Consolidation dynamique d’applications Web haute disponibilité

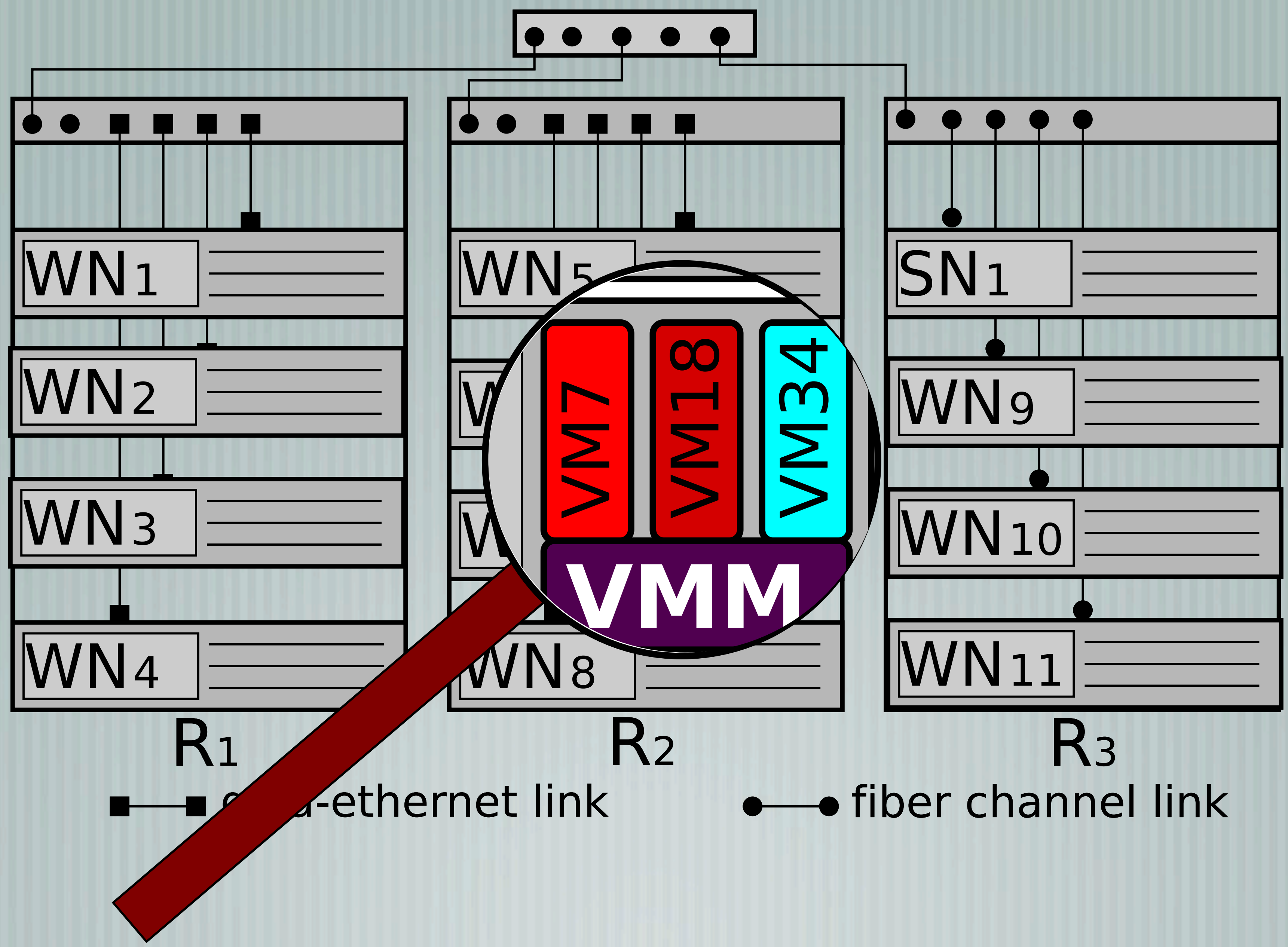
Fabien Hermenier[1], Julia Lawall[2], Jean-Marc Menaud[1], Gilles Muller[3]

1 ASCOLA Mines de Nantes, INRIA, LINA. prenom.nom@mines-nantes.fr
2 DIKU, Université de Copenhague. julia@diku.dk
3 INRIA/LIP6-Régal. gilles.muller@lip6.fr

