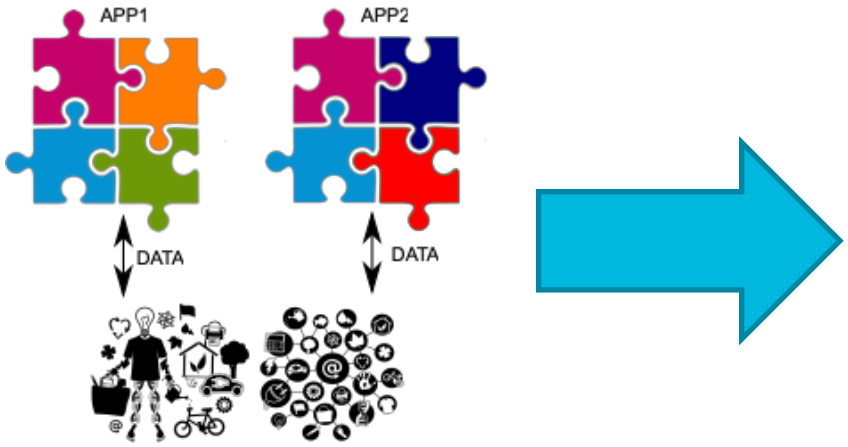
Maverick Chardet
Hélène Coullon
Christian Perez
Dimitri Pertin

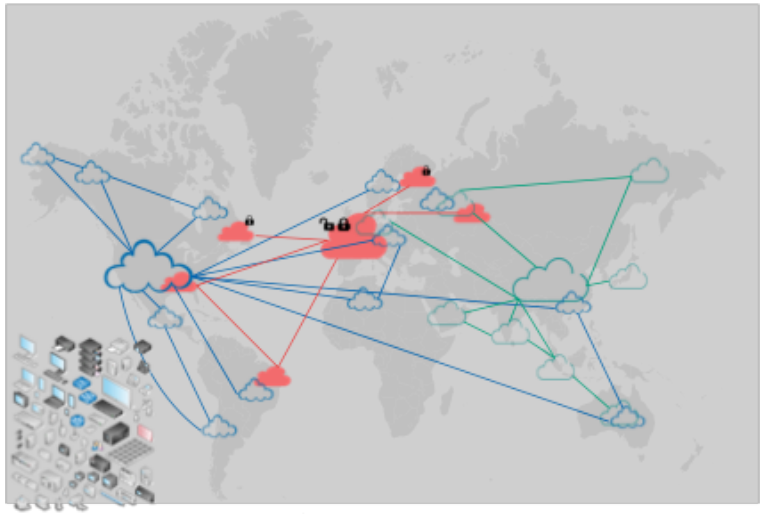
Inria Orange Lab 2018-10-30

1. MADEUS AND MAD

IMT Atlantique
Bretagne-Pays de la Loire
École Mines-Télécom

MOTIVATION - DEPLOYMENT





Deployment

1. Placement: mapping modules / components to resources
   - Bin-packing problems
2. Software commissioning / configuration
   - Allocate resources
   - Create and configure components / modules
   - Component interactions and dependencies

STATE OF THE ART

Most *component models* are made to:

- Clearly separate the different components/functionalities of applications
- Describe the functional interactions between components
- **Embed predefined life-cycles for components (*i.e.* Create, configure, destroy)**

Programmable life-cycle
- Improve *flexibility* and *expressivity* (granularity choice)

Automatic temporal coordination of life-cycles
- Improve *safety* (control)
- Introduction of *automatic parallelism* (performance)

Automatic Parallelism at 3 levels
1. Same-Component-Multiple-Host (SCMH)
2. Inter-component
3. Inter-life-cycles

Academic: Deployware, TOSCA, Aeolus, Fractal, GCM, GCM/Proactive
Production tools: Chef, Puppet, Ansible, Juju, Kubernetes

STATE OF THE ART

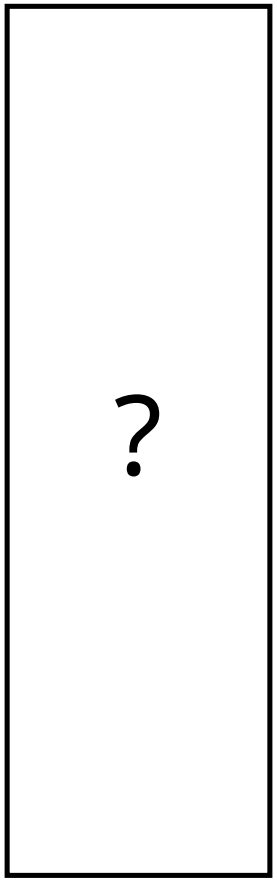
	Prog. Life-cycle	Temp. Coordination	Auto. Parallelism
Deployware	No	Yes (fixed order)	Inter-component
Fractal, GCM, GCM/proactive	Yes	No	Inter-component
TOSCA	Yes	Yes	Inter-component
Aeolus (Blender)	Yes	Yes	Inter-life-cycles

	Prog. Life-cycle	Temp. Coordination	Auto. Parallelism
Chef, Puppet, Ansible	Yes	Yes (seq order)	SCMH
Juju	Yes	Yes	Inter-component
Kubernetes	No	Yes (fixed order)	Inter-component

- **Madeus** is a *low-level deployment model*
- Any existing component can be encapsulated in Madeus
- Madeus is inspired from **Aeolus**
- Madeus focuses on **performance**
  - *Intra-component parallelism*

Component

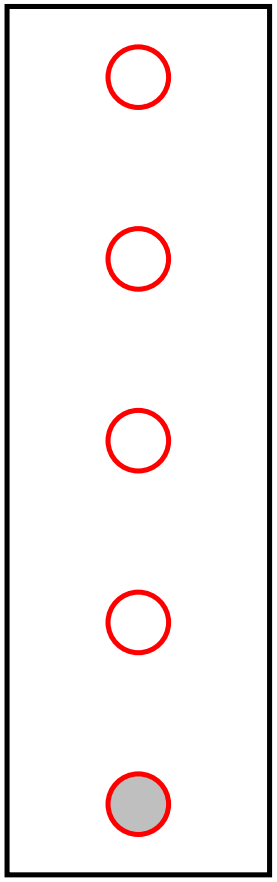
- Usually corresponds to a *module* of a distributed system or app
- Has its own life-cycle



MADEUS – PLACE

Place

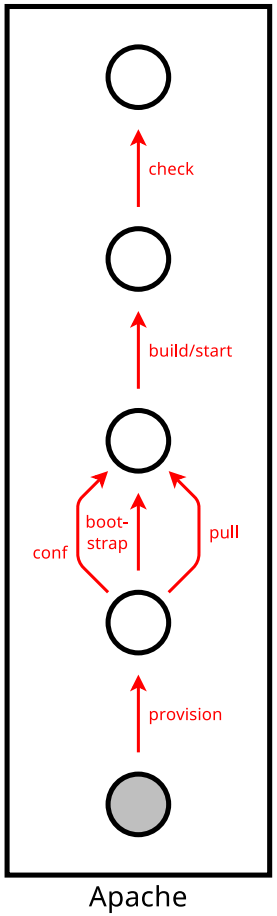
- A “*milestone*” in the component life-cycle
- Can act as a *synchronization* mark if multiple actions are performed in parallel



MADEUS – TRANSITION

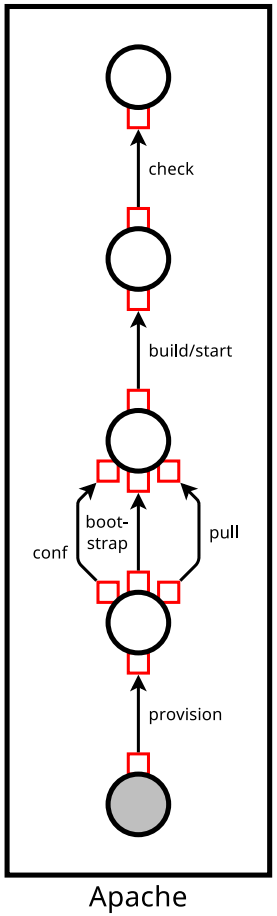
Transition

- Bound to an *action* (*i.e.* a function)



