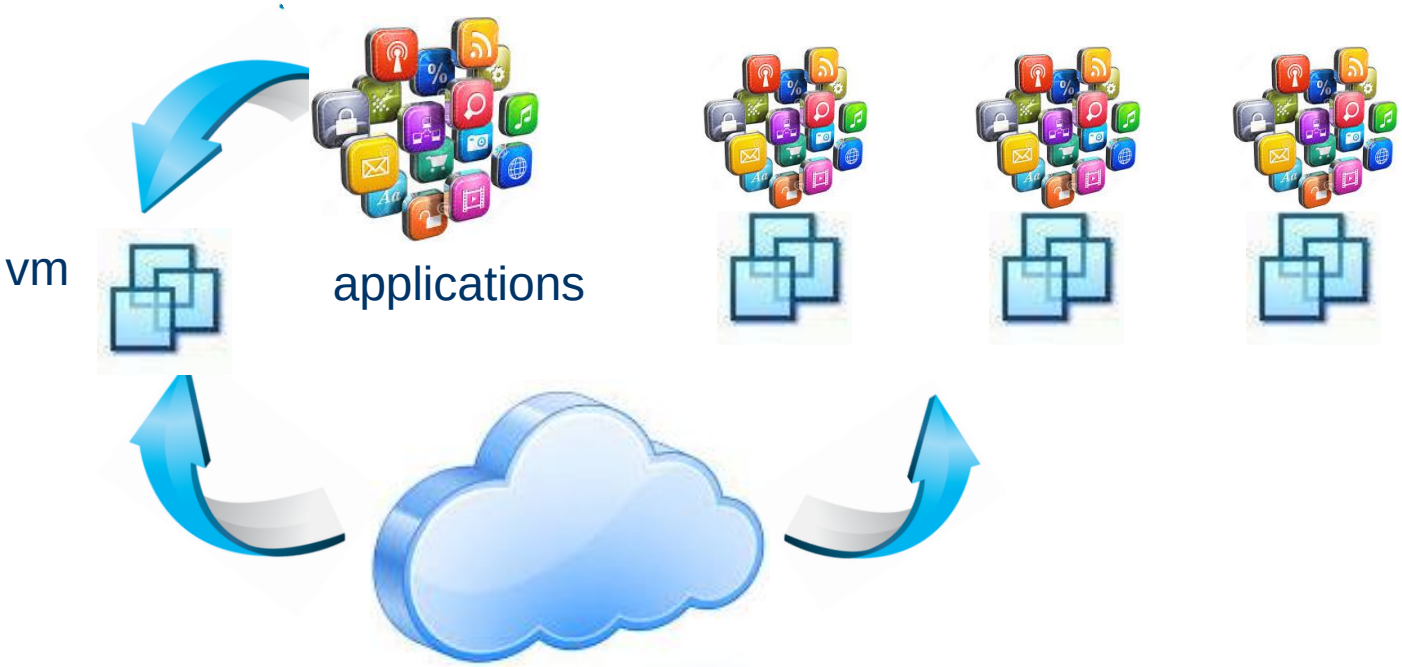
Modelling Adaptation Policies as Domain-Specific Constraints

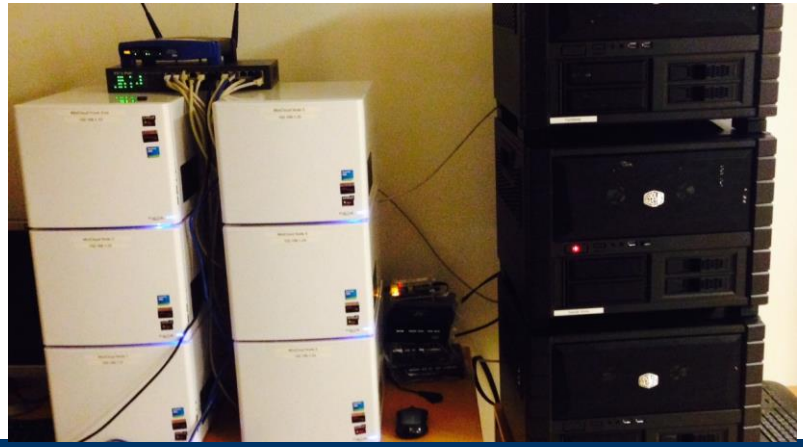
Hui Song, Xiaodong Zhang, Nicolas Ferry,
Franck Chauvel, Arnor Solberg, Gang Huang

