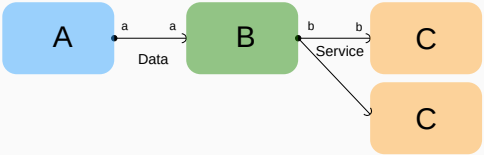
Integrated Model-checking for the Design of Safe and Efficient Distributed Software Commissioning

Hélène Coullon, associate prof. IMT Atlantique, Inria, LS2N - Nantes France
Claude Jard, prof. Université de Nantes, LS2N - Nantes, France
Didier Lime, associate prof. Ecole Centrale, LS2N - Nantes France

2019-12-04 iFM'19 Bergen Norway

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

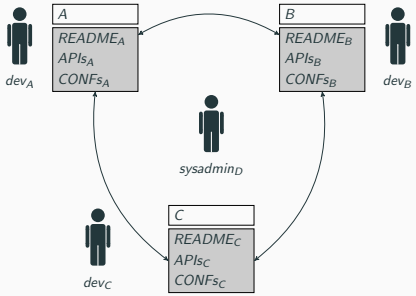
Code (development) ✓

- modular code and communications
- software engineering pratices (composition, code reuse, etc.)

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



- 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

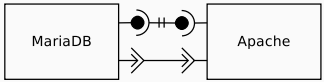
- Languages and models for distributed software commissioning
  - Software Engineering (SE) properties: composition, code-reuse, separation of concerns etc.
- Safety of distributed software commissioning
- Efficiency of distributed software commissioning

Table of contents

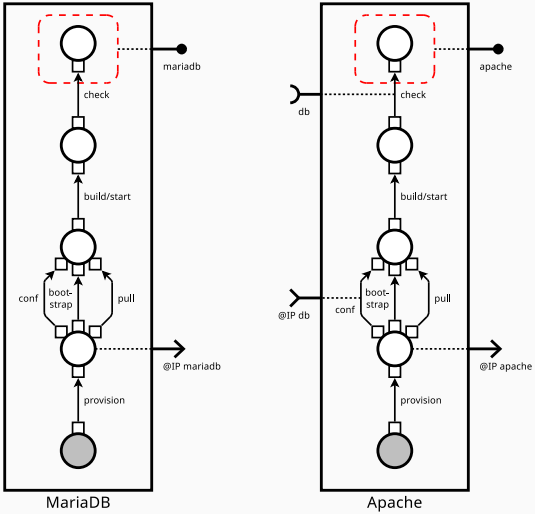
Background - the Madeus model

Madeus (1/2)

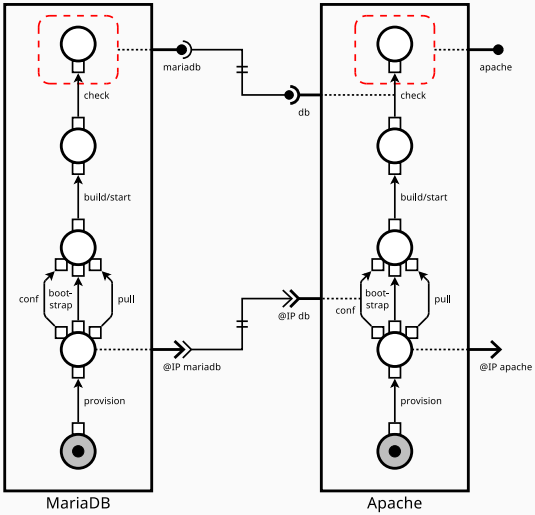
- $\text{dev}_{\text{apache}}$ and dev_{mdb} design their respective component commissioning
- sysadmin connect compatible ports of components to build the overall commissioning of Apache/MariaDB
- separation of concerns: $\text{dev}_{\text{assembly}}$ does not need to know details about each component
- automatic coordination, correct order of execution, automatic parallelism