Dock

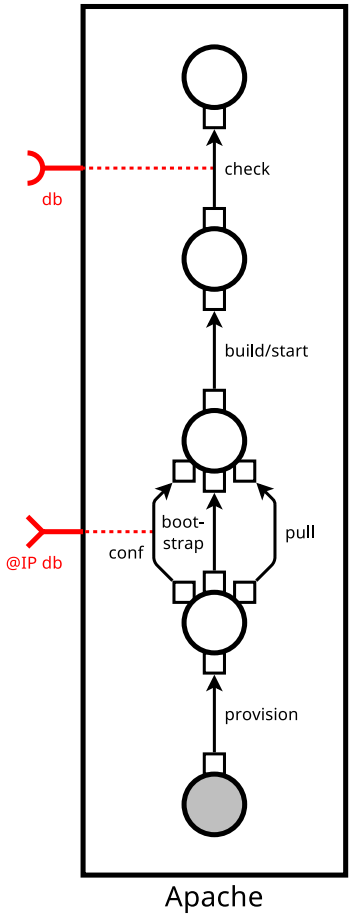
- Allows to handle *synchronization* of parallel actions
- Attached to a place
- Connection point for transitions
- Two kinds of docks: *input* and *output*



MADEUS – INPUT PORT

Input port

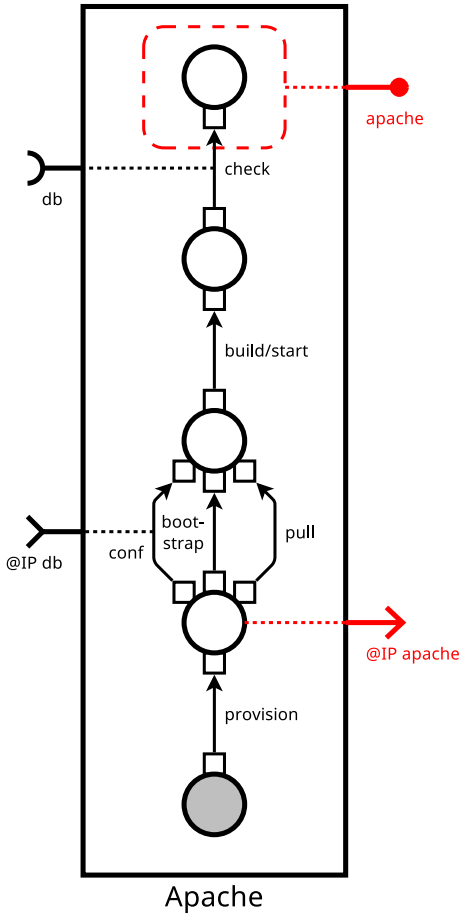
- Bound to transitions that require some data or service
- These transitions can only be triggered when the port is *connected*



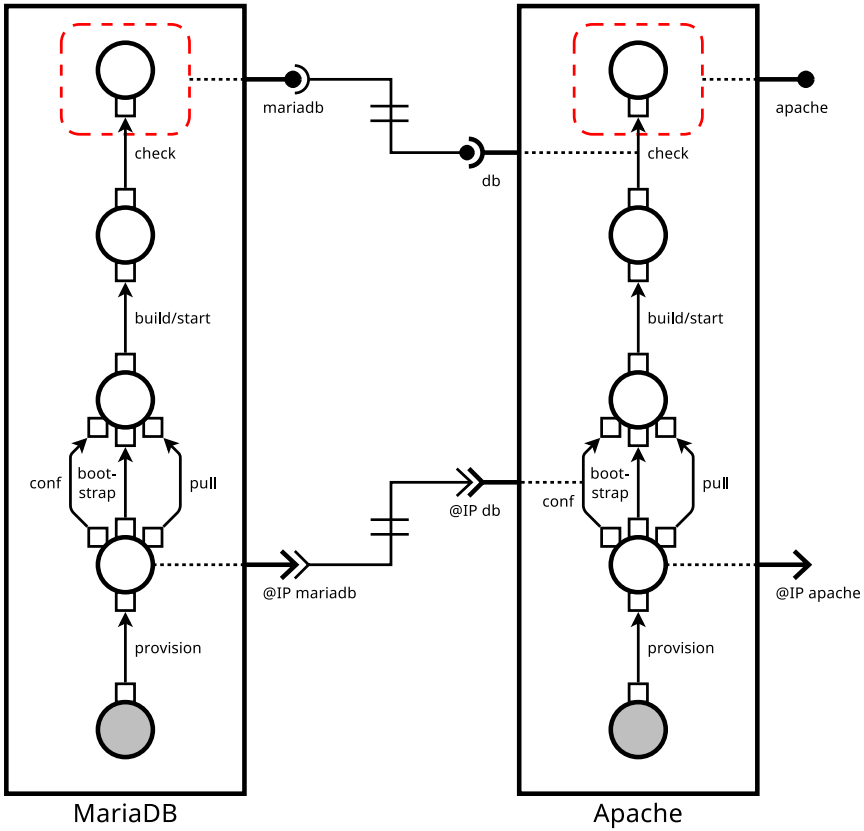
MADEUS – OUTPUT PORT

Output port

- Data output port: provides data / acts as a register (*e.g.* IP address)
- Service output port: indicates that a service is provided by the component



MADEUS – ASSEMBLY & CONFIGURATION



Token

- Represents the state in the life-cycle of the component
- Either present on or absent of each place, dock and transition

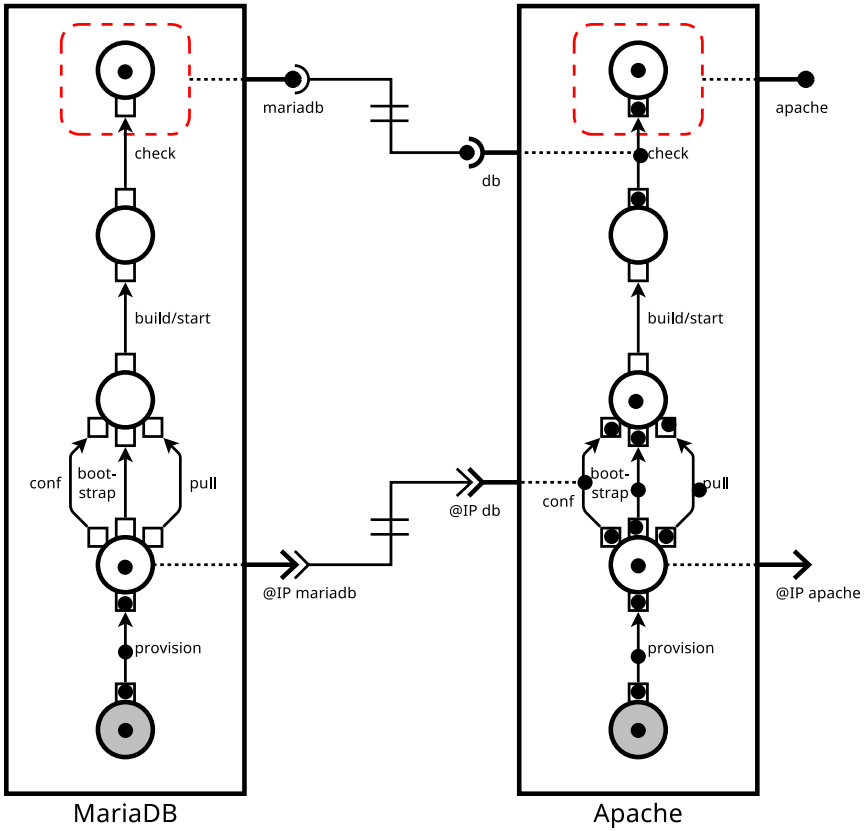
Assembly

- Set of components instances
- Connections between their ports
- Similar to a *main* function

Configuration <*mk, ebl, val*>

- *mk* marking = location of tokens
- *ebl* enabled = wether or not connections are enabled
- *val* values = values stored in the data output ports

