flexible autonomous datacenter resource management

Sophie Demassey
CMA Centre de Mathématiques Appliquées
MINES ParisTech

Fabien Hermenier
Université Nice Sophia Antipolis
i3S

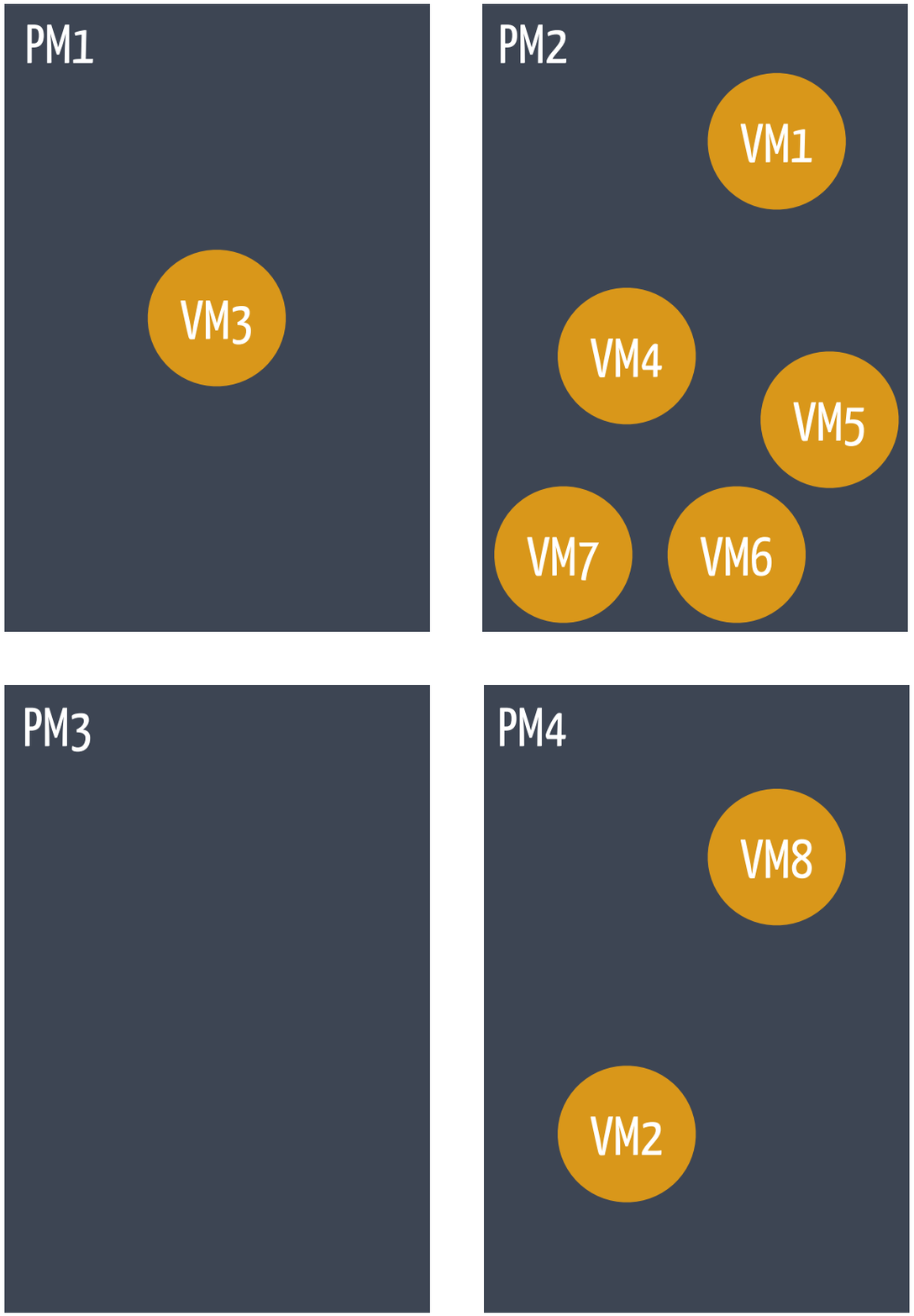
optimality

approximation

speed

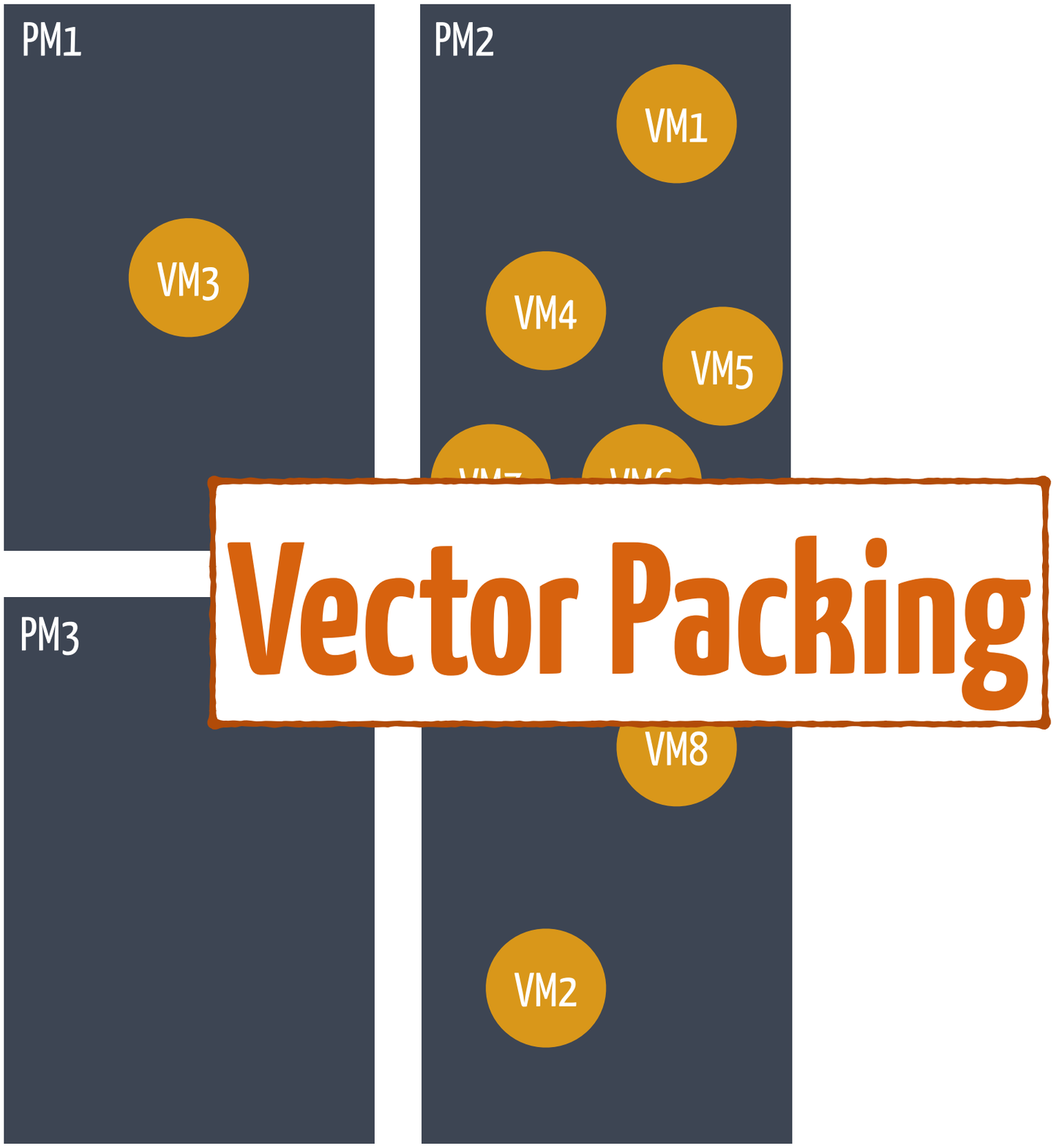
flexibility

datacenter resource management



Physical Machines (PM)

Virtual Machines (VM)