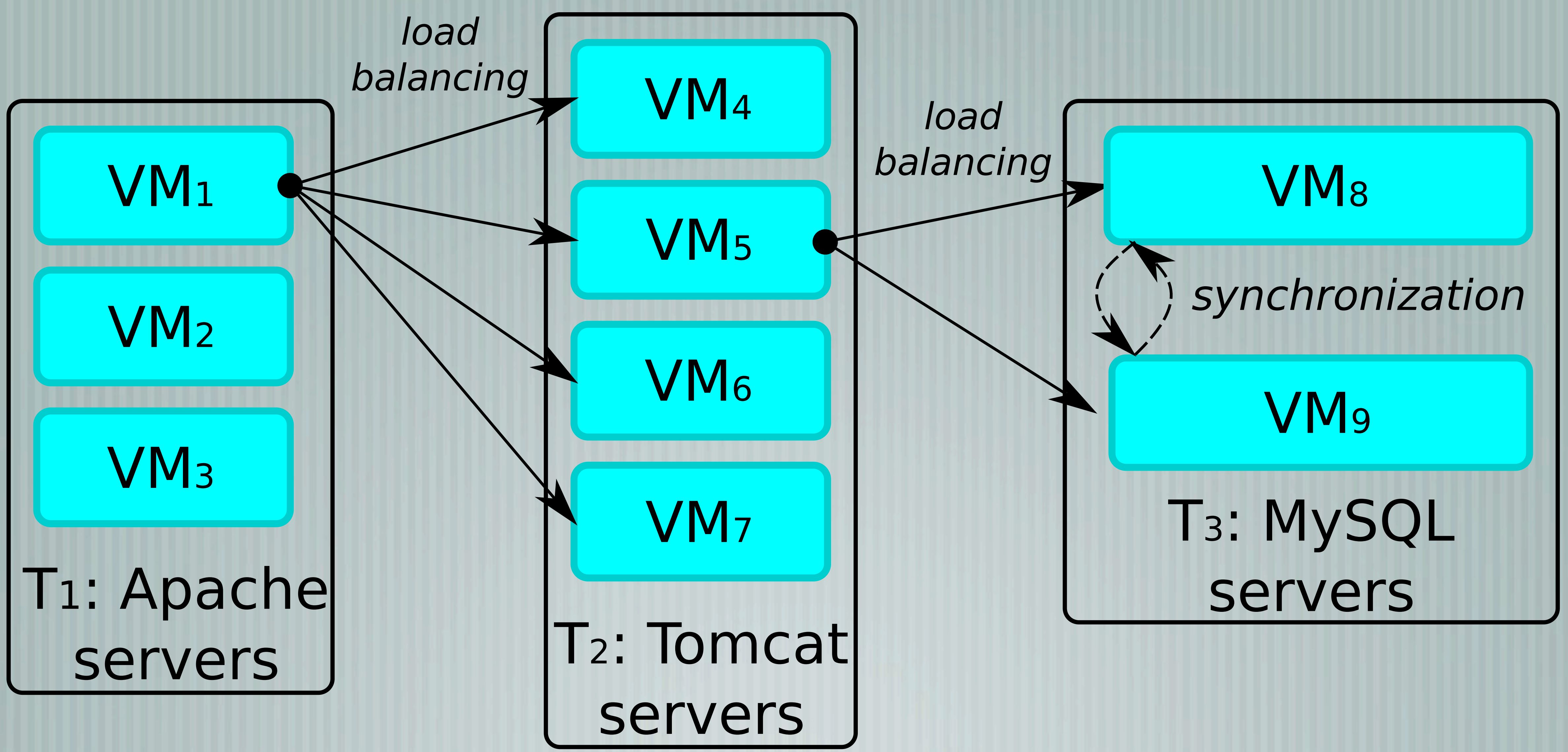
Main

- Application domain : Highly Available Web Applications running on virtualized datacenter
- Goal : Dynamic consolidation to optimize datacenter ressource usage
- Contribution : Plasma, a dynamic consolidation manager, that can be configured to take into account resource and placement constraints for HA application

Virtualized datacenter



Virtualized highly-available Web application



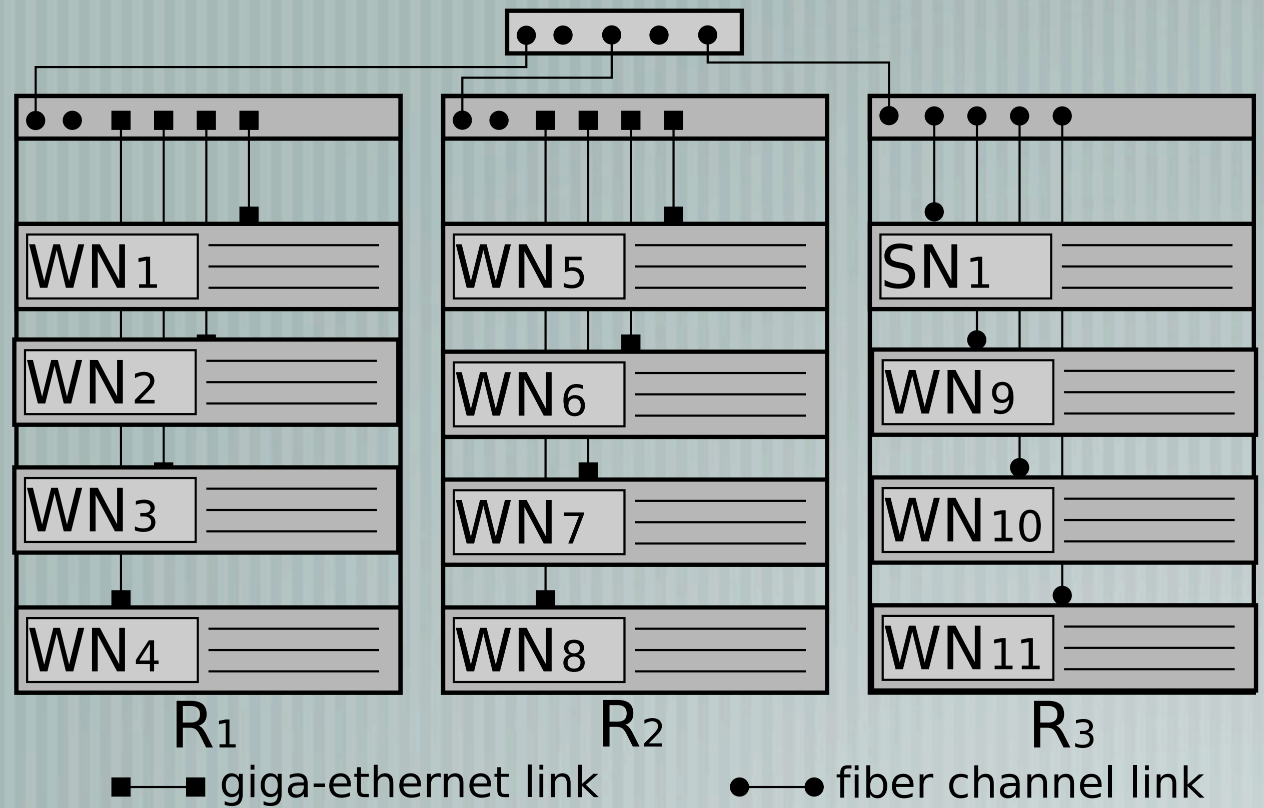
Dynamic consolidation meets high-availability

- System administrator wants
  - to stack VMs on nodes to improve resource usage
  - an autonomous management of the VMs
- Application administrator wants its VMs placed wrt. :
  - their resource requirements
  - fault tolerance to hardware failure for replicated services
  - a network latency compatible with the synchronization protocol

The challenge

- Some problems
  - multiple specific placement constraints
  - concurrent/overlapping constraints
  - constraints expressed by non-expert users
- One proposition
  - easy specification of placement constraints with declarative scripts
  - extensible autonomous VM manager, specialized by the constraints