SINTEF ICT, Oslo, Norway

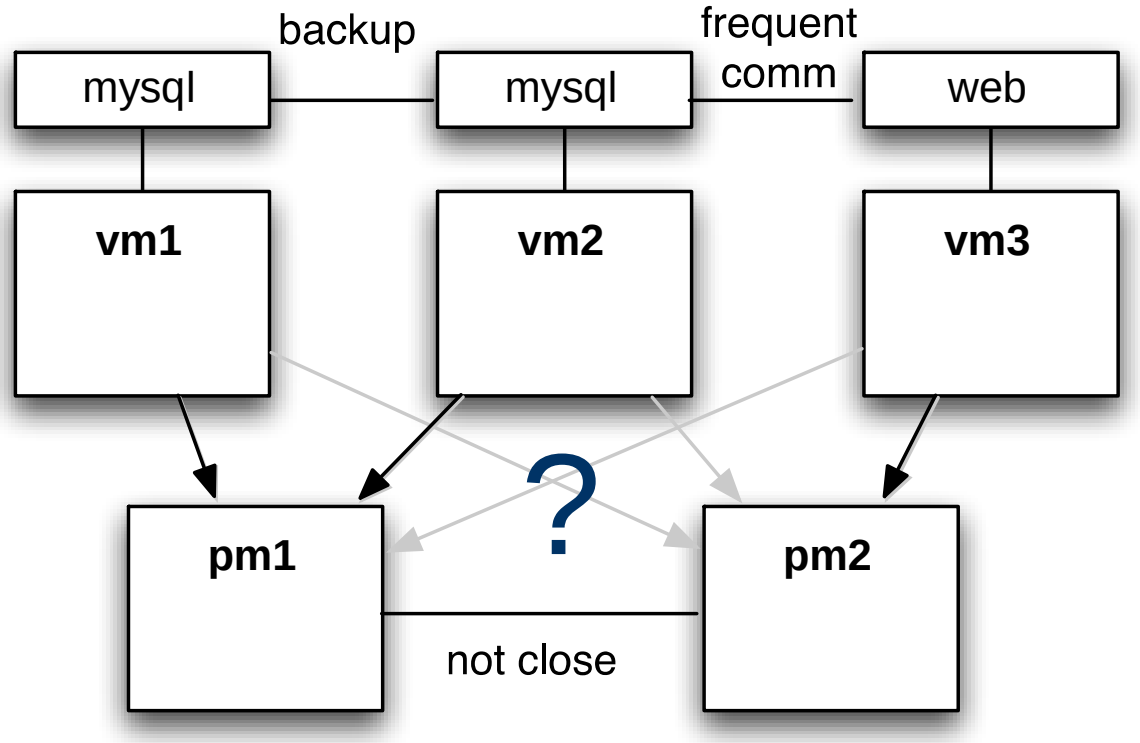
Peking University, Beijing, China

VM Placement in Cloud



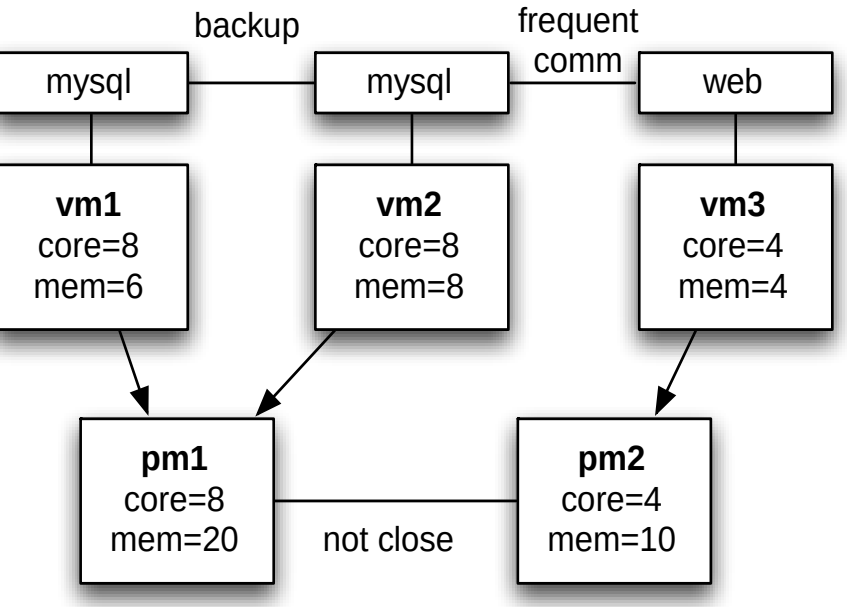


Thinking in Models



- Resource limitation
- Consolidation
- Backup split
- Frequent close
- Migration cost