MADEUS – SEMANTICS



7 semantics rules

1. *Leaving a place*
2. *Firing a transition*
3. *Ending a transition*
4. *Joining a place*

5. *Enabling use-provide connection*
6. *Disabling use-provide connection*
7. *Enabling data connection*

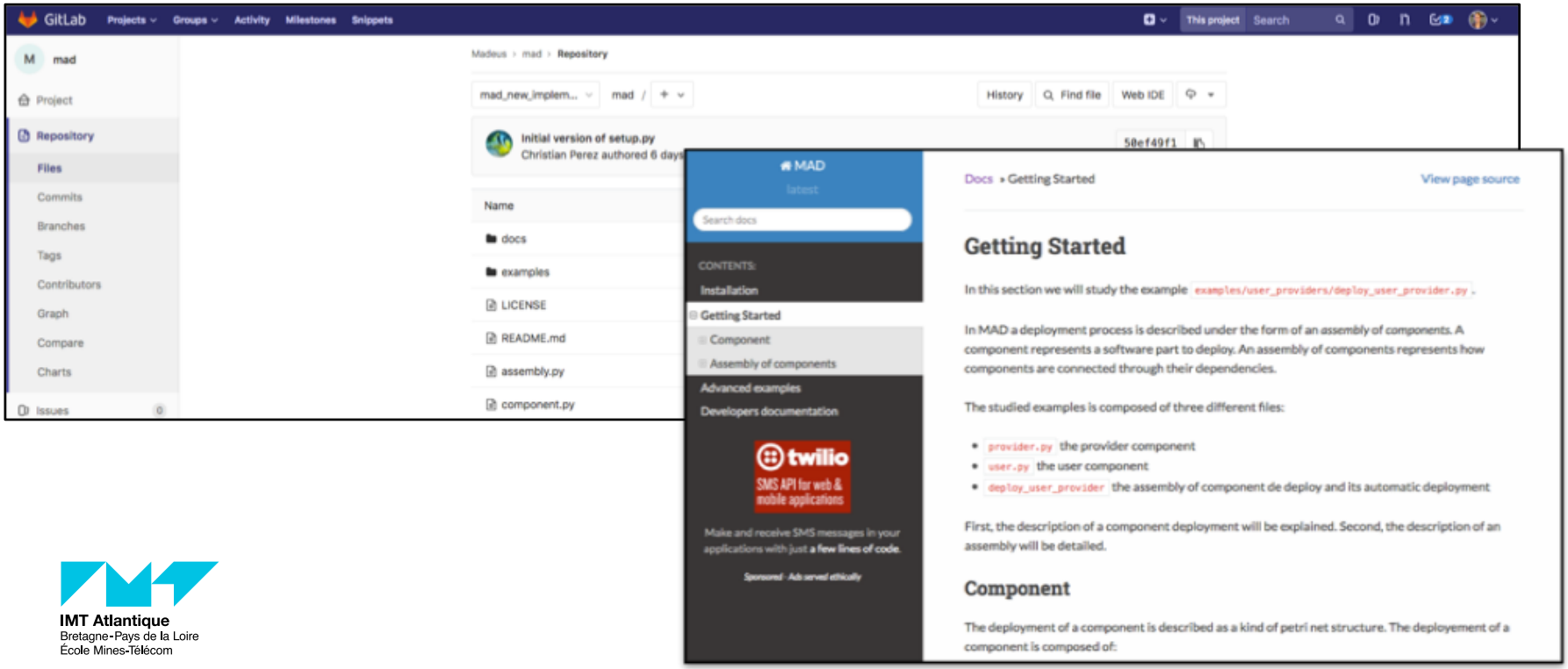
IMT Atlantique
Bretagne-Pays de la Loire
École Mines-Télécom

Madeus: A formal deployment model. Maverick Chardet, Hélène Coullon, Christian Perez, Dimitri Pertin. 4PAD symposium @ HPCS 2018.

MADEUS APPLICATION DEPLOYER

MAD

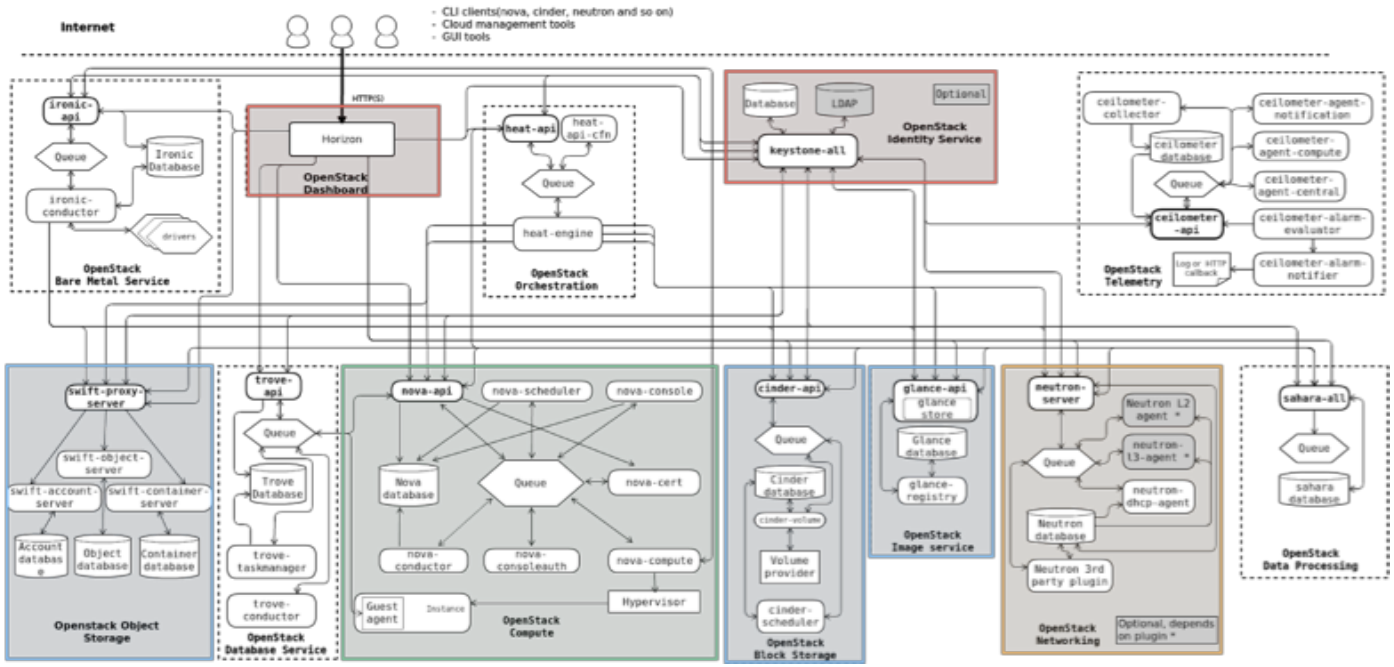
- Implementation of the Madeus model
- Written in Python
- v0.2 Open Source GPL v3
- Available at https://gitlab.inria.fr/Madeus/mad/tree/mad_new_implementation
- Documentation at https://mad.readthedocs.io/en/latest/



