Robust Reconfiguration of Cloud Applications

Francisco Durán

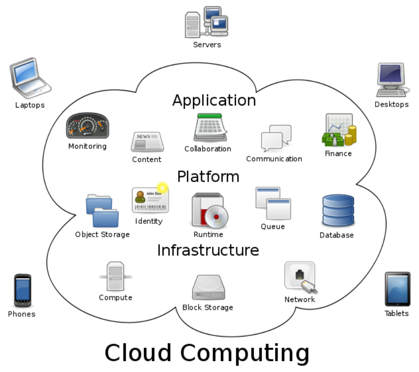
University of Málaga, Spain

Gwen Salaün

Grenoble INP, Inria, France

Introduction

- Cloud computing aims at delivering resources and applications as a service over a network (*e.g.*, the Internet)



- Cloud applications are often complex distributed applications composed of multiple software running on separate virtual machines

- Setting up, (re)configuring, and monitoring these applications are difficult tasks, and involve complex management protocols

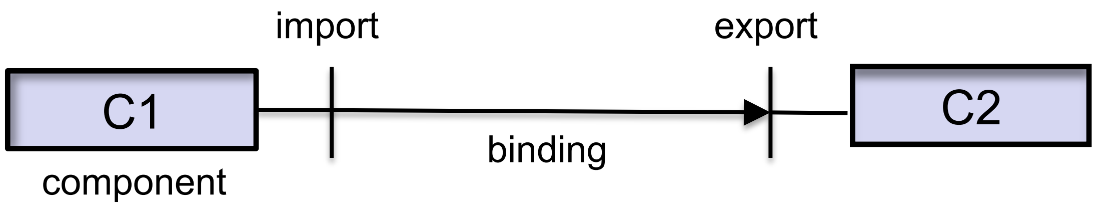
- In this talk, we present an innovative protocol, which automates the reconfiguration of component-based systems running over several virtual machines

Outline

1. Reconfiguration Protocol
2. Verification with Maude
3. Concluding Remarks

Application Model

- An application model consists of a set of components and a set of bindings connecting these components together
- A component is composed of input and output ports, namely imports and exports
- An import can be either optional or mandatory
- A binding connects an import of one component to an export of another component



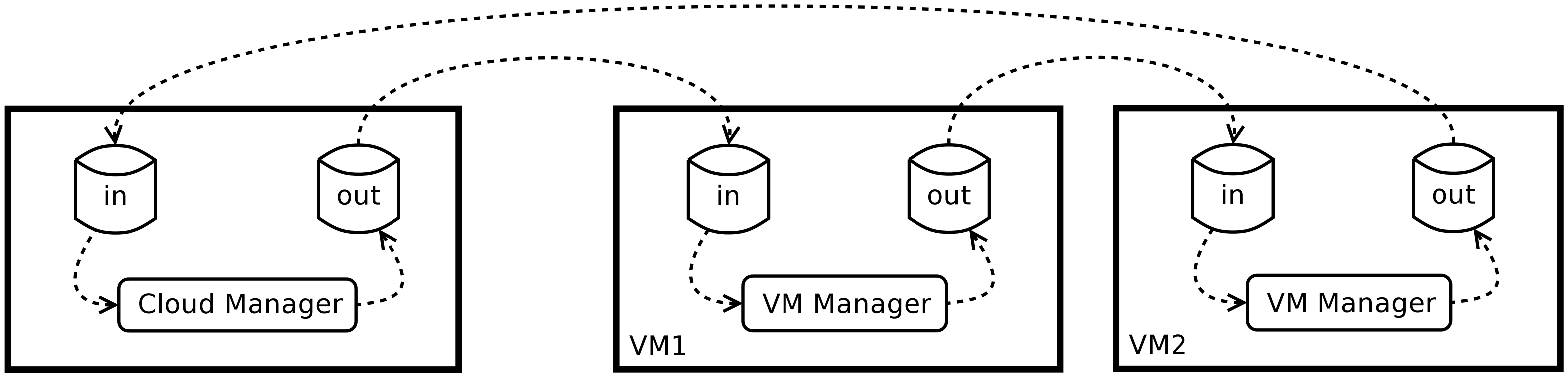
- Components are distributed over separate virtual machines