Datacenter description

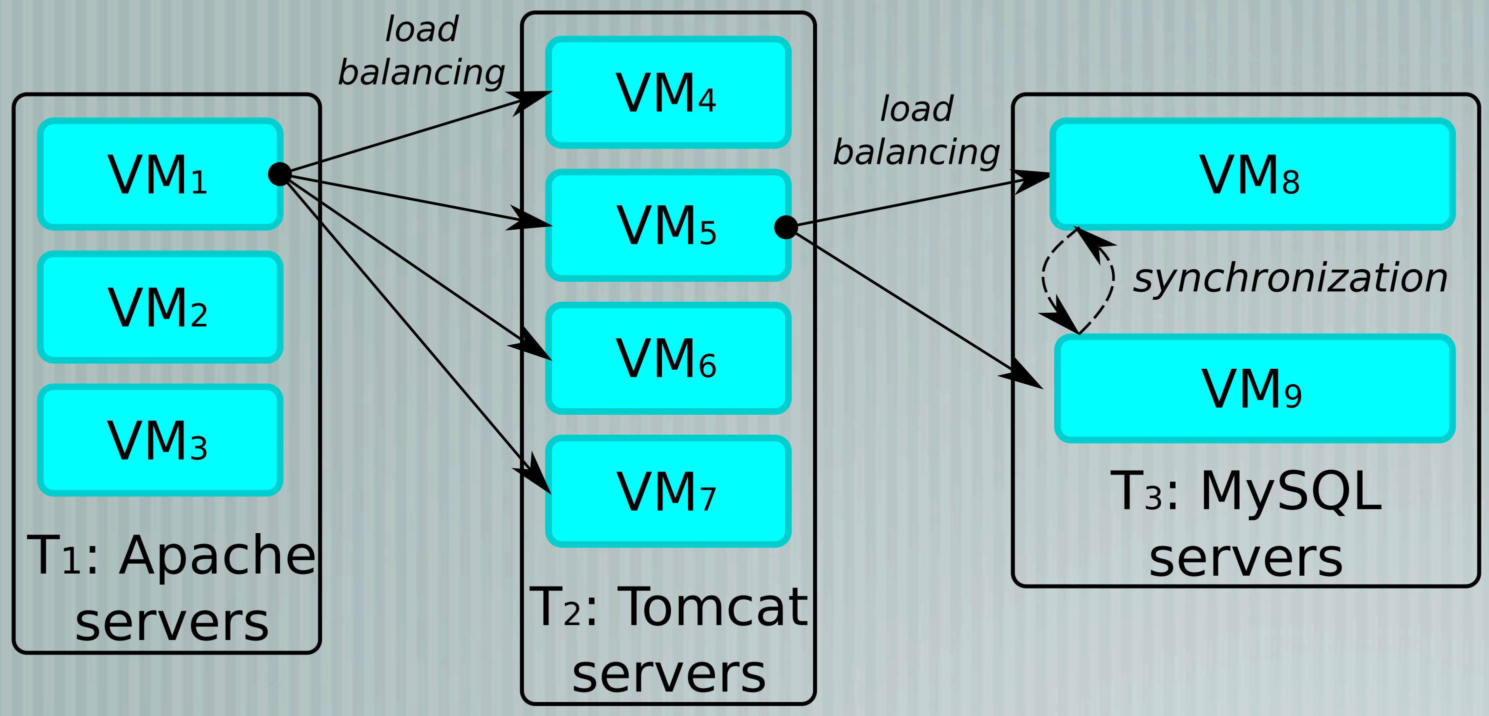



```
// Infrastructure
$R1 = {WN1 , WN2 , WN3 , WN4 };
$R2 = WN [5..8];
$R3 = WN [9..11] + {SN1 };

// Classes of latency
$small = {$R3 };
$medium = $R [1..3];

// Constraints
ban ( $ALL_VMS , {SN1 });
ban ( $ALL_VMS , {WN5 });
fence ($A1 , $R2 + $R3 );
```

Application description



// The 3 tiers
$T1 = {VM1 , VM2 , VM3 };
$T2 = VM [4..7];
$T3 = VM [8..9];

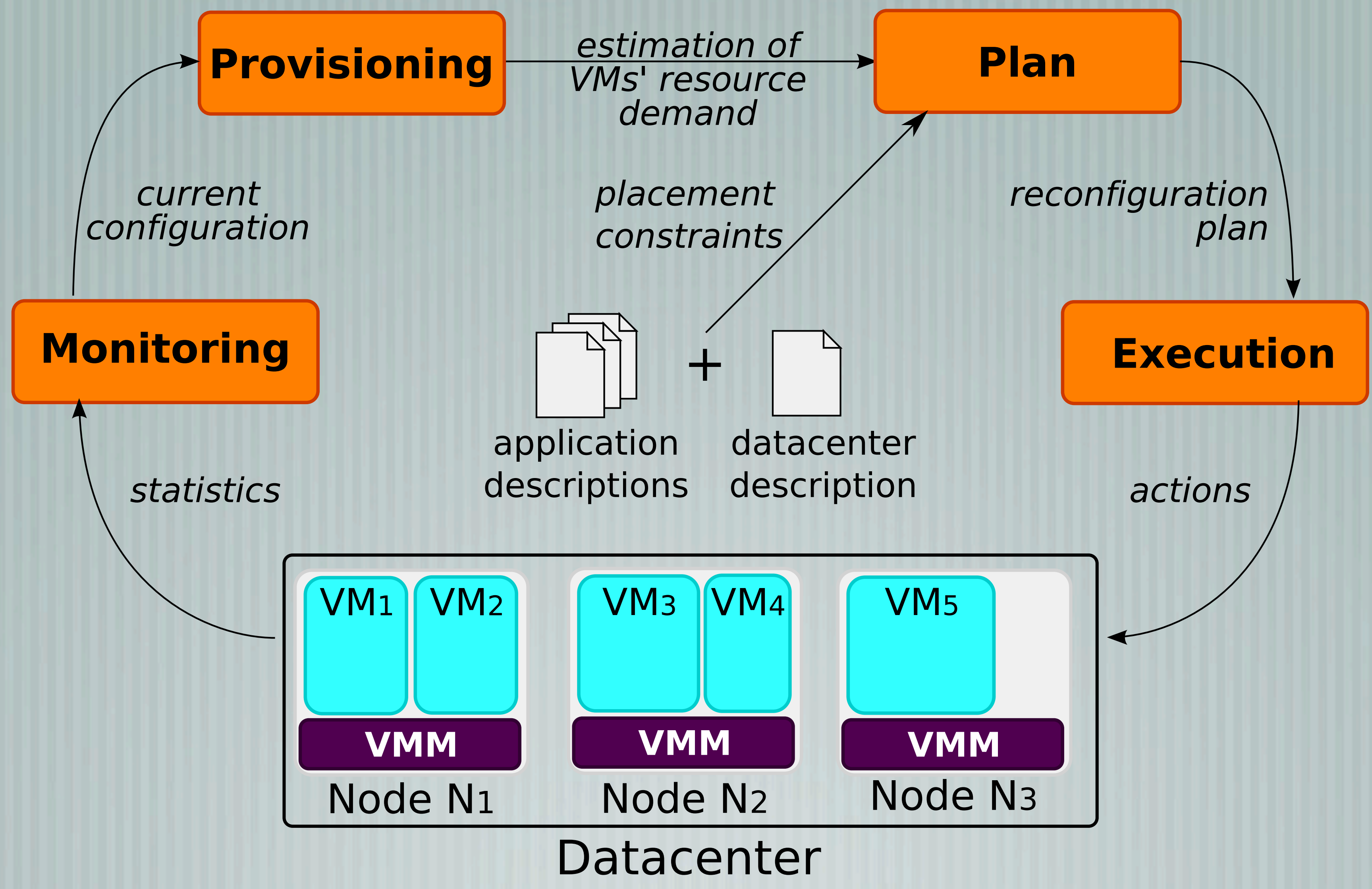
// Fault tolerance to hw. failures
spread($T1);
spread($T2);
spread($T3);