EVALUATION – CASE STUDY

OpenStack

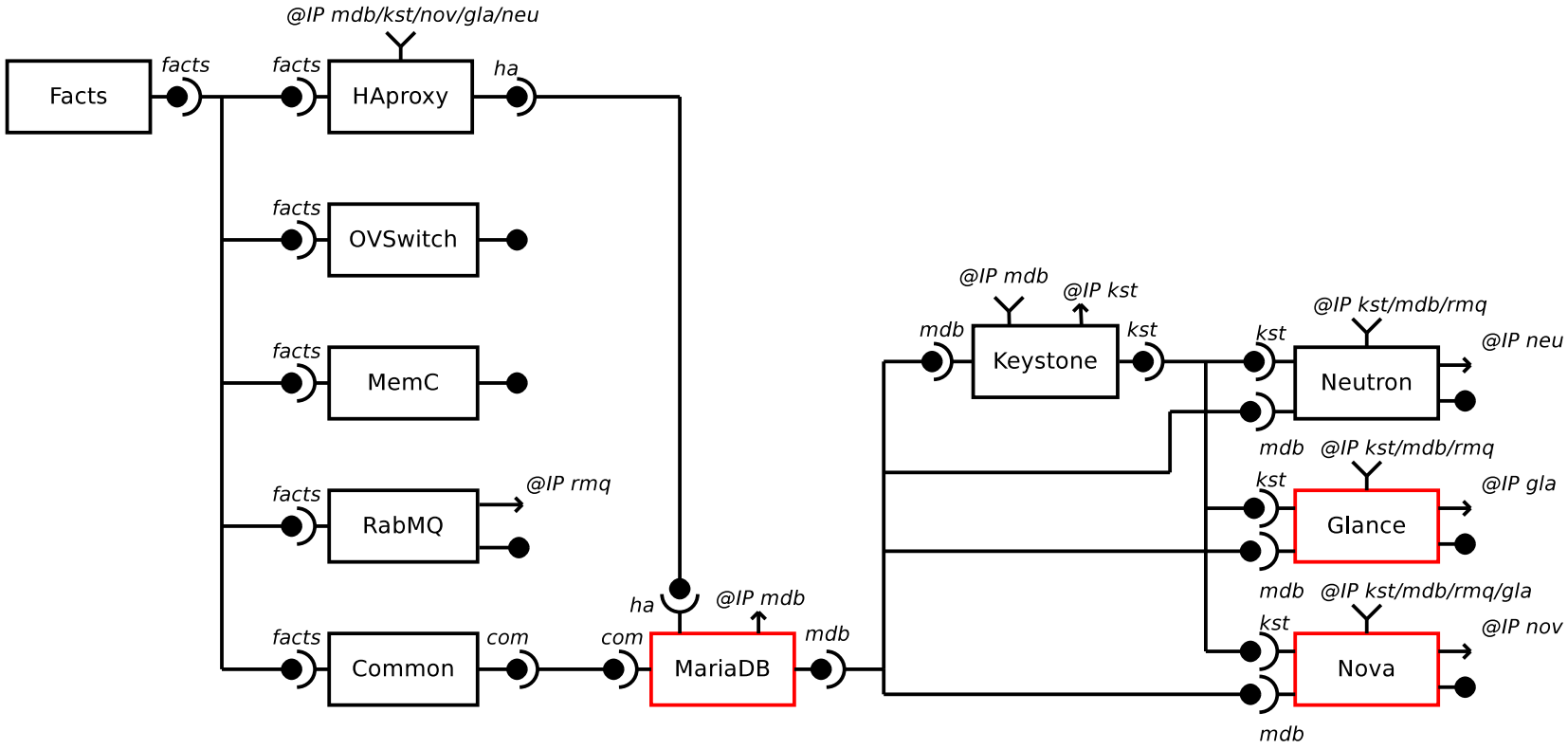
- Large distributed software
- Modular (component) architecture composed of more than 30 projects
- Each project composed of multiple services
- More than 150 services
- Ansible, Juju, Kubernetes, TripleO etc. has been used to deploy OpenStack



Kolla-Ansible OpenStack Deployment

- Our deployment reference is the Kolla project
  - Production tool to deploy OpenStack ONLY
  - Deploy a containerized minimal OpenStack by using **Ansible**
  - **11 projects**, **36 services**
  - Deployment on **three nodes**:
    - controller node (16 services)
    - Network node (11 services)
    - Compute node (9 services)