Participants

- The cloud manager (CM) guides the reconfiguration by instantiating VMs and posting reconfiguration operations

- Each VM is equipped with a VM manager in charge of (dis)connecting ports and starting/stopping components

- Communications between CM/VM and VMs are handled via FIFO buffers

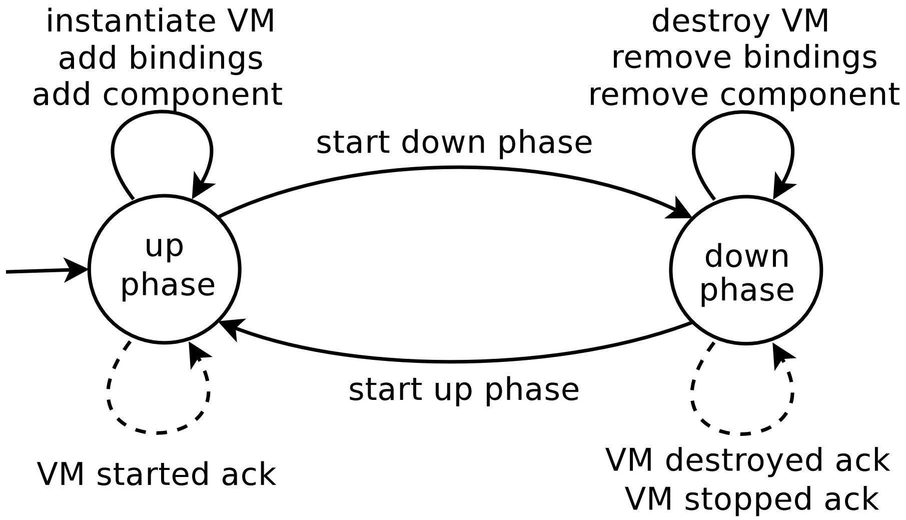


Protocol Features

- All reconfiguration tasks are automatically achieved by cloud / VM managers
- VM managers are in charge of starting/stopping their own components in a decentralized manner (no centralized manager)
- The protocol is also loosely-coupled because each VM manager does not have a global view of the current state of the application (other VMs)
- The protocol is robust: during its application, some important architectural invariants are preserved, *e.g.*, all mandatory imports of a started component are connected to started components

Cloud Manager

- The CM submits reconfiguration operations to the running application and keeps track of the state of the deployed VMs
- Reconfiguration operations: instantiation/destruction of a VM, addition/removal of a component on/from an existing VM, and addition/suppression of bindings
- The CM applies successively up and down phases, *e.g.*, a down phase involves shutdown operations only (VM shutdown, binding removal, ..)