Developing the Adaptation Behaviour



- Concentrate VMs to fewer pms to save energy
  - but only when the sum of VMs' memory does not exceed the pms' memory
  - Don't move very big VMs
- Separate backup VMs to different PMs
  - But only when the sum of VMs' memory…
  - Don't move very big…
  - Concentrate VMs…, when possible
- Put frequently communicating VMs closer
  - Separate backup VMs…
    - But only when the sum…
    - Don't move very big…
  - Concentrate…
  - …

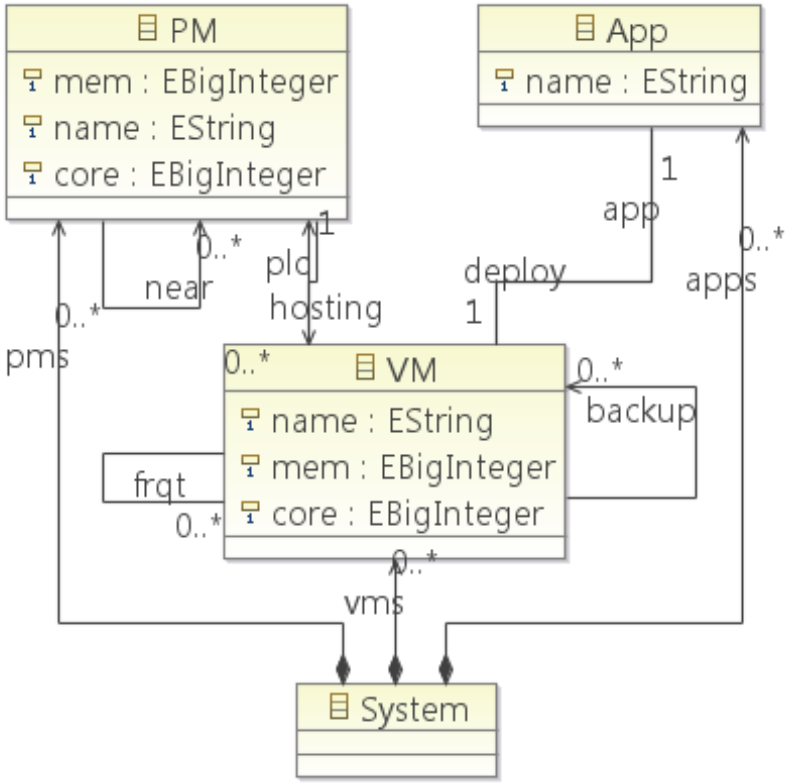
```
if backup(vm1,vm2) and vm1.host = vm2.host,
  if vm1.mem < vm2.mem or ( frqt(vm1, vm2) and not vm1.mem >> vm2.mem )
    if vm2.mem < pm2.available and vm1.core <= pm2.core
      move vm1 to pm2
    else
      …
```

If written in Action-based adaptation policy

Modelling Adaptation Policies

- Challenges
  - Many interrelating concerns
- Actions policies (if-then-else or event-condition-action)?
  - Explosion of branches
  - Hard to introduce new concerns
  - Abstraction gap between concerns and actions
- New way of modeling!
  - Just write down the constraints themselves
    - "what the system should be like" rather than "how to achieve that”
  - Potential conflicts?
    - Soft constraints with different weights

Modeling Adaptation Policies as Constraints



Memory Limitation

context PM **inv**:
hosting->collect(mem)->sum() <= mem
(priority: mandatory)

Consolidation

context PM **inv**: **self**.hosting->size() = 0
(priority: low)