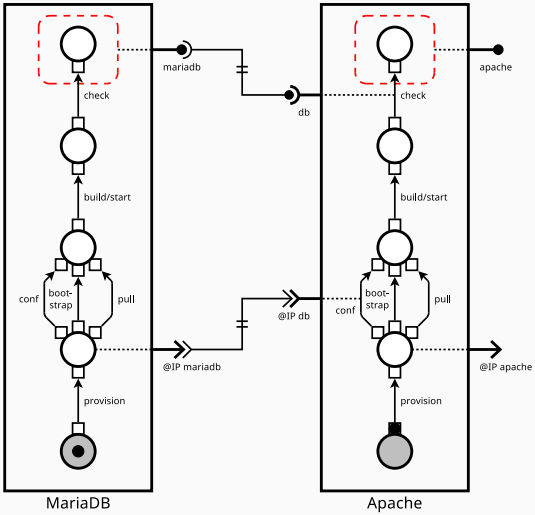
Madeus (2/2)


mariadb
check
build/start
conf
boot-
strap
pull
@IP mariadb
provision
MariaDB
apache
db
check
build/start
@IP db
conf
boot-
strap
pull
@IP apache
provision
Apache

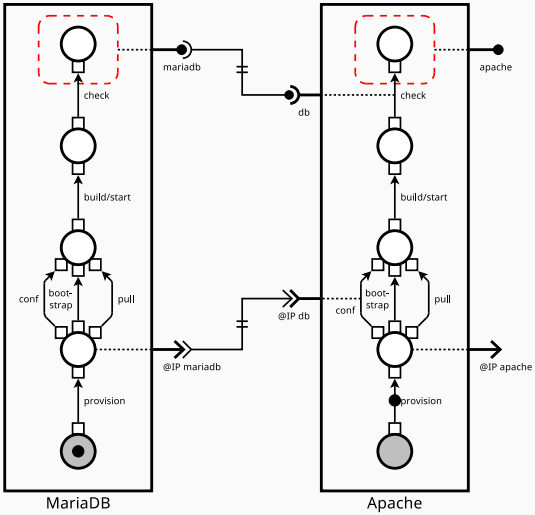
Madeus execution



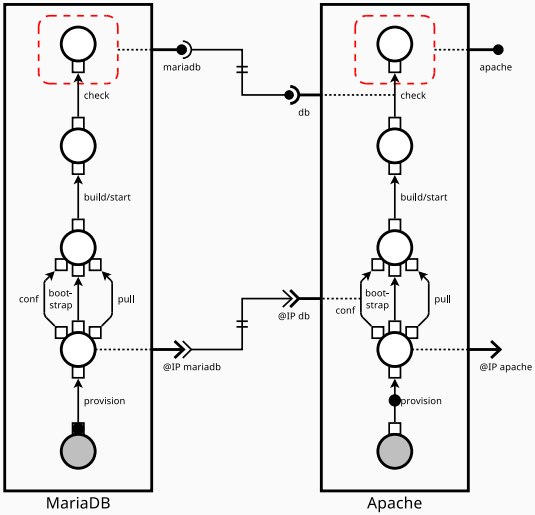
Madeus execution



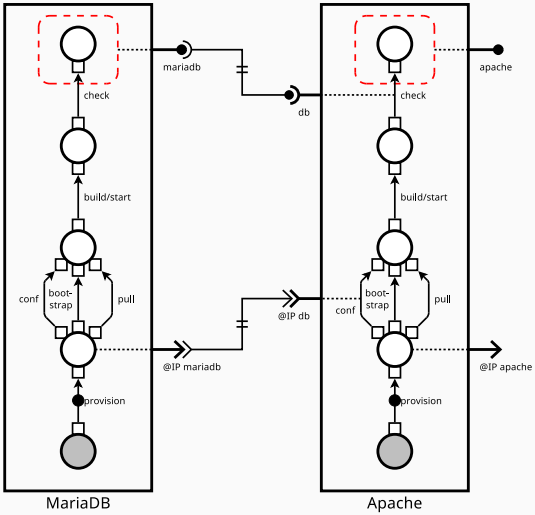
Madeus execution



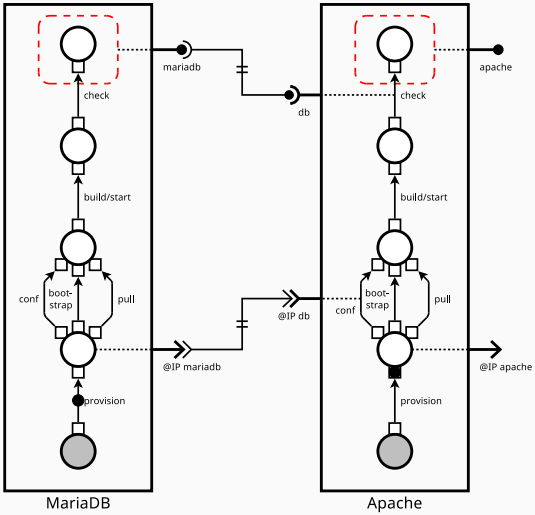
Madeus execution