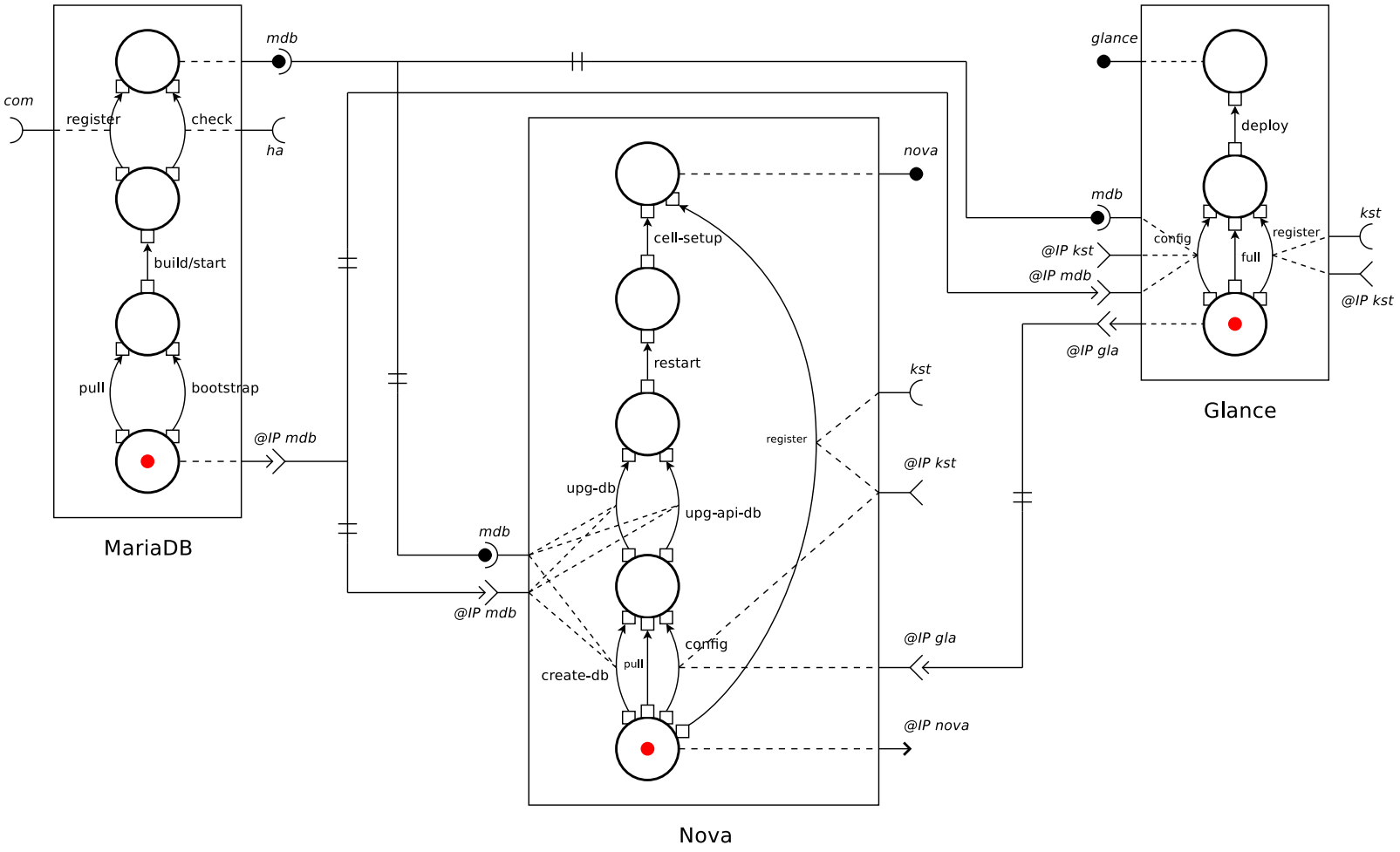
EVALUATION – EXPRESSIVITY



Kolla-ansible deployment in Madeus

- 11 Madeus components
- Component dependencies
  - Use-provide
  - Data

EVALUATION – EXPRESSIVITY



	Places	Transitions	Ports
Total	32	30	47

EVALUATION – PERFORMANCE

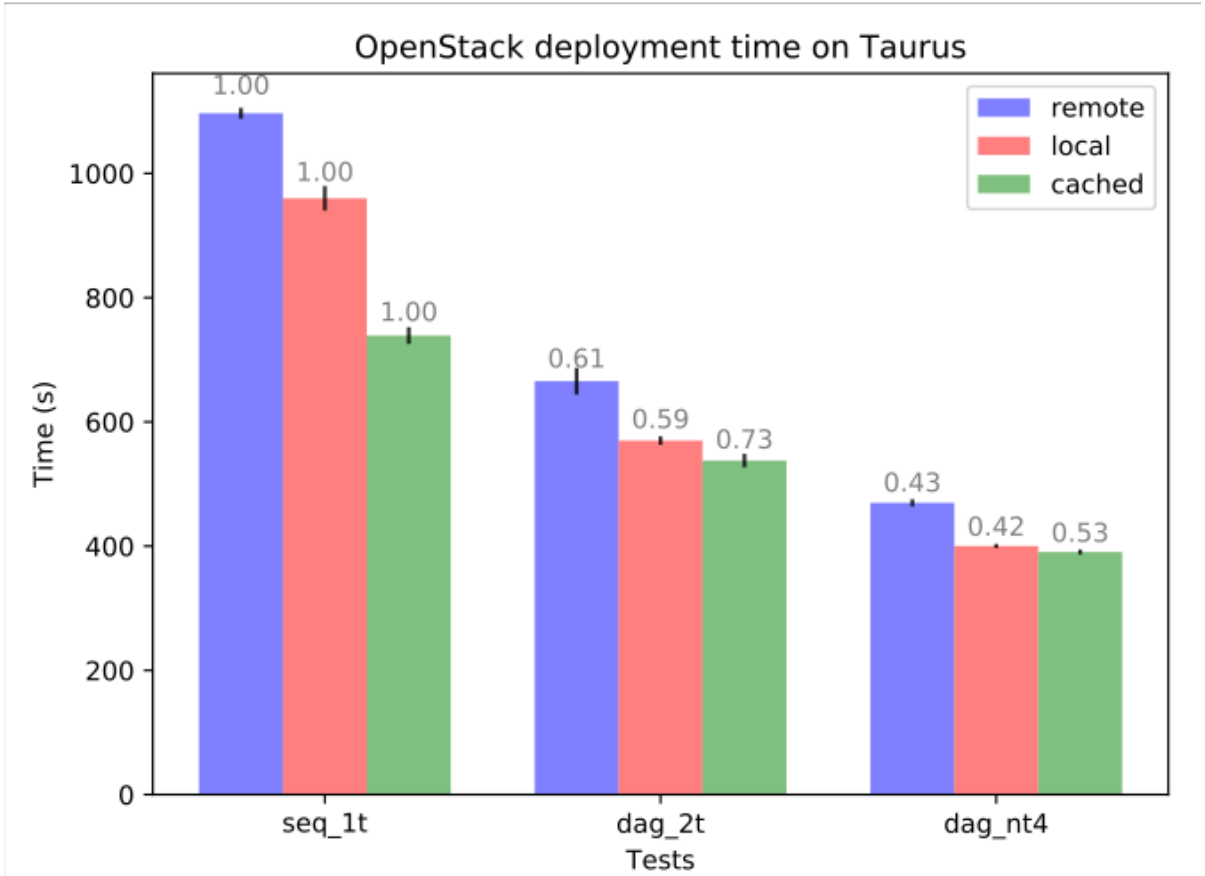
Cluster	CPU	Memory	Network
Taurus (Lyon)	2 x 6 cores / CPU	32 GB	10 Gbps

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

Kolla-ansible OpenStack Deployment

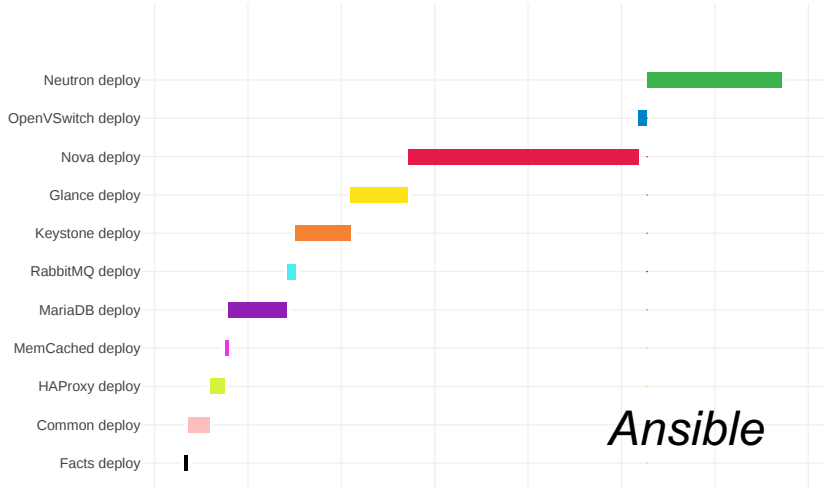
- Deployment versions
  - *spmd-1t* = **Kolla-ansible**
  - *dag-2t* = **Aeolus** (simulated with Madeus, no parallel transitions)
  - *dag-nt* = **Madeus**
- Docker image management
  - *Remote* (Docker Hub)
  - *Local* (dedicated local registry in the cluster)
  - *Cached* (all nodes already store docker images)

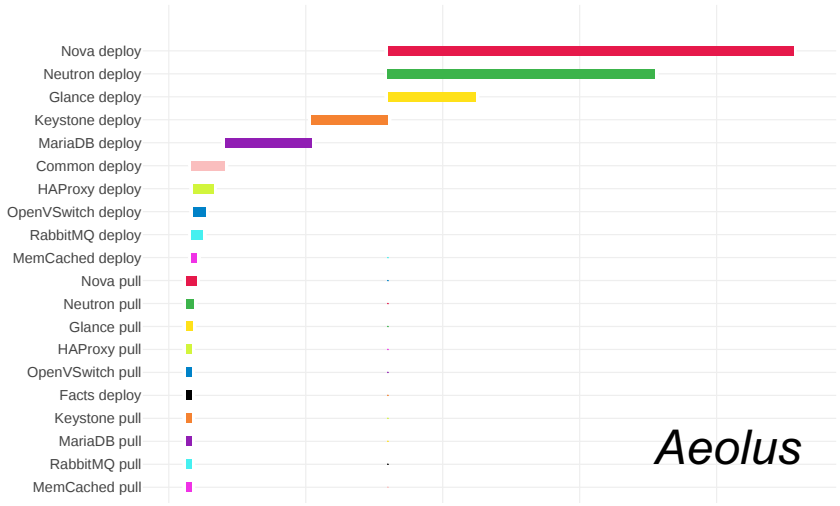
EVALUATION – PERFORMANCE

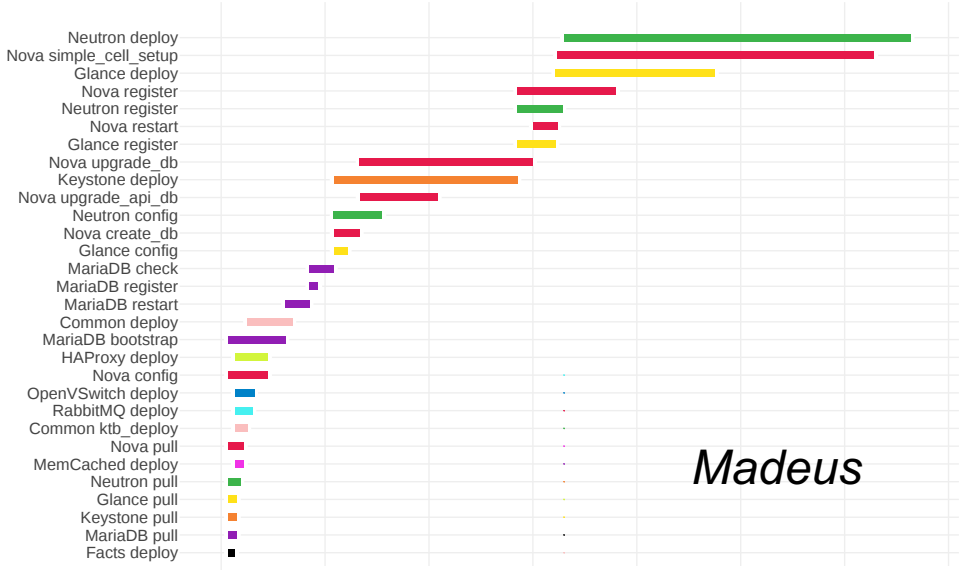


- Up to **58% gain compared to Kolla-Ansible**
- Up to 32% gain compared to Aeolus