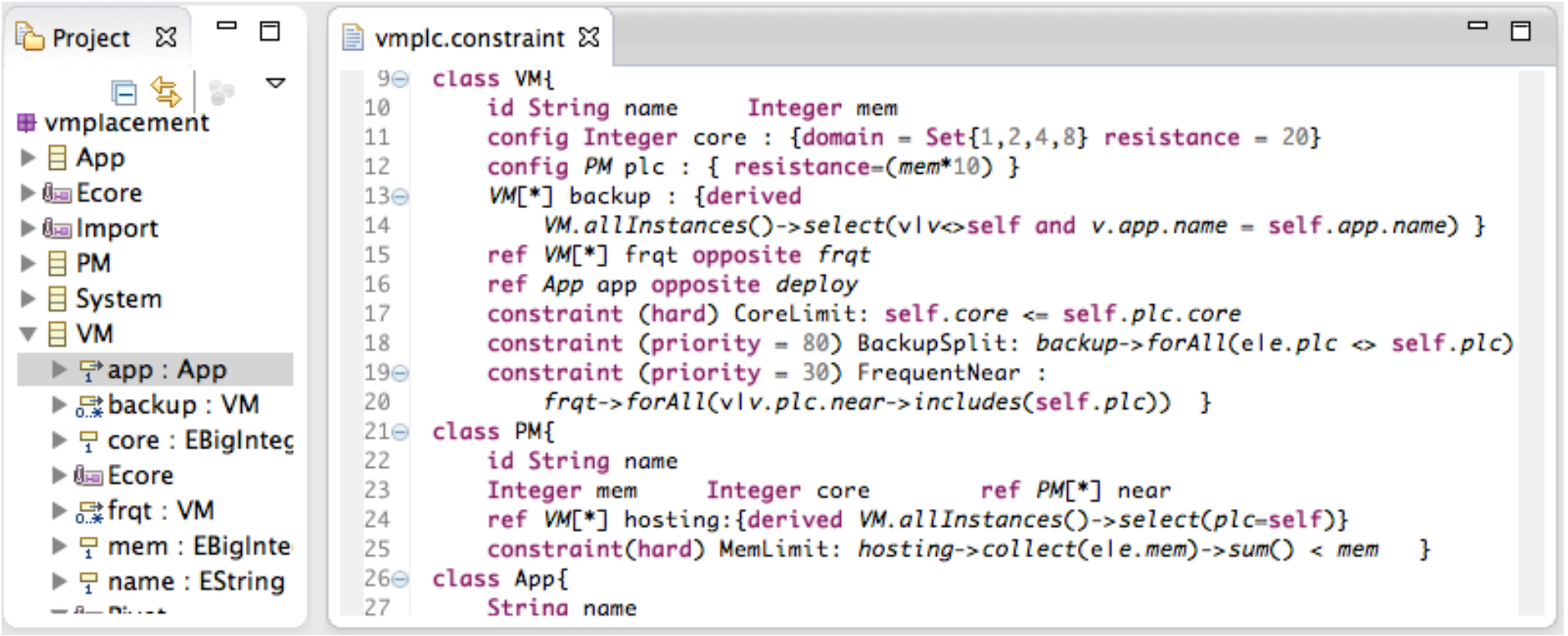
Backup split:

context VM **inv**:
backup->forall(e|e.plc != self.plc)
(priority: high)

Migration cost:

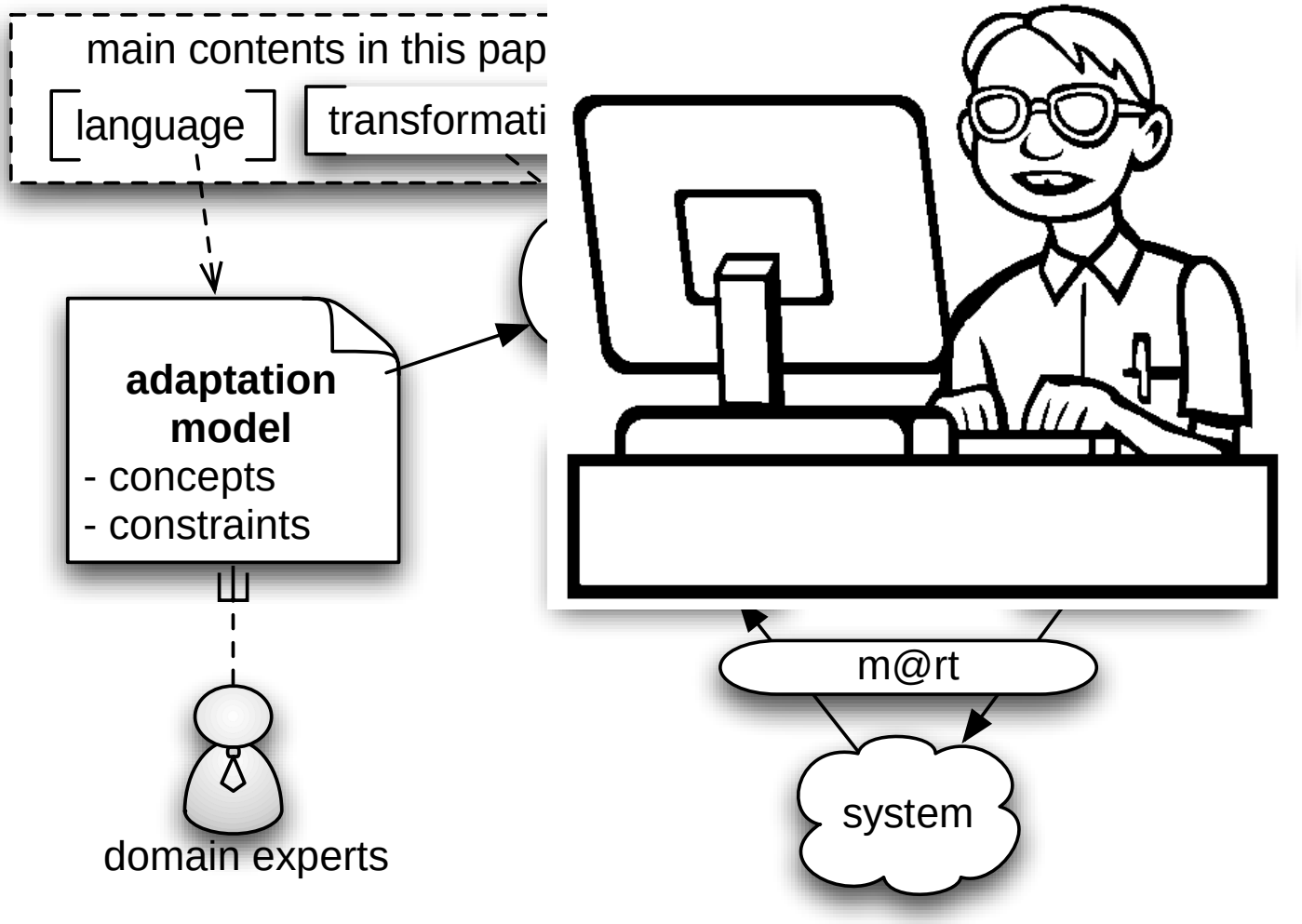
vm1.plc = pm1 (priority: 8)
Vm3.plc = pm2 (priority: 4)

Constraint Modelling Language and Editor

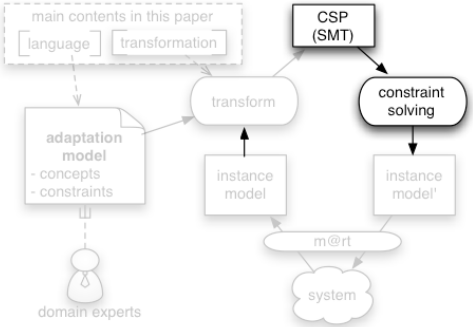


https://bitbucket.org/huis/constraintml

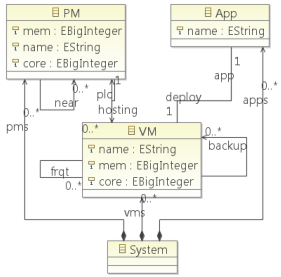
Adaptation Based on Constraints



Satisfactory Modulo Theory



Memory Limitation
context PM **inv**:
hosting->collect(mem)->sum() <= mem



$$\text{VM} : \{vm_1, vm_2, vm_3\},\ \text{PM} : \{pm_1, pm_2\},\ \text{int},\ \text{boolean}$$

$$vc : \text{VM} \to \text{int},\ pc : \text{PM} \to \text{int},\ vmem : \text{VM} \to \text{int},\ pmem : \text{PM} \to \text{int}$$

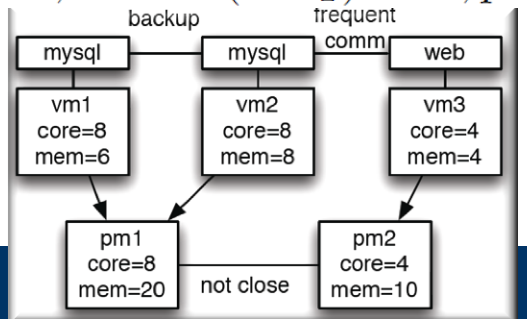
$$plc : \text{VM} \to \text{PM},\ frqt : \text{VM} \times \text{VM} \to \text{bool},\ near : \text{PM} \times \text{PM} \to \text{bool}$$