// Efficient synchronization
latency ($T3 , $medium);

Constraints

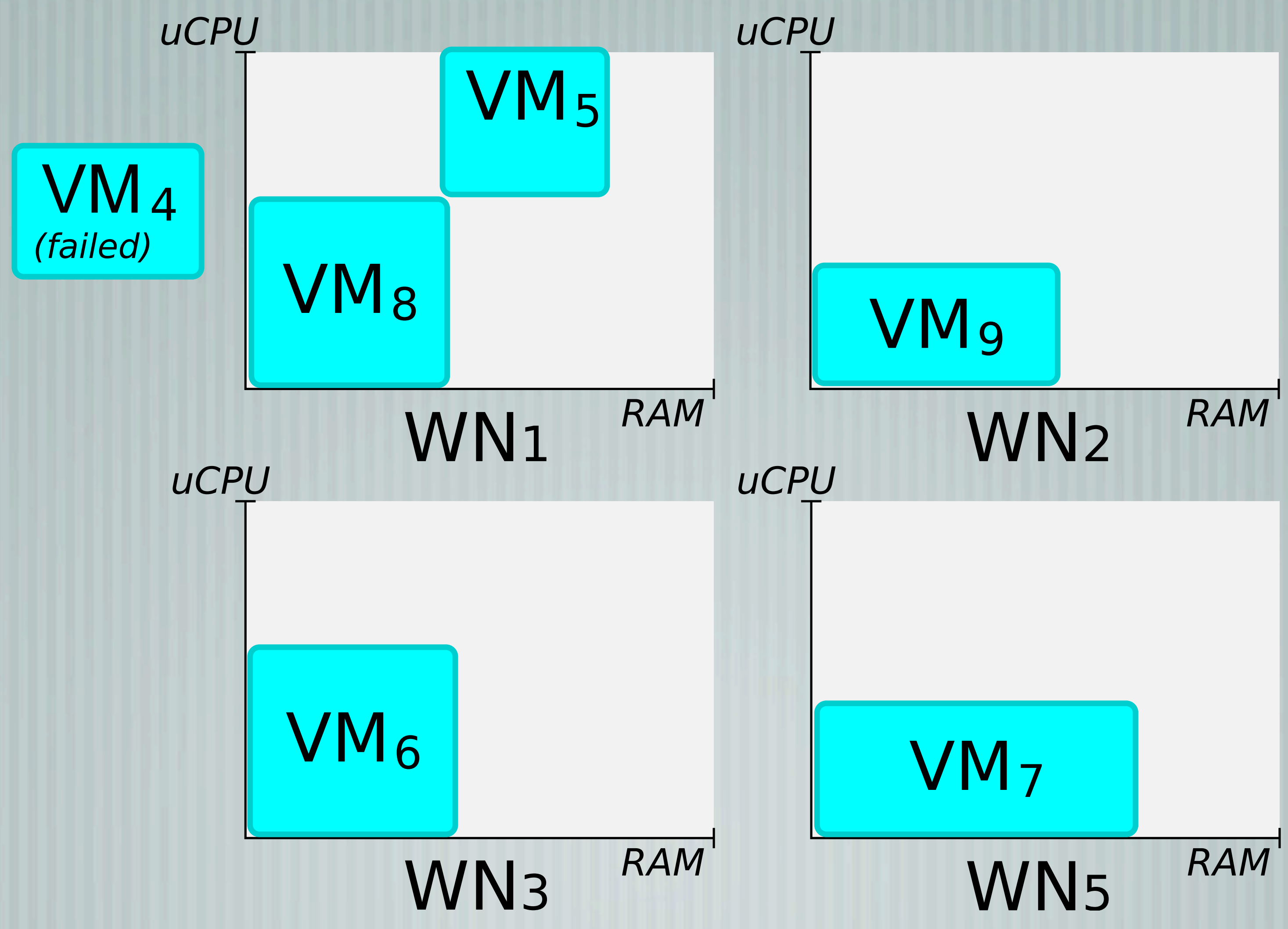
- **ban**({VM1, VM2}, {N1, N2})
  - prevents a set of VMs from being hosted on a given set of nodes
- **fence**({VM1, VM2}, {N1, N2})
  - forces a set of VMs to be hosted on a set of nodes
- **spread**({VM1, VM2})
  - ensures that the specified VMs are never hosted on the same node at the same time
- **latency**({VM1, VM2}, {{N1, N2}, {N3, N4}})
  - forces a set of VMs to be hosted on a single group of nodes.