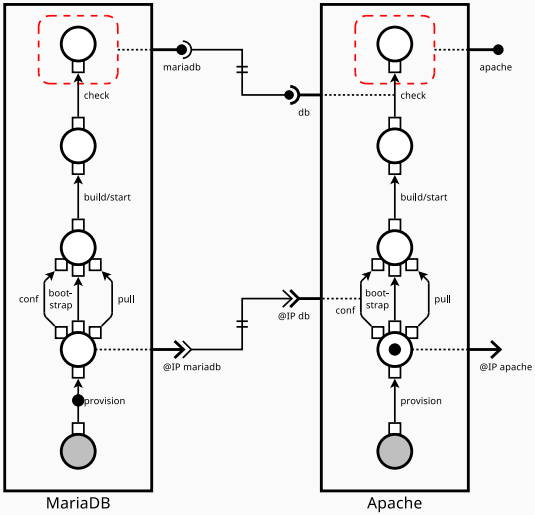
Madeus execution



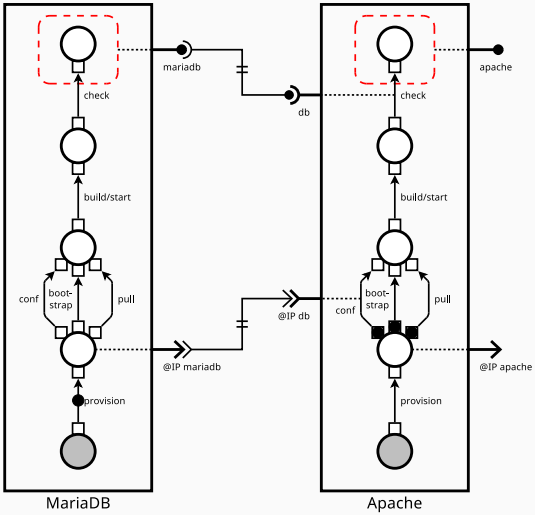
Madeus execution



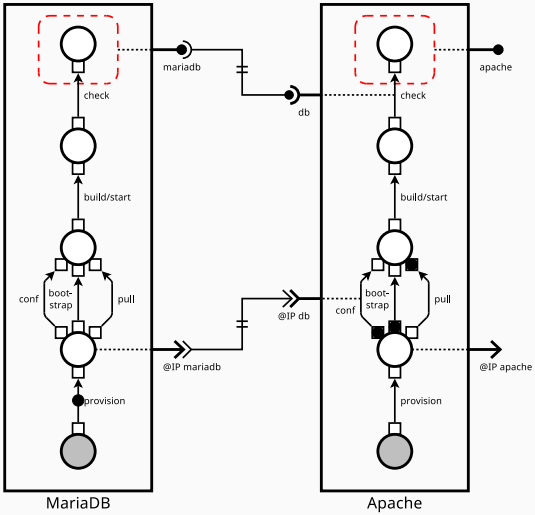
Madeus execution



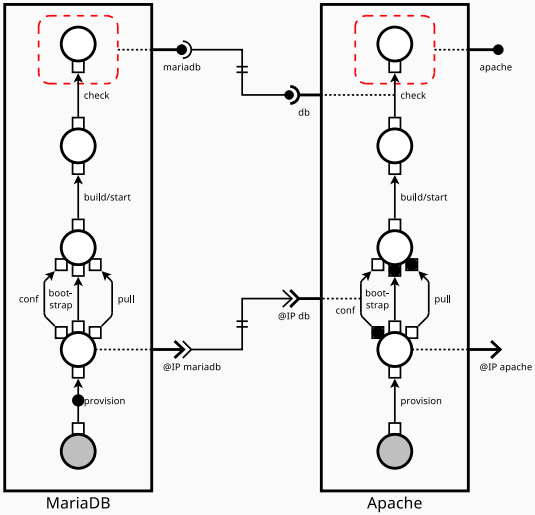
Madeus execution



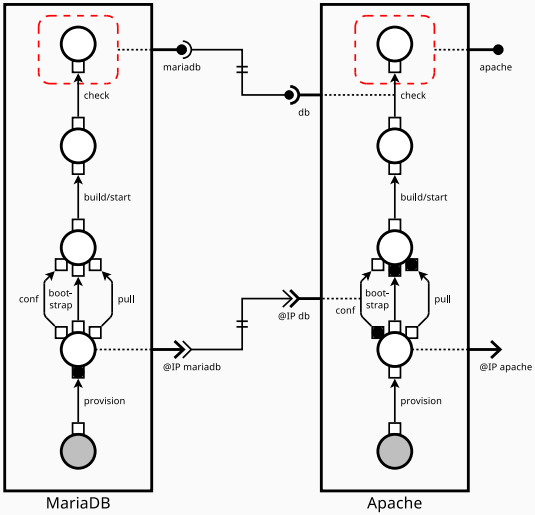
Madeus execution



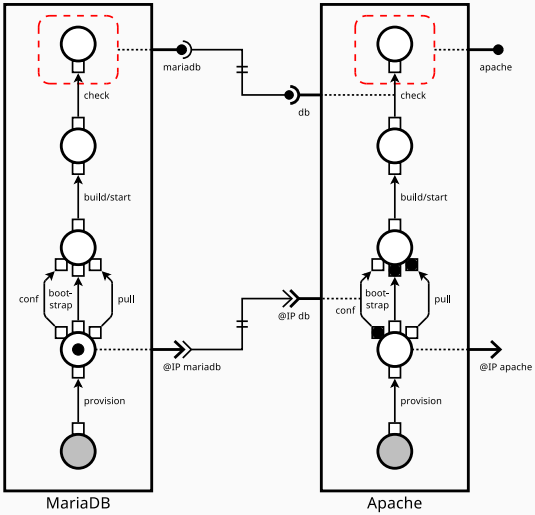
Madeus execution



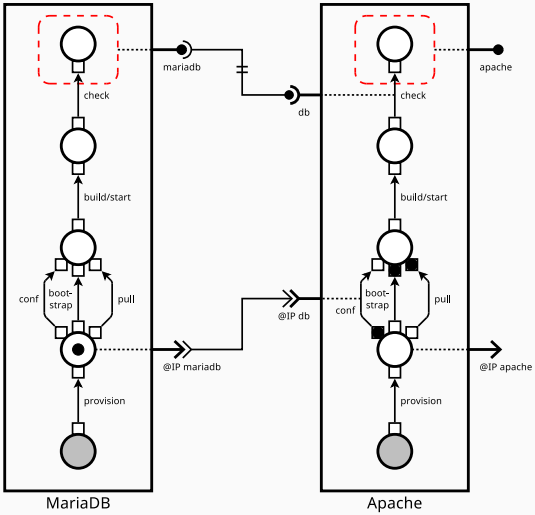
Madeus execution



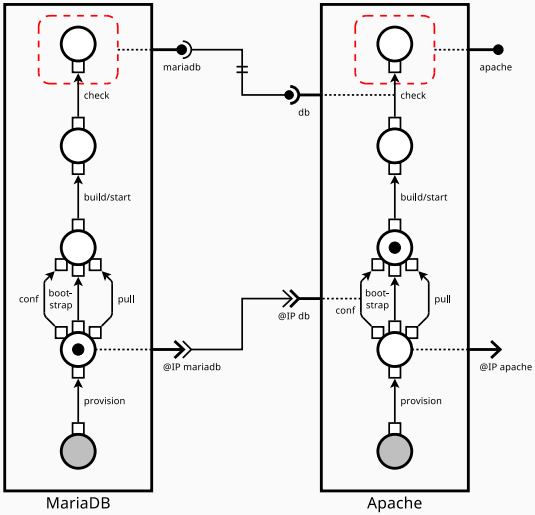
Madeus execution