EVALUATION – PERFORMANCE







IMT Atlantique
Bretagne-Pays de la Loire
École Mines-Télécom

CONCLUSION

Conclusion

- New formal deployment model Madeus
- Madeus adds a level of parallelism *intra-component*
- Madeus increases the *performance* of the deployment
- Evaluated on OpenStack in comparison to Kolla-Ansible

Perspectives

- *Model Checking* of Madeus assemblies, *Coq* formalization of Madeus
- *Performance model* of Madeus and *scheduling* algorithms
- *Decentralization* of Madeus
- Higher abstraction level tools: *MAD-Ansible* etc.
- Extension of Madeus to *reconfiguration*

Dimitri Pertin
ex-Postdoc
somwhere in Asia

Maverick Chardet
PhD student

Christian Perez
Research director Inria

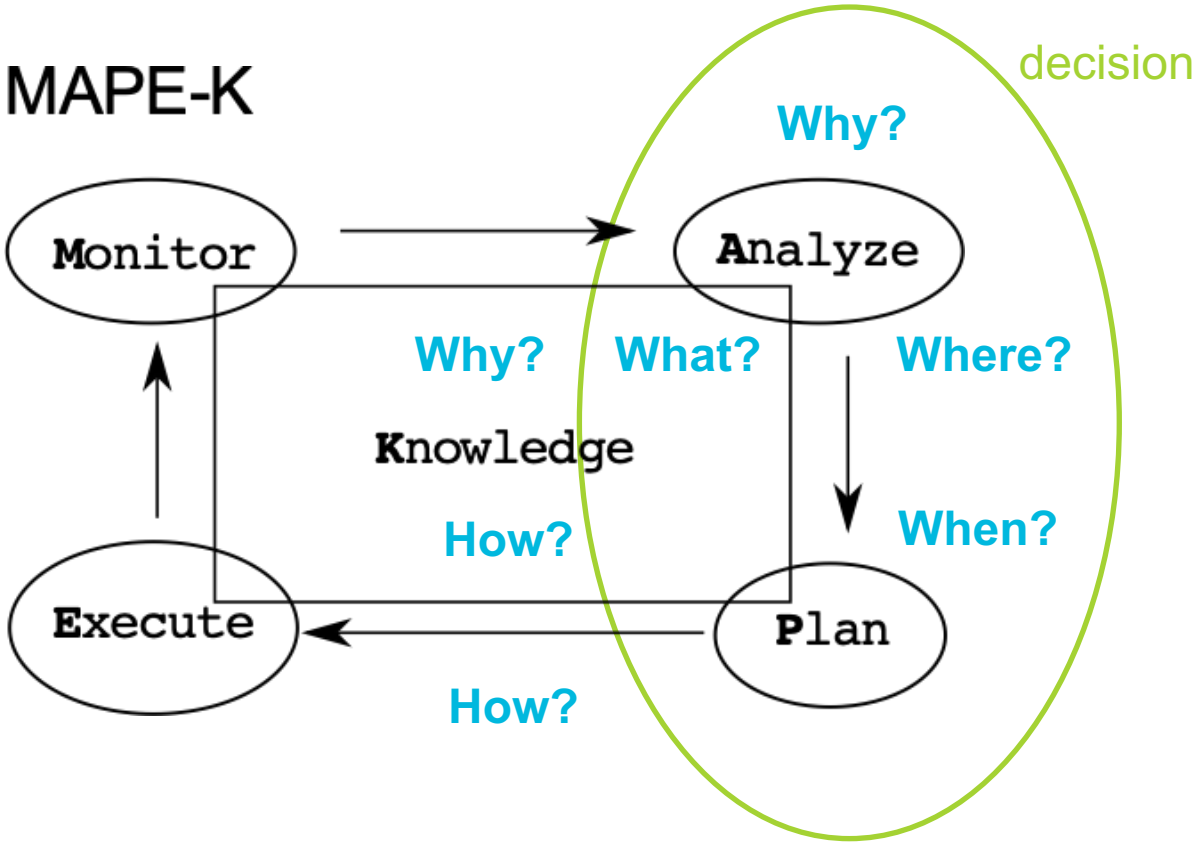
2. MADEUS-B

IMT Atlantique
Bretagne-Pays de la Loire
École Mines-Télécom

Deployment = specific reconfiguration

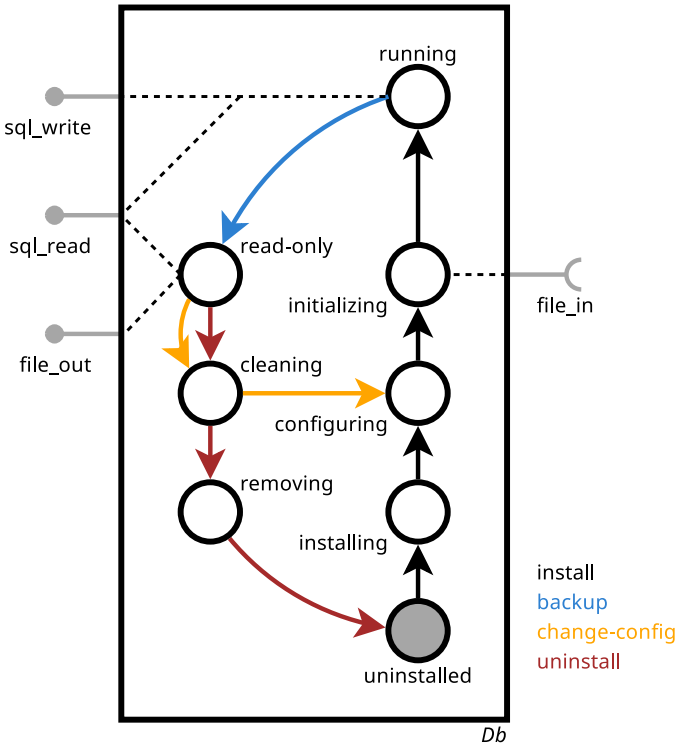
- Rolling upgrade
- Dynamic information regarding the infrastructure
  - Fault tolerance
  - Scalability
  - Performance models
- Dynamic external data events
  - IoT, smart-* applications
  - Dynamic energy considerations
  - Dynamic security considerations

MOTIVATION – AUTO RECONFIGURATION



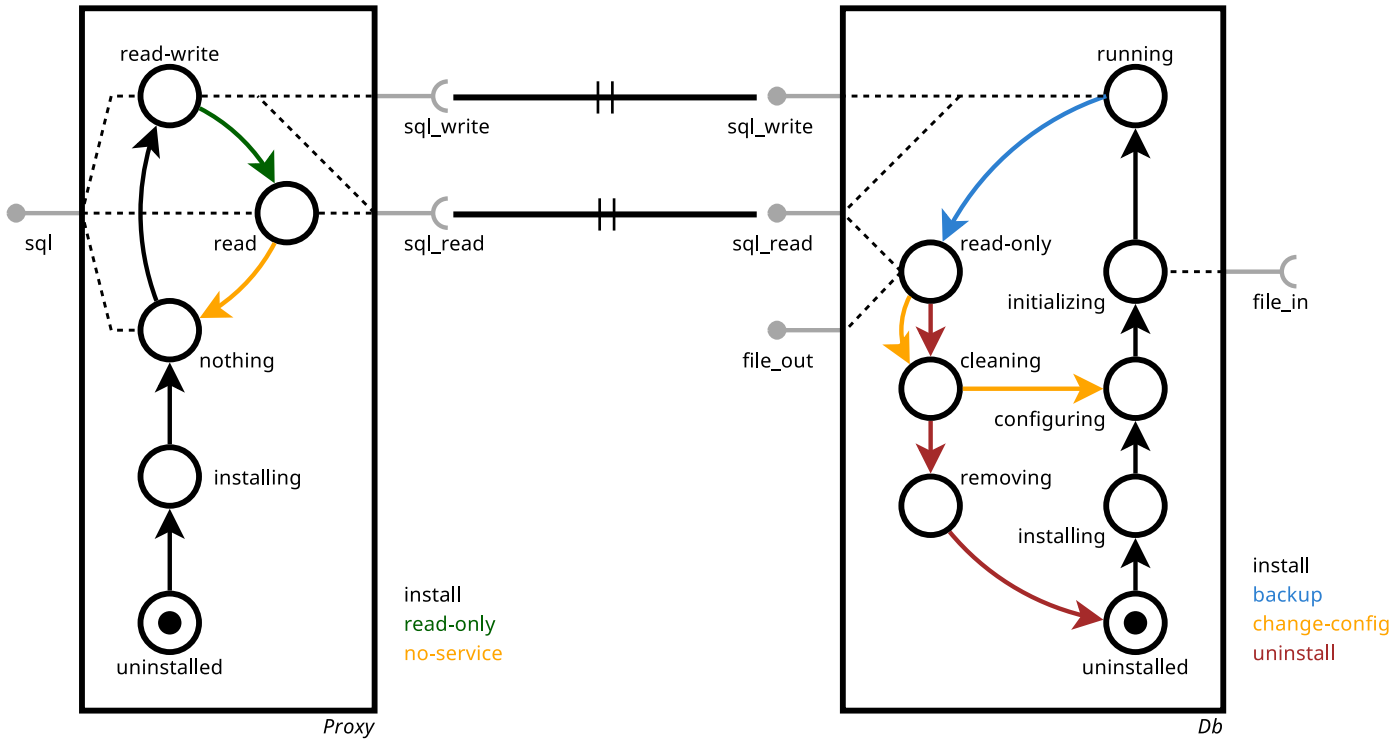
- Reconfiguration **Model** / ***Knowledge***: « ***what*** »?
- Reconfiguration ***Execution:*** « ***how*** »?
  - We consider the ***decision already done and known***