$$\forall vm, pm.\ (plc(vm) = pm \Rightarrow vc(vm) \leq pc(pm))$$

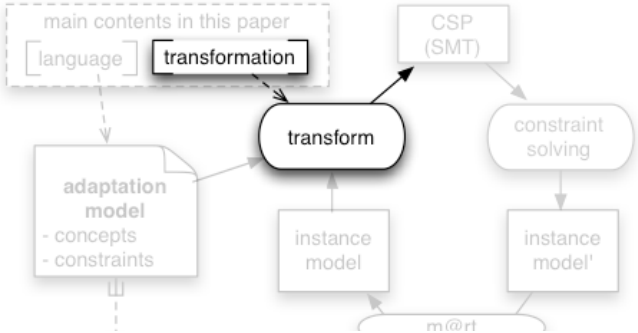
$$\forall pm. \left(\sum_{vm \in VM} \text{ite}(plc(vm) = pm, vmem(vm), 0)) \leq pmem(pm) \right)$$

$$\forall vm, vm'.\ (frqt(vm, vm') \Rightarrow near(plc(vm), plc(vm'))), \forall pm.\ (near(pm, pm))$$

$$vc(vm_1) = 8, vmem(vm_1) = 6, plc(vm_2) = pm_1, frqt(vm_2, vm_3) \ldots$$



Transformation



- OCL expressions
  - Principle #1: values, elements and most operations keep the same
  - Principle #2: replace property reference by function call
    - vm1.plc = pm1 => $plc(vm_1) = pm_1$
    - vm1.frqt = $\{frqt(vm_1, vm_1)?\, vm_1 : \bot, frqt(vm_2, vm_1)?\, vm_2 : \bot \cdots\}$
  - Principle #3: properties that do not change will be resolved
    - vm1.backup->forall(e|e.plc!=vm1.plc) => $plc(vm_1)! = plc(vm_2)$
  - Principle #4: complex operations have simple alternatives
    - vm1.frqt.forall(e.x) => $(frqt(vm_1, vm_1)?\, x(vm_1) : True)\ \&\ (frqt(vm_2, vm_1)?\, x(vm_2) : True\ \&\ \ldots$
- A new set of partial-evaluation semantic rules on OCL
  - "*which this slide is too narrow to contain*"