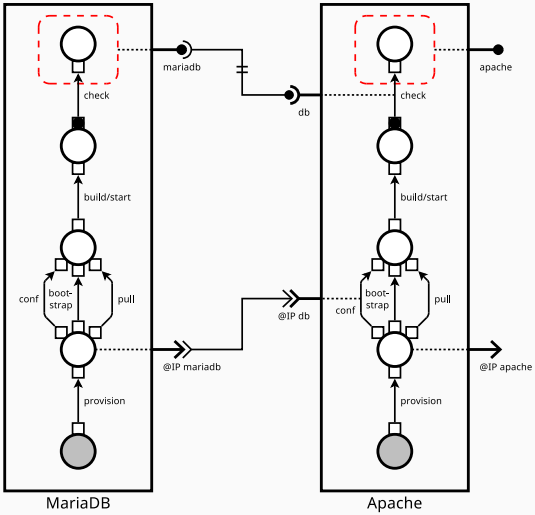
Madeus execution



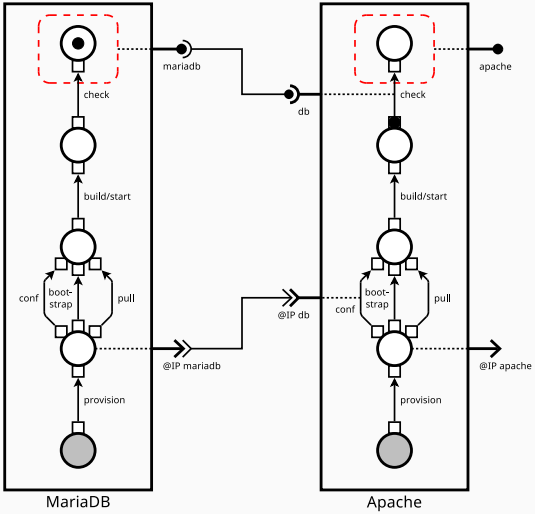
Madeus execution



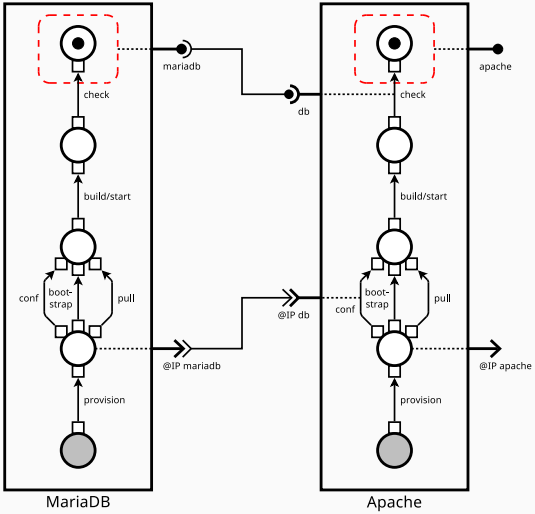
Madeus execution



Madeus execution

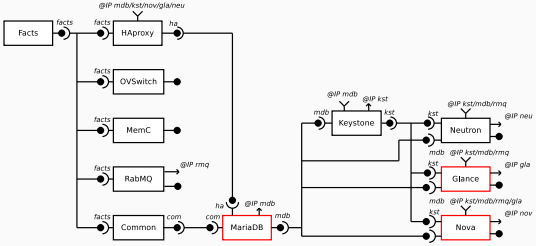


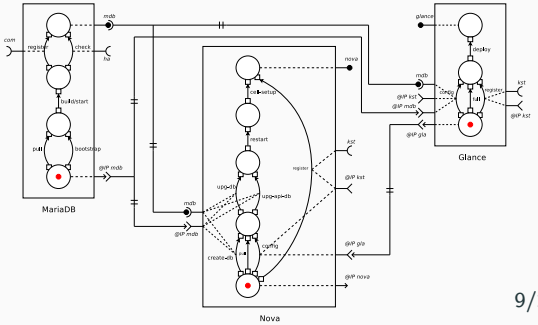
Madeus execution



Few results

- "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).
- up to 60% performance gain compared to Kolla-ansible





Safe and Efficient Software Commissioning with MADA

MADA

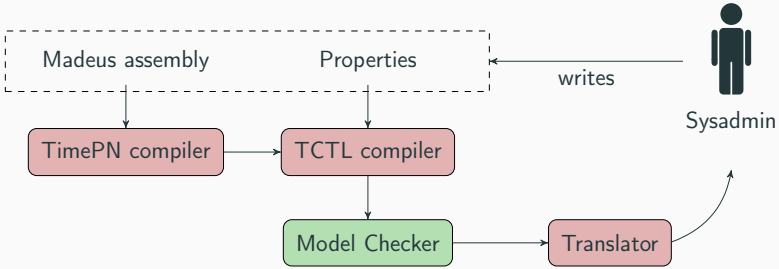
Hypothesis: *the commissioning of a distributed software system already exists and the developer wants to use Madeus to enhance its efficiency*

1. how to divide existing intricate commissioning scripts in interesting subtasks to introduce parallelism?
2. how to find correct dependencies between commissioning tasks?
3. how to avoid safety issues such as deadlocks, wrong order of configurations?