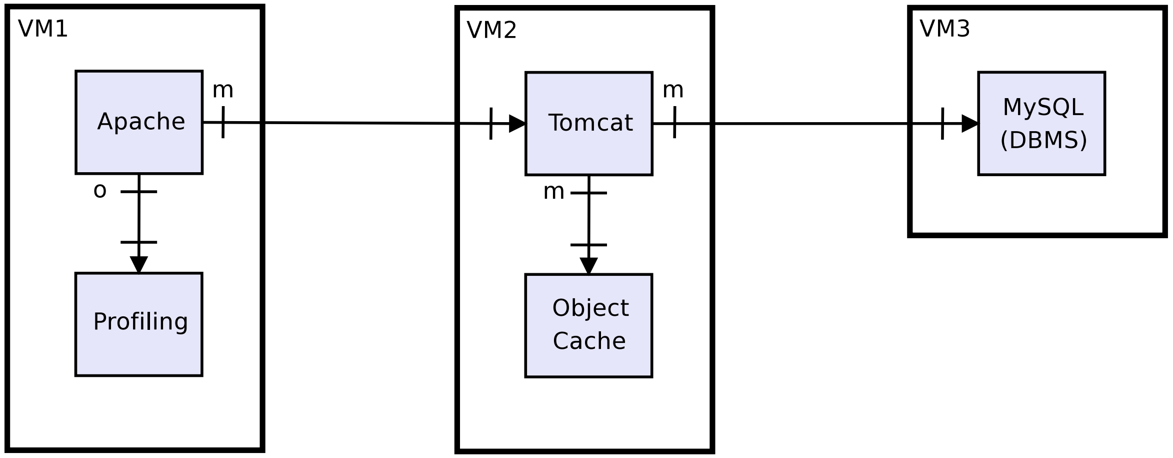
VM Instantiation

- When a VM is instantiated, the VM manager is in charge of starting all the components
- A component without imports or optional imports only can be started immediately
- A VM manager sends a binding message (IP, port, etc.) to each VM with a component that requires a connection to an export
- A VM manager sends a start message to the partner VM when it starts a local component
- A component can be started when all its mandatory imports are bound to started components

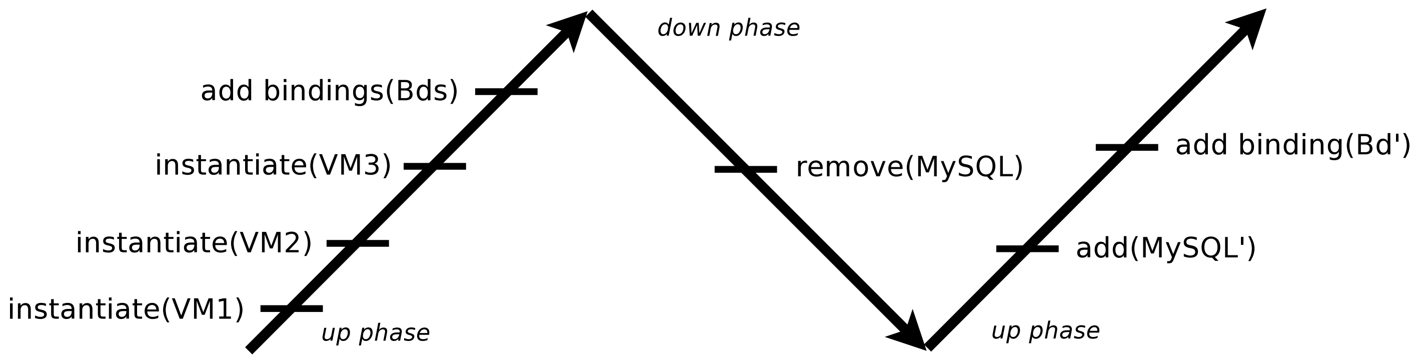
VM Destruction

- All components on a VM to be destroyed need to be properly stopped as well as all components bound on them through mandatory imports
- A component that does not provide any service can be immediately stopped
- Shutting down a component implies a backward propagation of "*ask to unbind*" messages
- A forward propagation of "unbind confirmed" messages lets the components know that disconnection has been achieved
- When a component has received such messages for all components using that component on mandatory imports, it can stop itself

Reconfiguration Scenario (1/3)