Introduction of behaviors in Madeus components

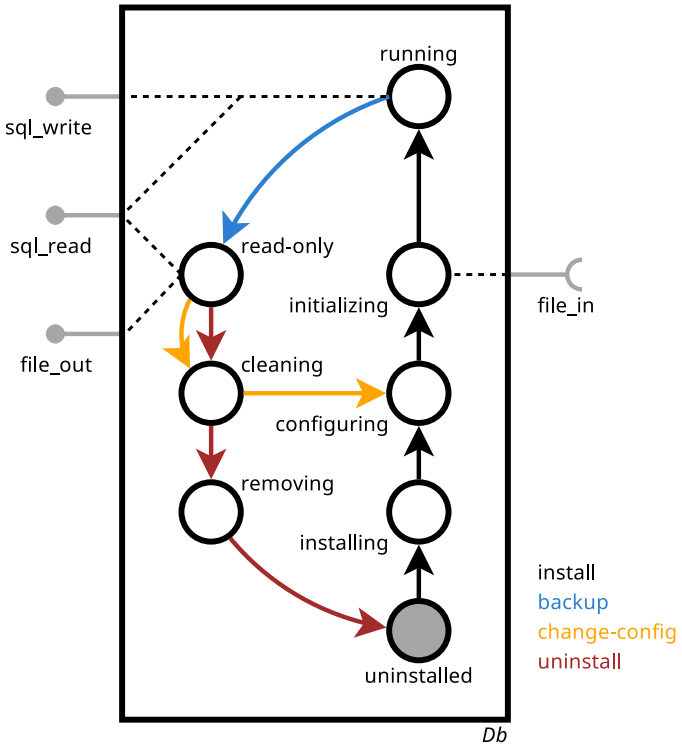


- A component can have as many behaviors as the developer needs
- A single behavior is active at runtime
- The behavior can be switched at runtime according to semantics rules

Assemblies


read-write
sql
read
nothing
installing
uninstalled
Proxy
sql_write
sql_read
file_out
install
read-only
no-service
sql_write
sql_read
running
read-only
initializing
cleaning
configuring
removing
installing
uninstalled
file_in
Db
install
backup
change-config
uninstall

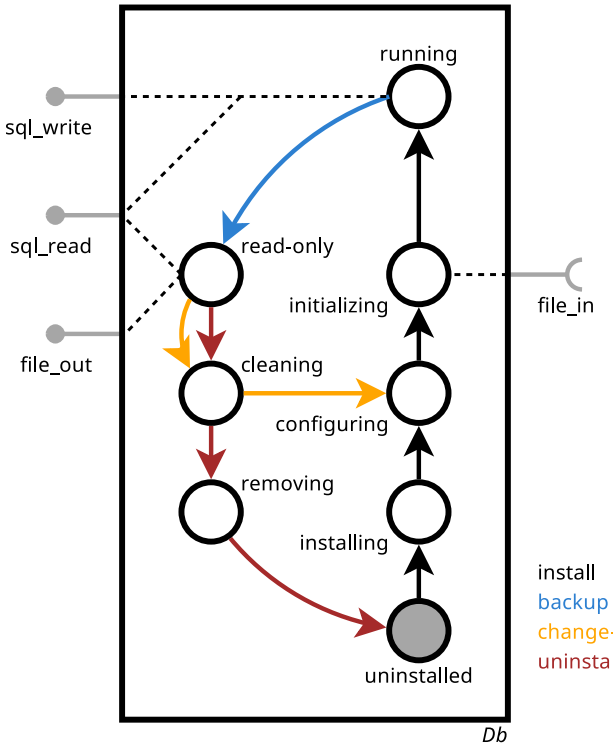
Reconfiguration language and semantics




```
behaviorChange (proxy, read-only)
wait (proxy)
disconnect (sql_write, db1, proxy)
new (db2 : Db)
new (transferer : Transferer)
connect (file_out, db1, transferer)
connect (file_in, transferer, db2)
behaviorChange (transferer, run)
behaviorChange (db2, install)
behaviorChange (db1, backup)
wait (db2)
behaviorChange (proxy, no-service)
wait (proxy)
disconnect (sql_read, db1, proxy)
connect (sql_read, db2, proxy)
connect (sql_write, db2, proxy)
behaviorChange (proxy, install)
behaviorChange(transferer, stop)
wait (transferer)
behaviorChange(db1, uninstall)
wait (db1)
del (db1)
del (transferer)
```

- Directly use behaviors in the reconfiguration
- *Asynchronism* in the reconfiguration (efficiency)