Control loop of Plasma



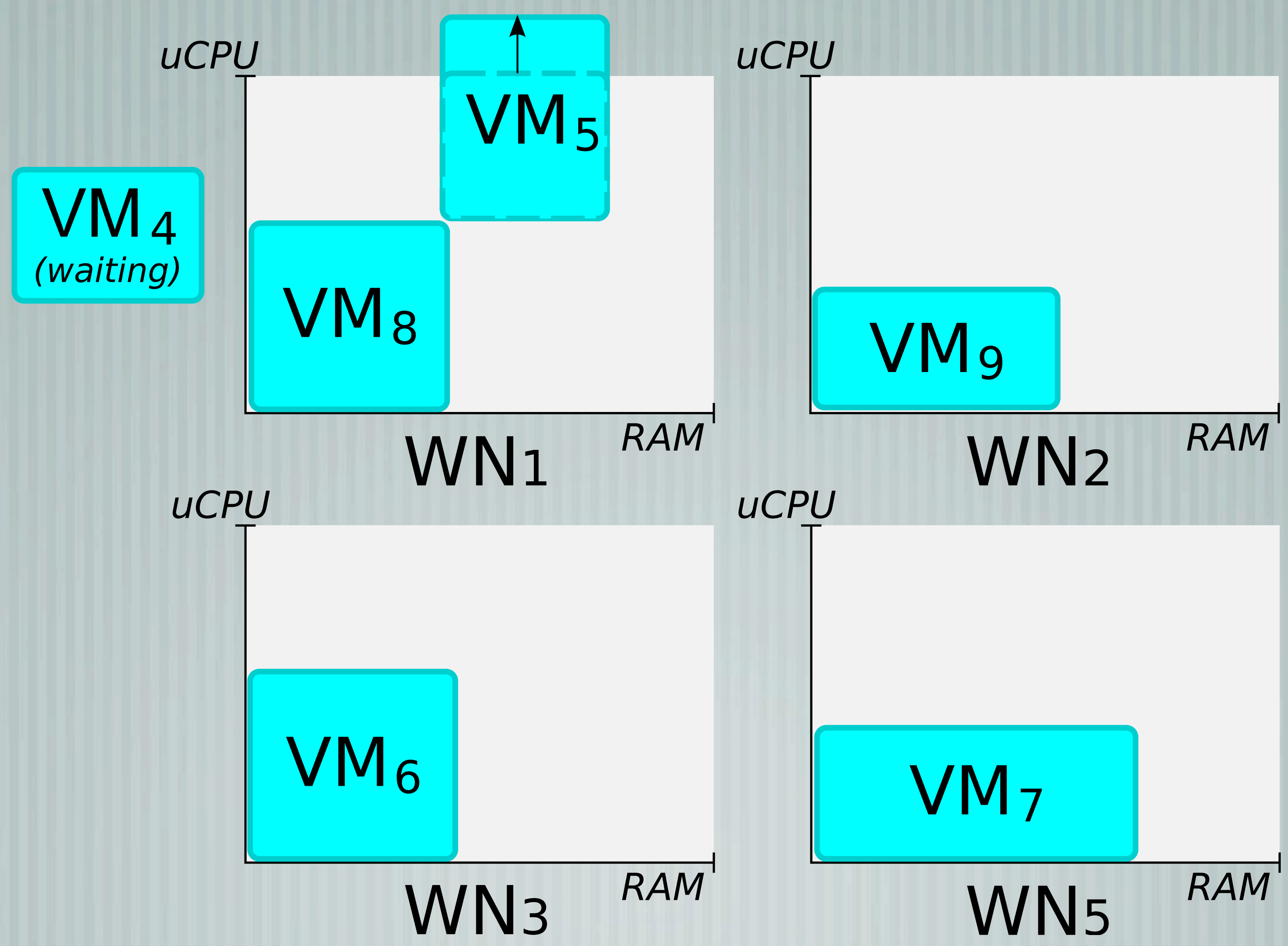
Sample loop iteration - Monitor

- Retrieves the current state of the datacenter



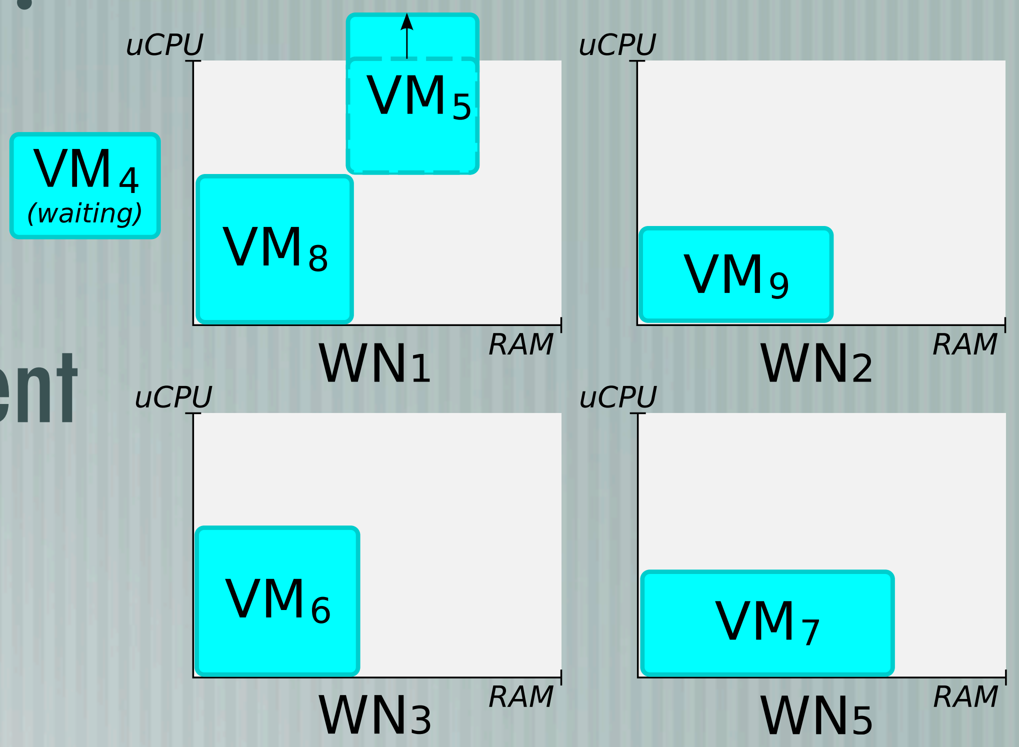
Sample loop iteration - Provisioning

- Estimates the needs of the applications



Sample loop iteration - Plan

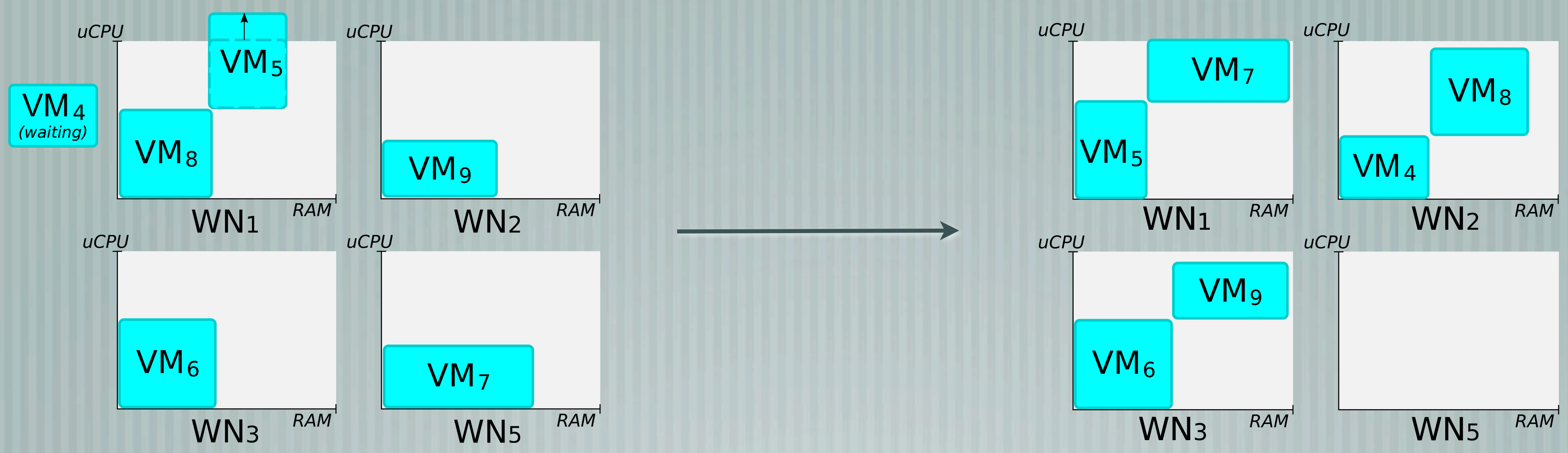
- Current configuration is not viable :
  - VM4 must be running
  - VM5 does not have access to sufficient uCPU resource
  - WN5 should not host any VMs