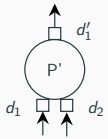
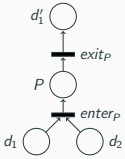
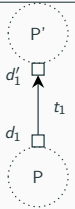
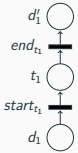
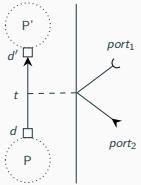
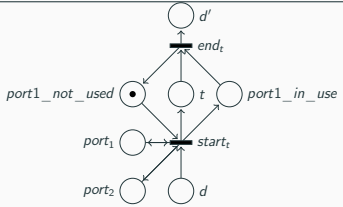
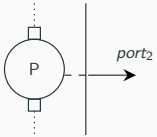
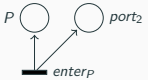
Goal

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

Overview

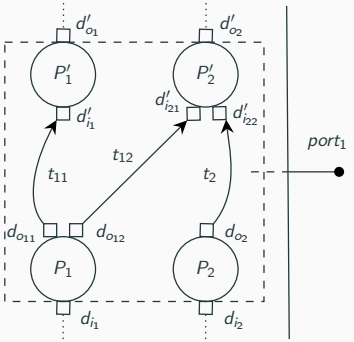

Madeus assembly
Properties
writes
Sysadmin
TimePN compiler
TCTL compiler
Model Checker
Translator

Time Petri Nets (1/2)

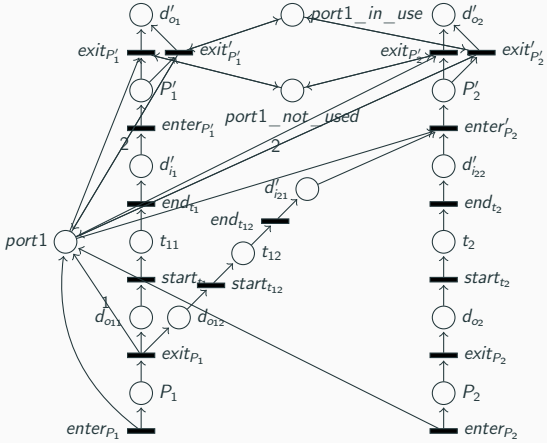
Madeus (1)	Petri net (1)	Madeus (2)	Petri net (2)
			
Madeus (3)	**Petri net (3)**	**Madeus (4)**	**Petri net (4)**
			

Time Petri Nets (2/2)

Madeus (5)



Petri net (5)



Properties (1/2)