Improve separation of concerns

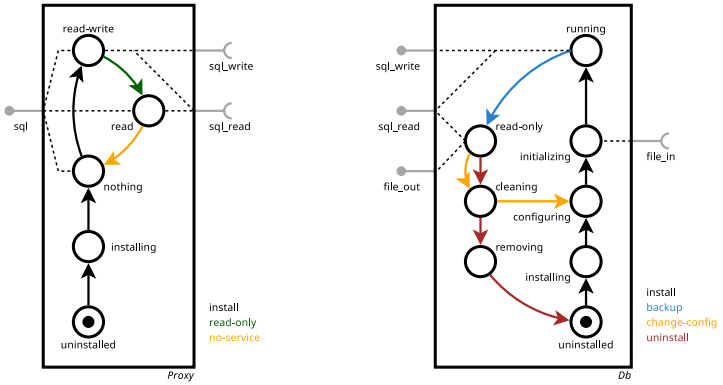


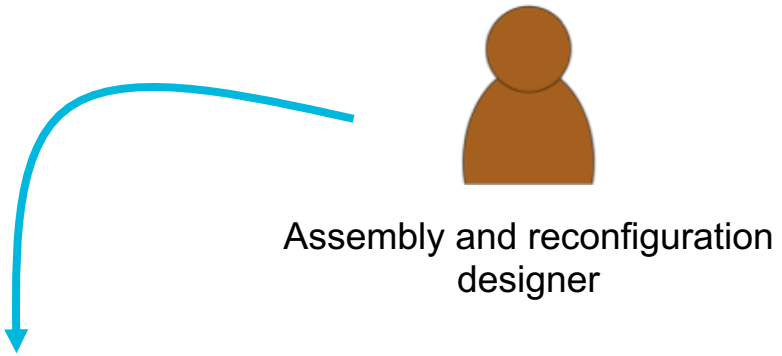
Component developer

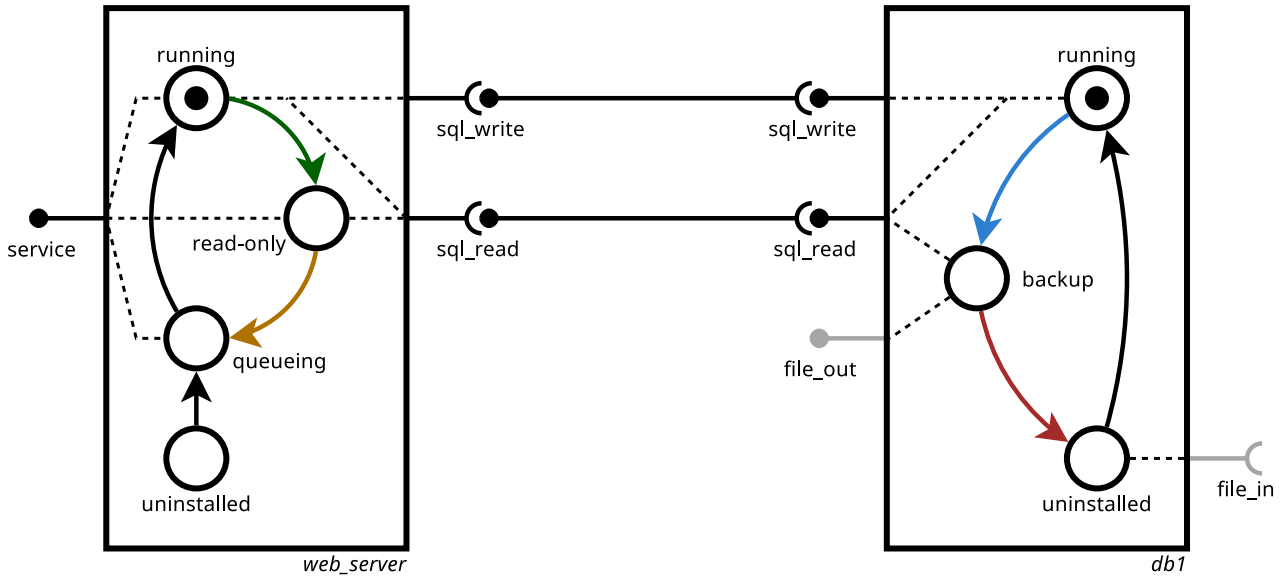
Assembly and reconfiguration designer

install
backup
change-config
uninstall

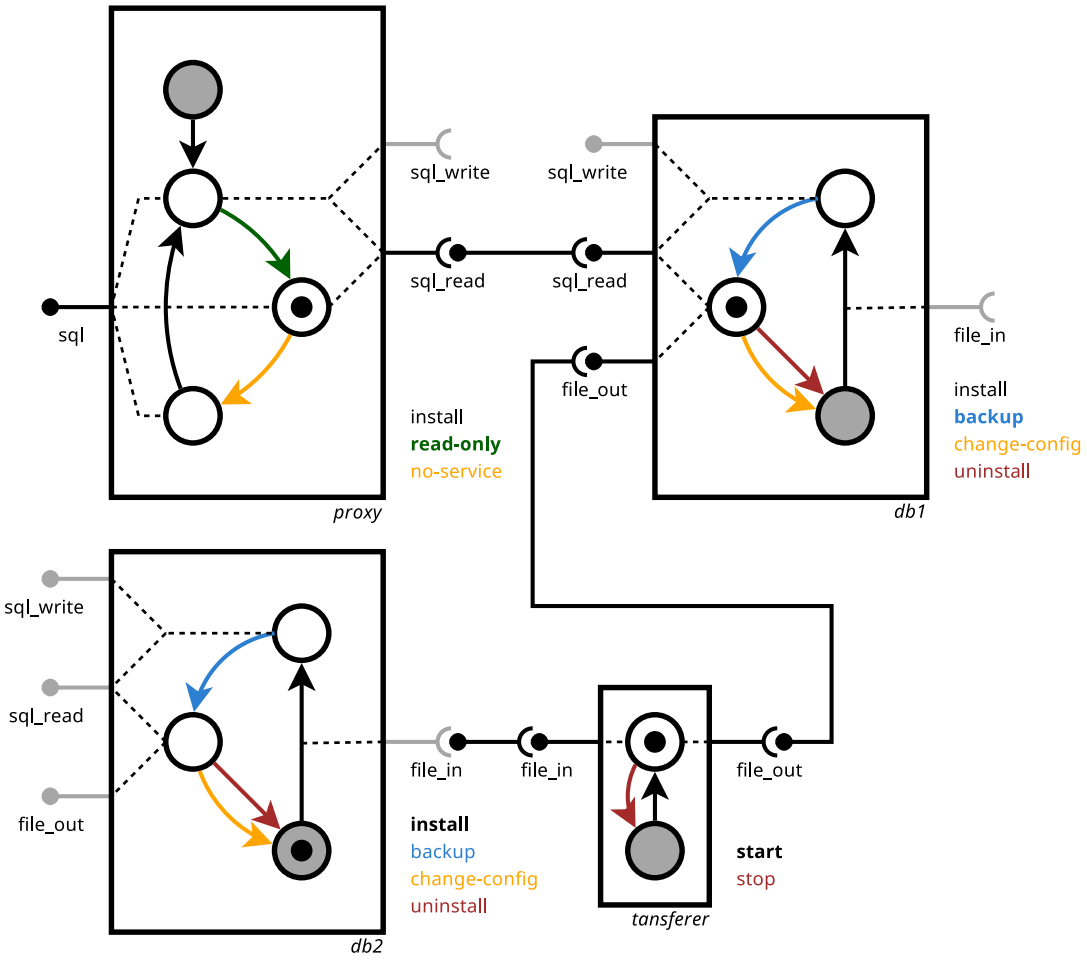
Behavioral interfaces







RECONFIGURATION EXECUTION




```
behaviorChange (proxy, read-only)
wait (proxy)
disconnect (sql_write, db1, proxy)
new (db2 : Db)
new (transferer : Transferer)
connect (file_out, db1, transferer)
connect (file_in, transferer, db2)
behaviorChange (transferer, run)
behaviorChange (db2, install)
behaviorChange (db1, backup)
wait (db2)
behaviorChange (proxy, no-service)
wait (proxy)
disconnect (sql_read, db1, proxy)
connect (sql_read, db2, proxy)
connect (sql_write, db2, proxy)
behaviorChange (proxy, install)
behaviorChange(transferer, stop)
wait (transferer)
behaviorChange(db1, uninstall)
wait (db1)
del (db1)
del (transferer)
```

CONCLUSION

Conclusion

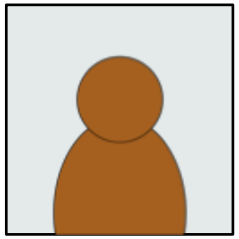
- Introduction of *reconfiguration* inside Madeus
- We expect good *performances*
- Under validation on real reconfiguration use-cases (OpenStack)

Perspectives

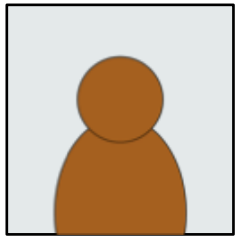
- Formalization of the *behaviors* and the reconfiguration language
- Equivalence proof for *behavioral interfaces*
- Increase reconfiguration capabilities

Maverick Chardet
PhD student

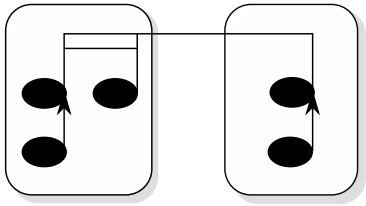
Christian Perez
Research director Inria



18 months postdoc



18 months engineer



VeRDi project
Pays de La Loire

THANK YOU !


Inria
INVENTEURS DU MONDE NUMÉRIQUE


IMT Atlantique
Bretagne-Pays de la Loire
École Mines-Télécom


LS2N
LABORATOIRE
DES SCIENCES
DU NUMÉRIQUE
DE NANTES