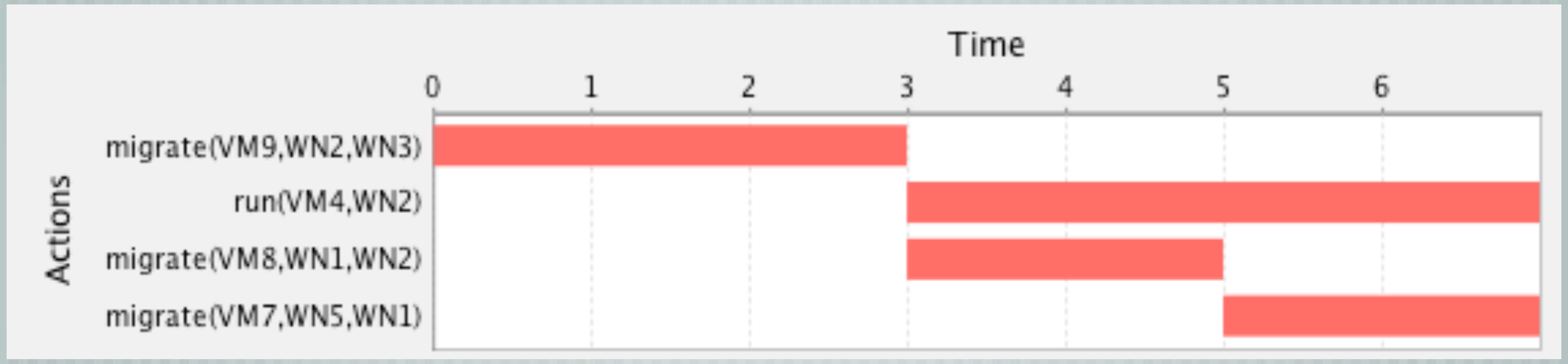
- Reconfiguration: actions on VMs and nodes to reach a viable configuration
  - migration, suspend, resume, shutdown, startup, . . .

Sample loop iteration - Plan

Compute a viable placement for the VMs



Schedule the actions



Compute a viable placement

- The approach : constraint programming
  - generation of a core model
  - placement constraints are translated into "CP constraints"

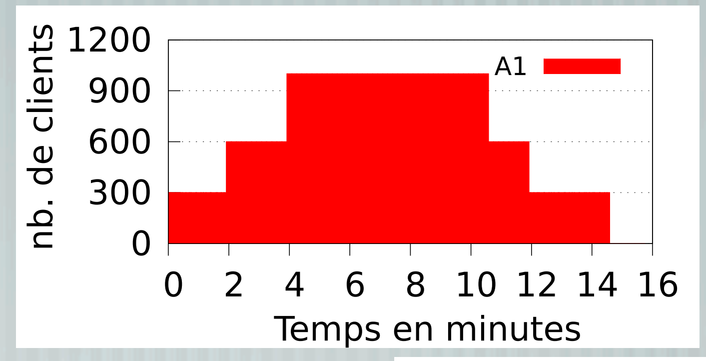
$$
\begin{aligned}
\mathcal{X} &= \{x_1, x_2, x_3\} \\
\mathcal{D}(x_i) &= [0, 4]\ , \forall x_i \in \mathcal{X} \\
\mathcal{C} &= \begin{cases} c_1: & x_1 < x_2 \\ c_2: & x_1 + x_2 + x_3 = 4 \\ c_3: & \mathit{allDifferent}(x_1, x_2, x_3) \end{cases}
\end{aligned}
$$

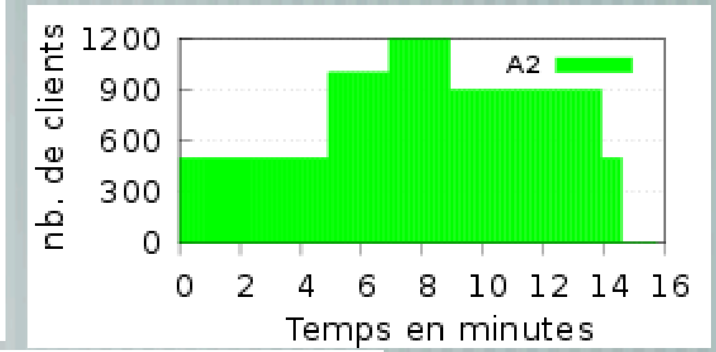
Constraint Programming

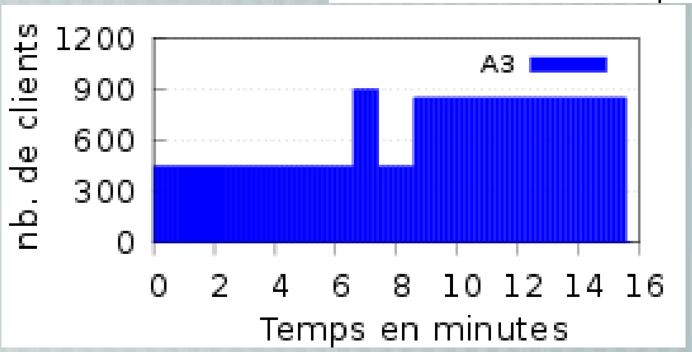
- Pro
  - high-level standardized constraints, portability of a model
  - good expressivity
  - deterministic composition
  - deterministic solving process
- Cons
  - hard to develop efficient custom constraints
  - exact solving duration
  - bad model leads to bad performance

Evaluation

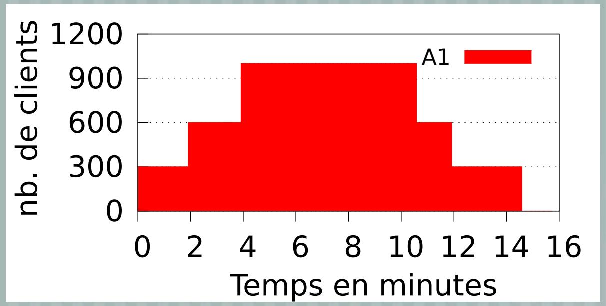
- RUBiS : The three tiers of each instance of RUBiS are deployed as 7 VMs (2,3,2)
- 3 applications
- 21 nodes