Constraint Solving

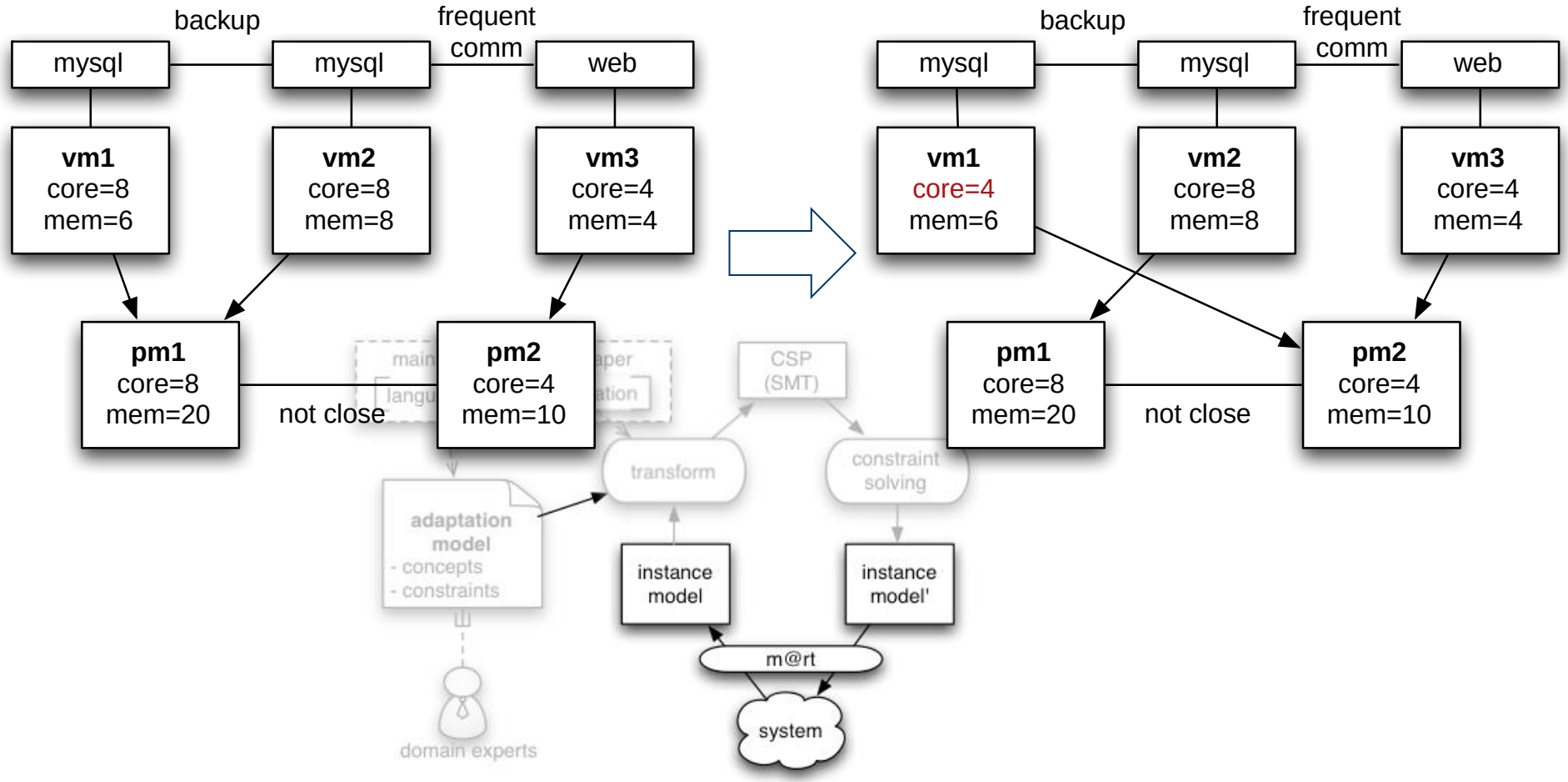
$$\frac{\begin{array}{c}\text{VM} : \{vm_1, vm_2, vm_3\},\ \text{PM} : \{pm_1, pm_2\},\ \text{int},\ \text{boolean} \\ vc : \text{VM} \to \text{int},\ pc : \text{PM} \to \text{int},\ vmem : \text{VM} \to \text{int},\ pmem : \text{PM} \to \text{int} \\ plc : \text{VM} \to \text{PM},\ frqt : \text{VM} \times \text{VM} \to \text{bool},\ near : \text{PM} \times \text{PM} \to \text{bool}\end{array}}{\begin{array}{c}\forall vm, pm.\ (plc(vm) = pm \Rightarrow vc(vm) \leq pc(pm)) \\ \forall pm. \left(\sum_{vm \in VM} \text{ite}(plc(vm) = pm, vmem(vm), 0)) \leq pmem(pm) \right) \\ \forall vm, vm'. (frqt(vm, vm') \Rightarrow near(plc(vm), plc(vm'))), \forall pm. (near(pm, pm))\end{array}}$$

$$vc(vm_1) = 8, vmem(vm_1) = 6, plc(vm_2) = pm_1, frqt(vm_2, vm_3)...$$

- Constraint Solver: Is there an interpretation to each function, to satisfy all the constraints?
  - Yes: Return the interpretation,
  - No: Ignore some "weakest" constraints, and return an interpretation to satisfy all the others – An optimisation problem
- Solver: Z3 by Microsoft research

Song, H., S. Barrett, A. Clarke, and S. Clarke (2013). Self-adaptation with End-User Preferences: Using Run-Time Models and Constraint Solving. In: Model-Driven Engineering Languages and Systems. pp.555–571.