- 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)
```

Properties (2/2)

HALP automatically transformed to TCTL (Time Computational Tree Logic) formulae

Qualitative properties

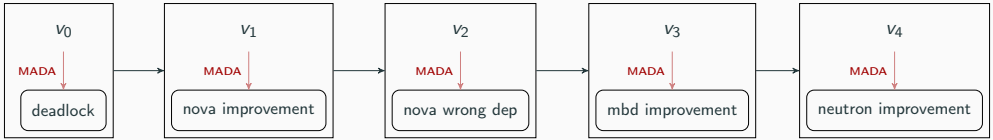
- deployability $\longrightarrow$ inevitability
- sequentiality $\longrightarrow$ observer subnet + invariant
- forbidden $\longrightarrow$ observer subnet + invariant

Quantitative properties

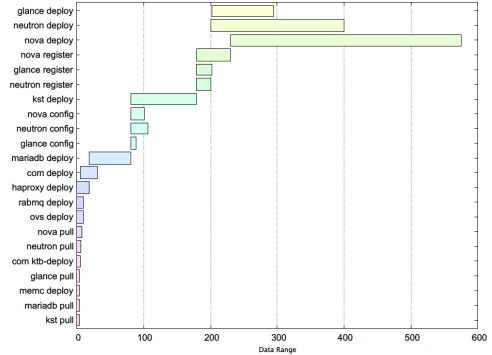
- parallelism $\longrightarrow$ max($\sum$(reachable markings))
- gantt boundaries: min/max costs + causality in the trace to get the critical path

Evaluation (1/4)

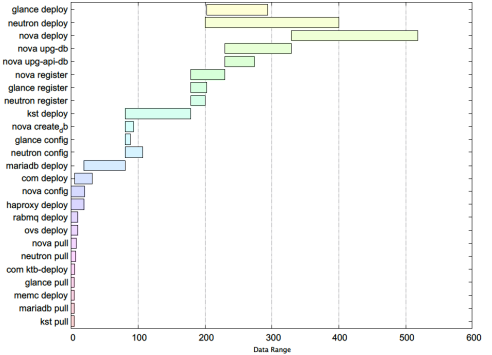
5 versions of the OpenStack commissionining successively enhanced with MADA



Evaluation (2/4)

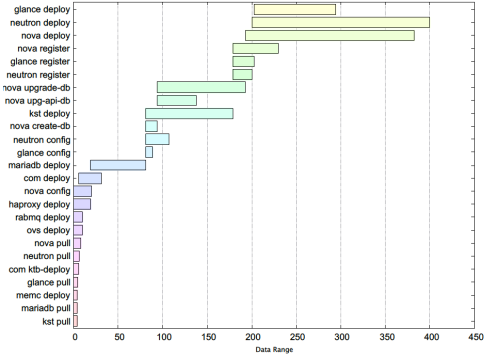


(a) 1-naive with critical path: *nova deploy, nova register, kst deploy, mariadb deploy, haproxy deploy*

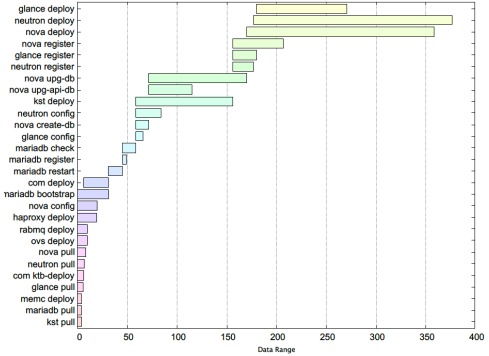


(b) 2-nova with critical path: *nova deploy, nova upg-db, nova register, kst deploy, mariadb deploy, haproxy deploy*

Evaluation (3/4)



(c) 3-nova with critical path: *neutron deploy, neutron register, kst deploy, mariadb deploy, haproxy deploy*



(d) 4-nova-mdb with critical path: *neutron deploy, neutron register, kst deploy, mariadb check, mariadb restart, mariadb bootstrap*

Evaluation (4/4)

Experiments conducted with the model checker Romeo

	0-deadlock	1-naive	2-nova	3-nova	4-nova-mdb
Madeus places	27	27	28	28	29
Madeus transitions	22	22	25	25	28
Madeus connections	30	30	30	30	30
Petri net places	113	113	124	124	134
Petri net transitions	75	75	84	84	92
Transformation time (ms)	1.6	1.6	1.8	1.7	1.5
Deployability	False	True	True	True	True
Resolution time (s)	0	41.6	78.7	88.7	152.6
Parallelism nova	-	1	2	2	2
Resolution time (s)	-	42.1	82.7	93.6	154.3
Parallelism full	-	10	11	11	12
Resolution time (s)	-	43.2	86.1	98.4	162.9
Gantt & critical path	-	Fig	Fig	Fig	Fig
Resolution time (s)	-	130.1	266.9	275.4	588.1
Boundaries	-	[575,615]	[518,554]	[400,423]	[377,398]
Resolution time (s)	-	130.1, 128.8	266.9, 269.7	275.4, 267.6	588.1, 580.8

Discussion

- Separation between the design of a Madeus commissioning and the real set of commissioning commands with its associated technical issues
- Same critical paths found on real experiments on Grid'5000
- No need for precise intervals but good order of magnitude between transitions
  - errors introduced on intervals
  - up to 95% of errors on short tasks, 40% of errors on bigger tasks
  - keep the same order of magnitude

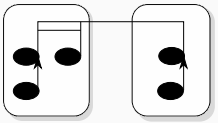
Conclusion and perspectives

Conclusion

- background on the Madeus model
  - separation of concerns
  - efficiency
- MADA to help in the design of Madeus commissionings
  - automatic transformation to equivalent Time Petri nets
  - high abstraction level properties for the user
  - qualitative and quantitative properties
  - evaluated on a real use-case

Perspectives

- Concerto: *A generalization of Madeus to dynamic reconfiguration of ditributed software systems.* Phd student: Maverick Chardet
- VeRDi: *Verified Reconfiguration Driven by execution*
  - Simon Robillard (Postdoc): SMT solvers for verification and synthesis of reconfiguration scripts in Concerto



Thank you !

Questions?