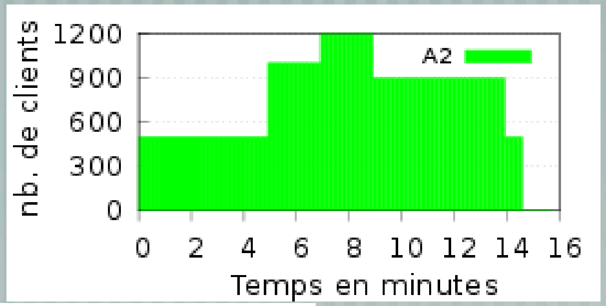


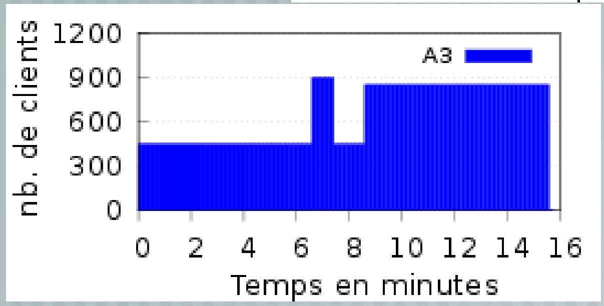


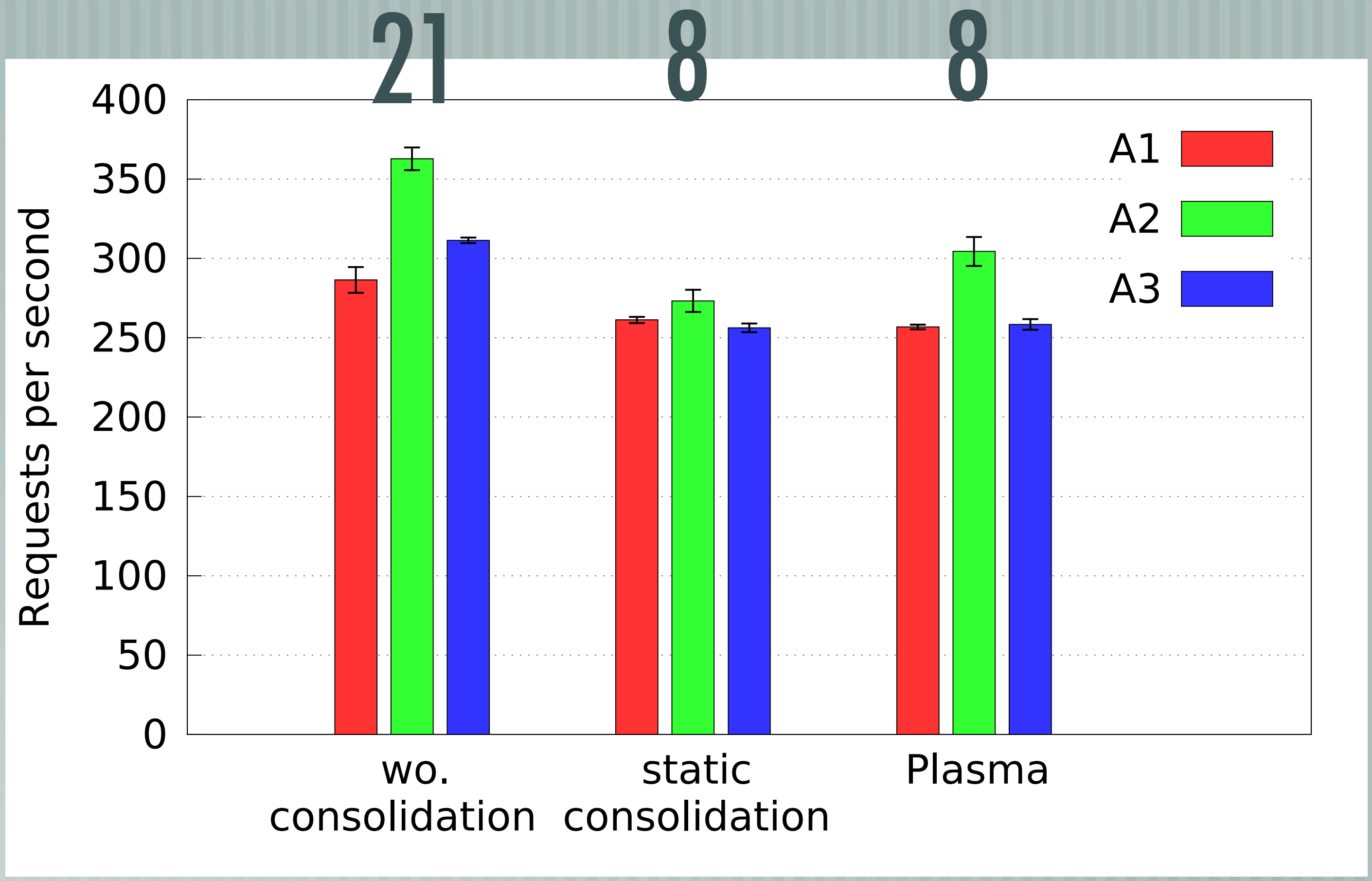


RUBiS Benchmark : Load spikes









- Improvement wrt. static consolidation (14.7% vs. 17.7%)
- About 12 reconfigurations (29 secs) per execution
- Longest reconfiguration: 10 migrations in 89 seconds

RUBiS Benchmark : external events

Time	Event	Reconfiguration Plan Actions	Reconfiguration Plan Duration
2'10	+ ban({WN8})	3 + **3 migrations**	0'42
4'30	+ ban({WN4})	2 + **7 migrations**	1'02
7'05	- ban({WN4})	no reconfiguration	
11'23	+ ban({WN4})	**no solution**	
11'43	- ban({WN8}) + ban({WN4})	**2 migrations**	0'28