models@runtime



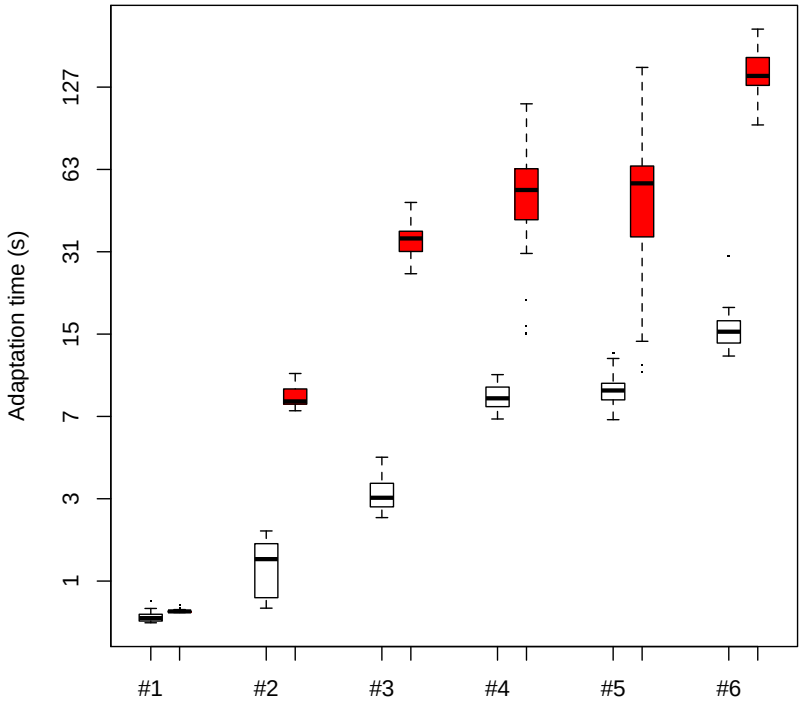
Nicolas Ferry, Hui Song, Alessandro Rossini, Franck Chauvel and Arnor Solberg, CloudMF: Applying MDE to Tame the Complexity of Managing Multi-Cloud Applications, UCC 2014, to appear

A Demo

- Focus on the adaptation effect
- Ignore SMT generation and models@runtime

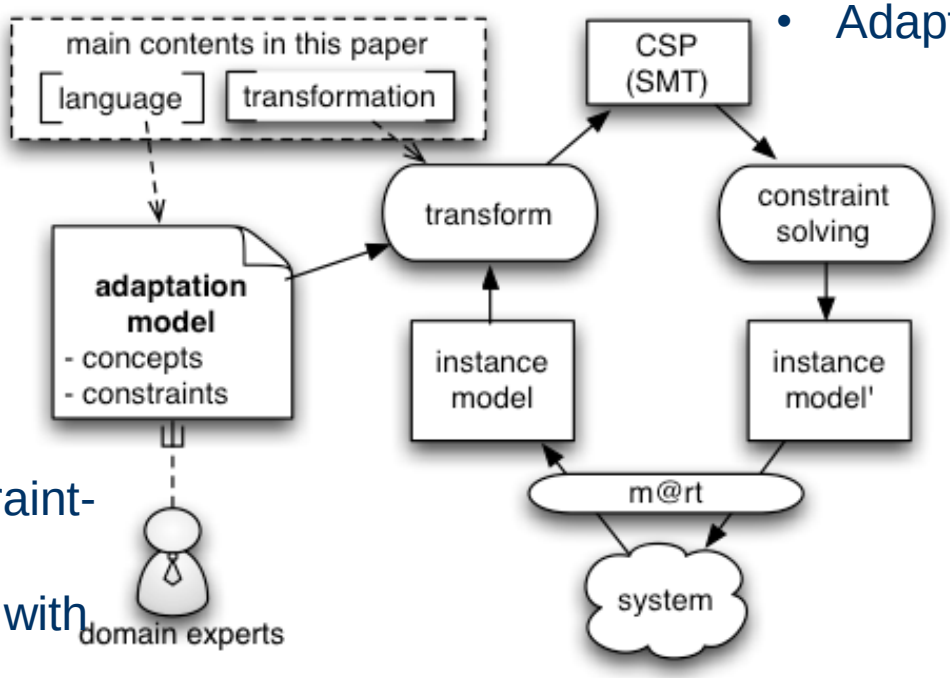
Performance

- Acceptable for medium sized private clouds
  - 60s: 100 vm, 10 pm, 600 properties, 10 changes (as in paper)
  - 60s: 500 vm, 50pm, 3000 properties, 10 changes (now)
- Why
  - Powerful new constraint solver
  - Usually simple constraints
  - A big portion of fixed properties
  - Partial evaluation!



A Short Summary

- SMT represention of architectural models
- OCL to SMT transformation, with partial evaluation



- SMT solving with soft constraints
  - Conflicting constraints
  - Adaptation costs

- Declarative constraint-based modelling
- A text-based DSL with a powerful editor

Application

- Directly used for adaptation
  - Constraints from cloud domain experts
  - Searching for better deployment that fits the context better
- Assessment of adaptation cost
  - *"Is a diverse system easier to be adapted for changing contexts"?*
  - "Dry-run" the solving process on controlled models
  - The total weight of broken constraints is the cost of performing the adaptation

Thank You!

Questions, Comments, Suggestions?

SINTEF ICT

hui.song@sintef.no

https://github.com/songhui/cspadapt