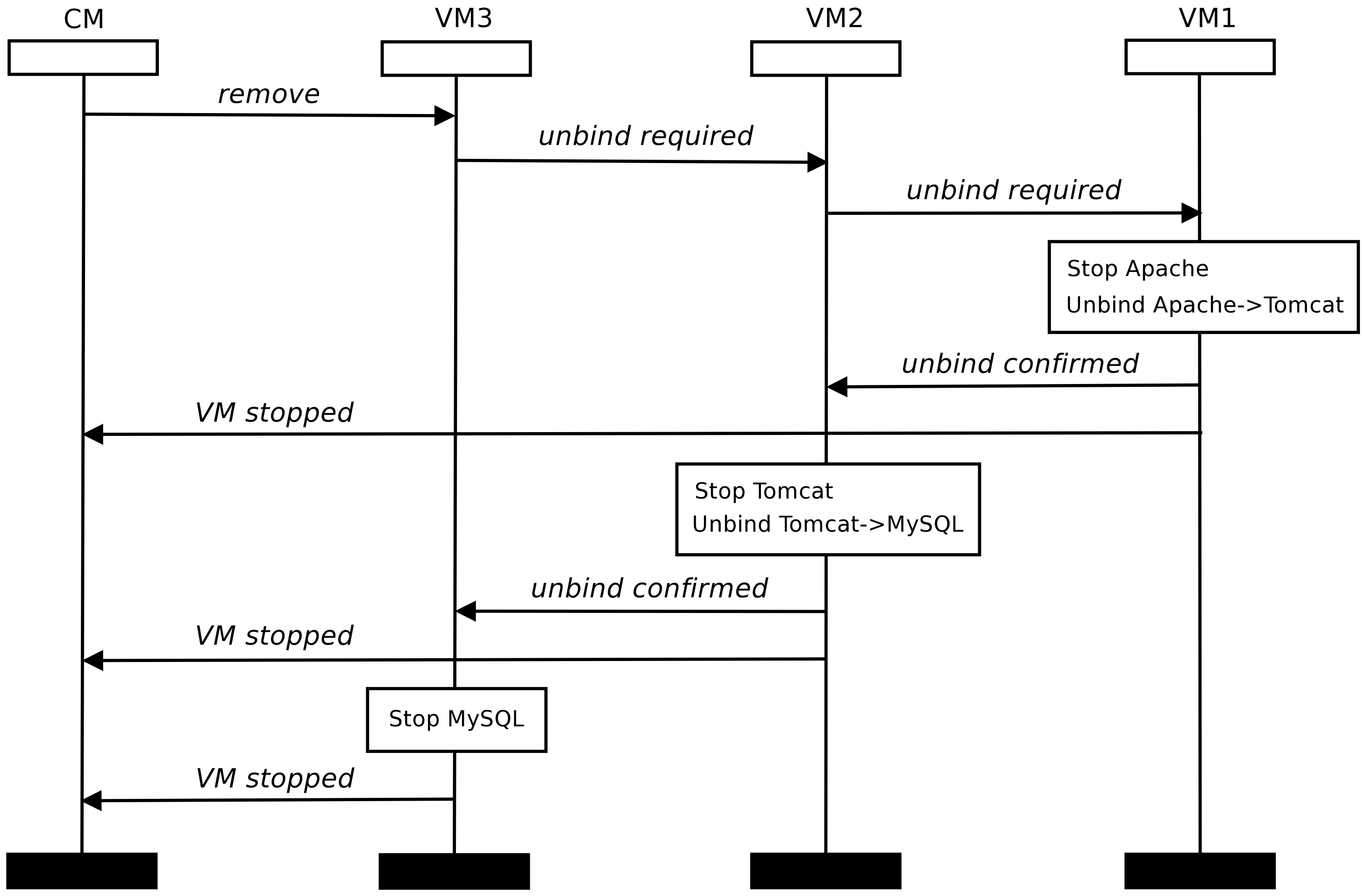


A 3-tier Web Application

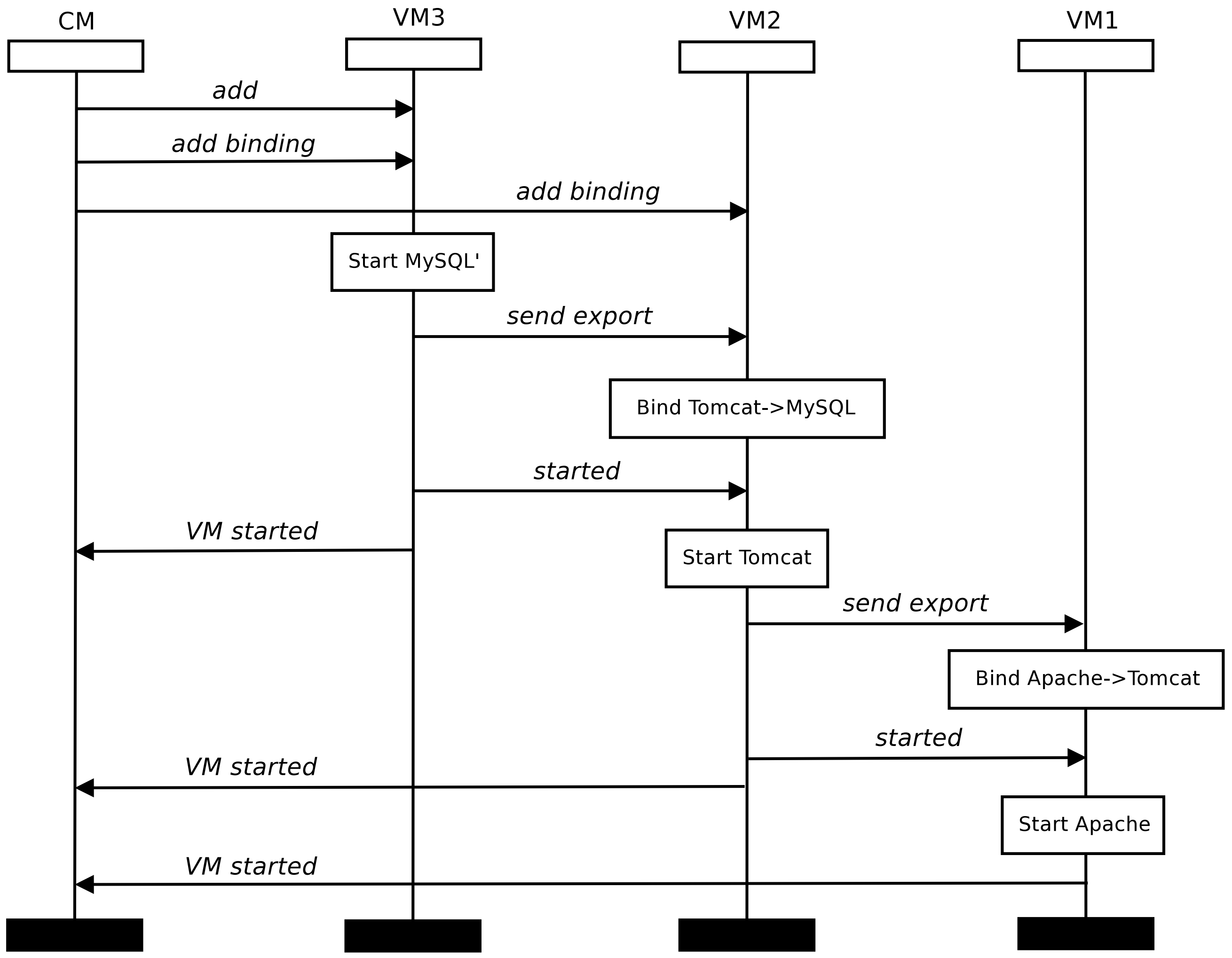


Up/down Scenario

Reconfiguration Scenario (2/3)



Reconfiguration Scenario (3/3)



Outline

1. Reconfiguration Protocol
2. Verification with Maude
3. Concluding Remarks

Specification and Verification

- We specified the models and protocol in Maude; we defined 17 rules for the start-up process and 27 rules for the shutdown process
- Simulation, reachability analysis, and model checking were very helpful for identifying and fixing several bugs
- We identified 12 key properties that the protocol must respect during any step of its application, *e.g.*, *"a VM being destroyed eventually succeeds in stopping all its components"*
- Experiments on more than 300 examples (application model and reconfiguration scenario), representing typical n-tier Web applications
- Found issues: simple errors, introduction of up/down phases, double propagation for stopping properly components

Outline

Concluding Remarks

- We have presented a robust protocol for dynamically reconfiguring cloud applications involving components distributed over several VMs

- The use of formal methods helped to detect and correct several bugs during the protocol design

Perspectives

- Improvement of the protocol to avoid up/down phases: non-trivial change since start and stop messages may be unwillingly mixed up

- Extending the protocol to take VM failures into account: this implies restoring a consistent state for the application and possibly repairing it

- Ongoing implementation by our colleagues from Orange Labs (OpenCloudware Project)