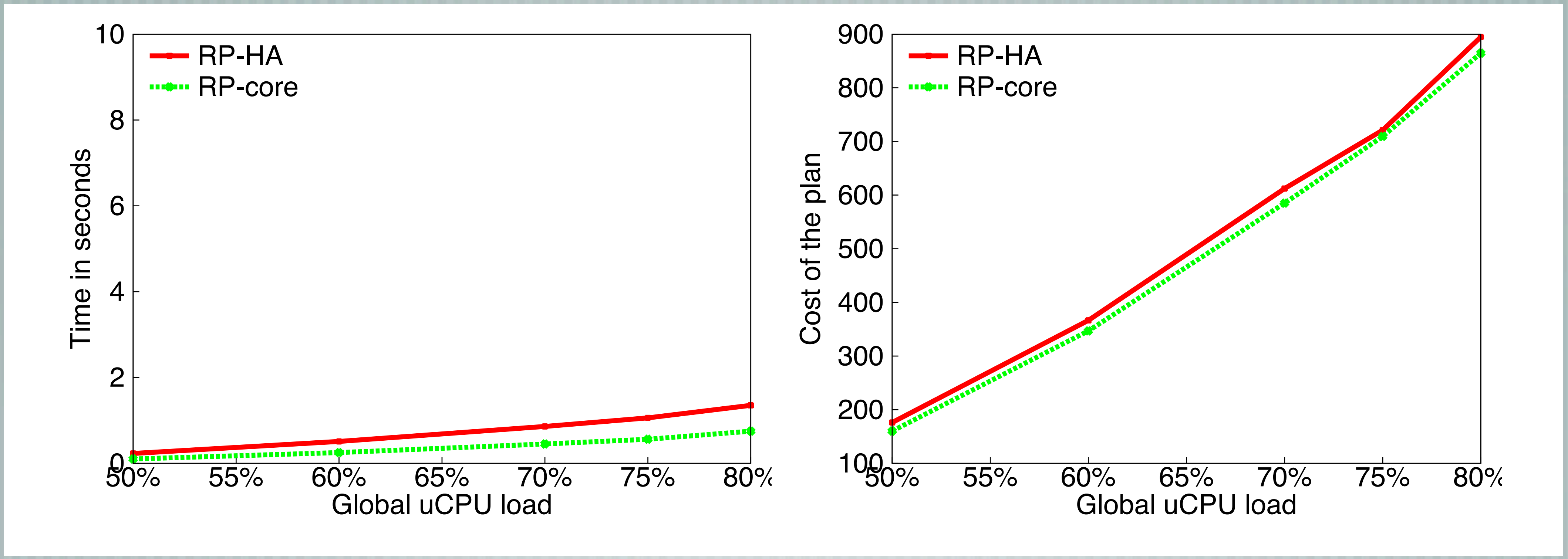
Hidden side effects on Entropy, not the sys-admin

Scalability evaluation

- Simulated instances
  - 200 nodes in 4 racks and 2 partitions
  - 400 VMs: 20 HA Web application (20 VMs each)
  - initial placement and uCPU usage computed pseudo-randomly
  - 1% of failed nodes
- Consolidation scenario
  - RP-Core without the application placement constraints,
  - RP-HA with the application constraints

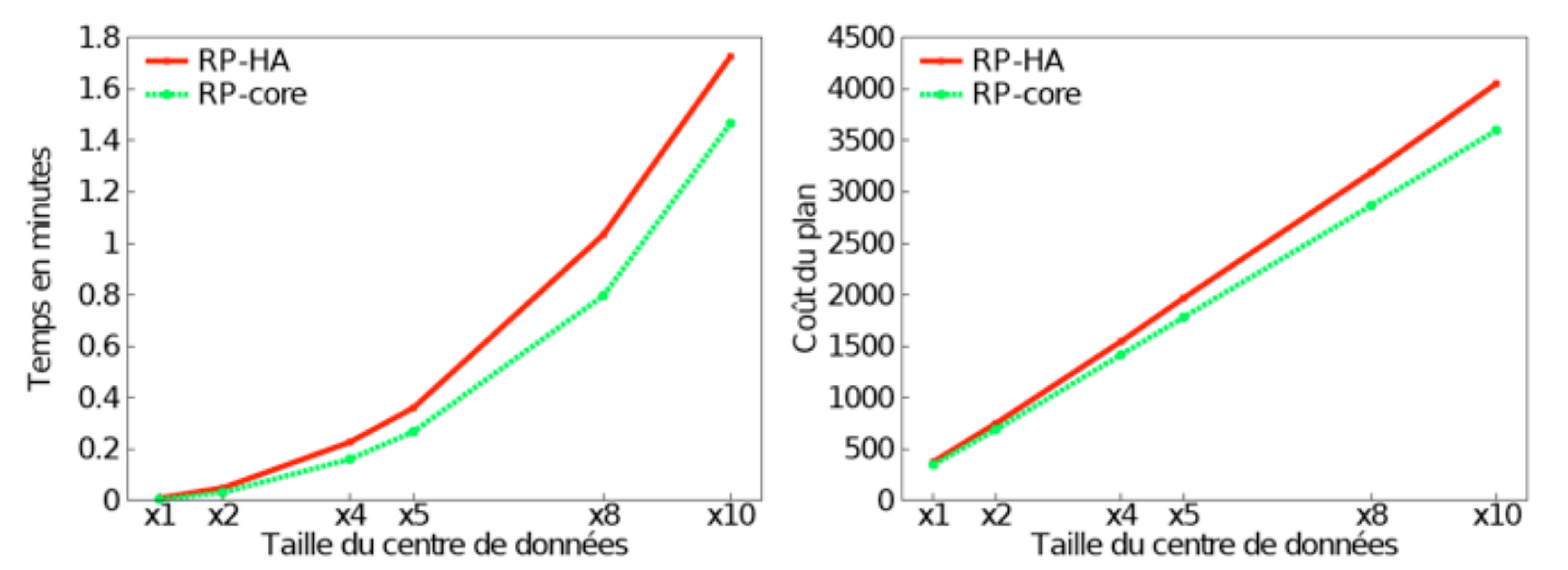
Impact of the global uCPU demand

- Impact of placement constraints is not significant



Impact of the problem size

- In practice
  - place 1117 candidates VMs on 1980 nodes with 600 spread + 200 latency
  - schedule 475 actions



(a) Temps de résolution

(b) Coût des solutions

Conclusion

- Plasma
  - configurable consolidation manager through scripts
  - scalable to datacenter with up to 2000 nodes/ 4000 VMs
  - placement constraints do not impact the solving process
- Futures works
  - new placement constraints for new concerns *(currently 10 constraints)*
  - improvment of the scalability using partitioning *(done)*
  - soft placement constraints with penalty cost.

Entropy

- an open-source project: http://entropy.gforge.inria.fr
- some publications: a Phd. thesis, VEE'09, CPAIOR'10, VTDC'10, XHPC'06, JFPC'10, CFSE[6,7,8]
- some industry partners: DGFiP, Orange Labs, Bull, etc.
- strong partnership with the Constraint team
- 2 awards