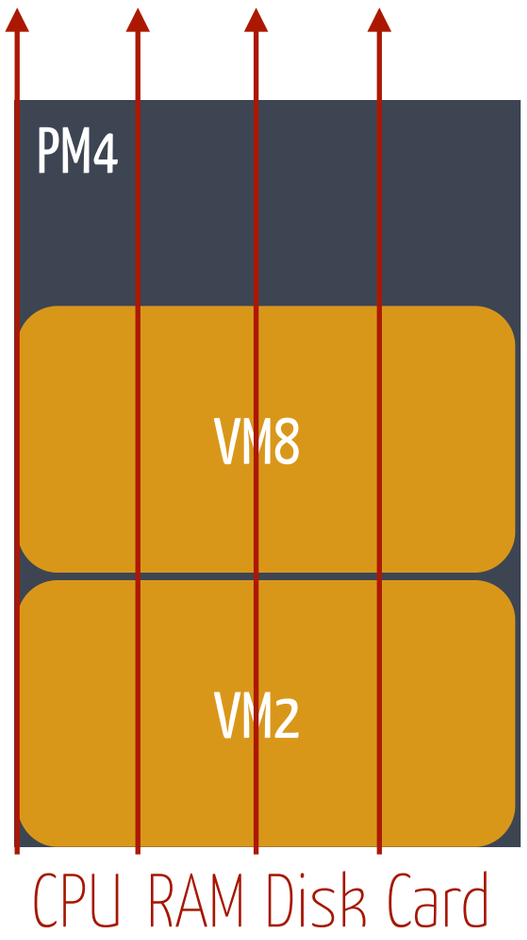


Vector Packing

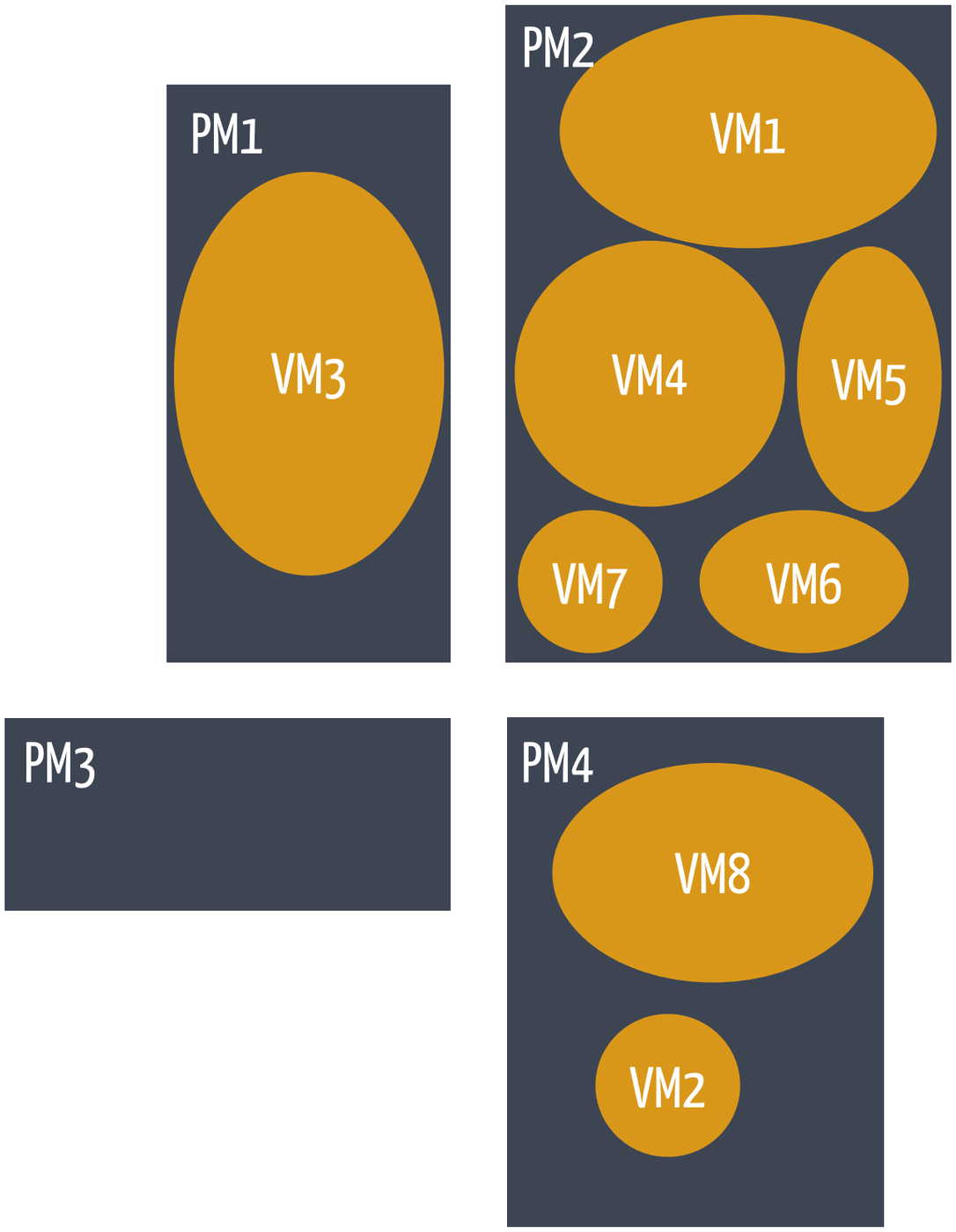
Physical Machines (PM)

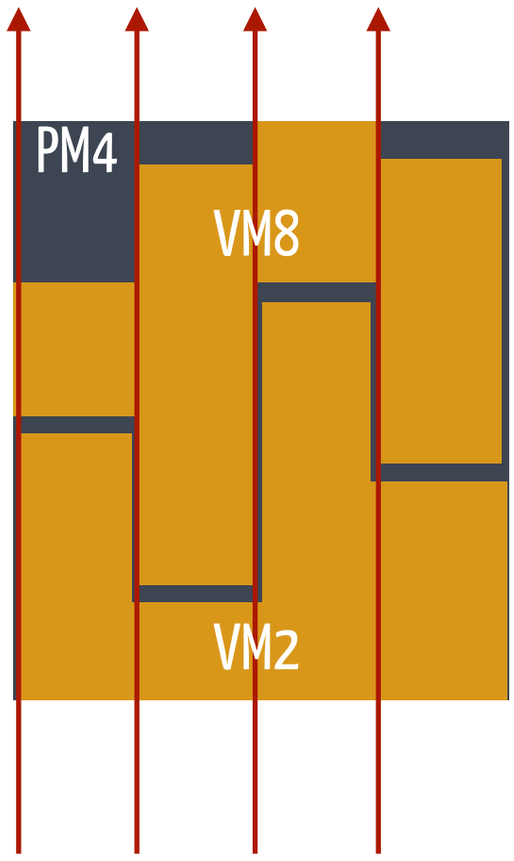
Virtual Machines (VM)

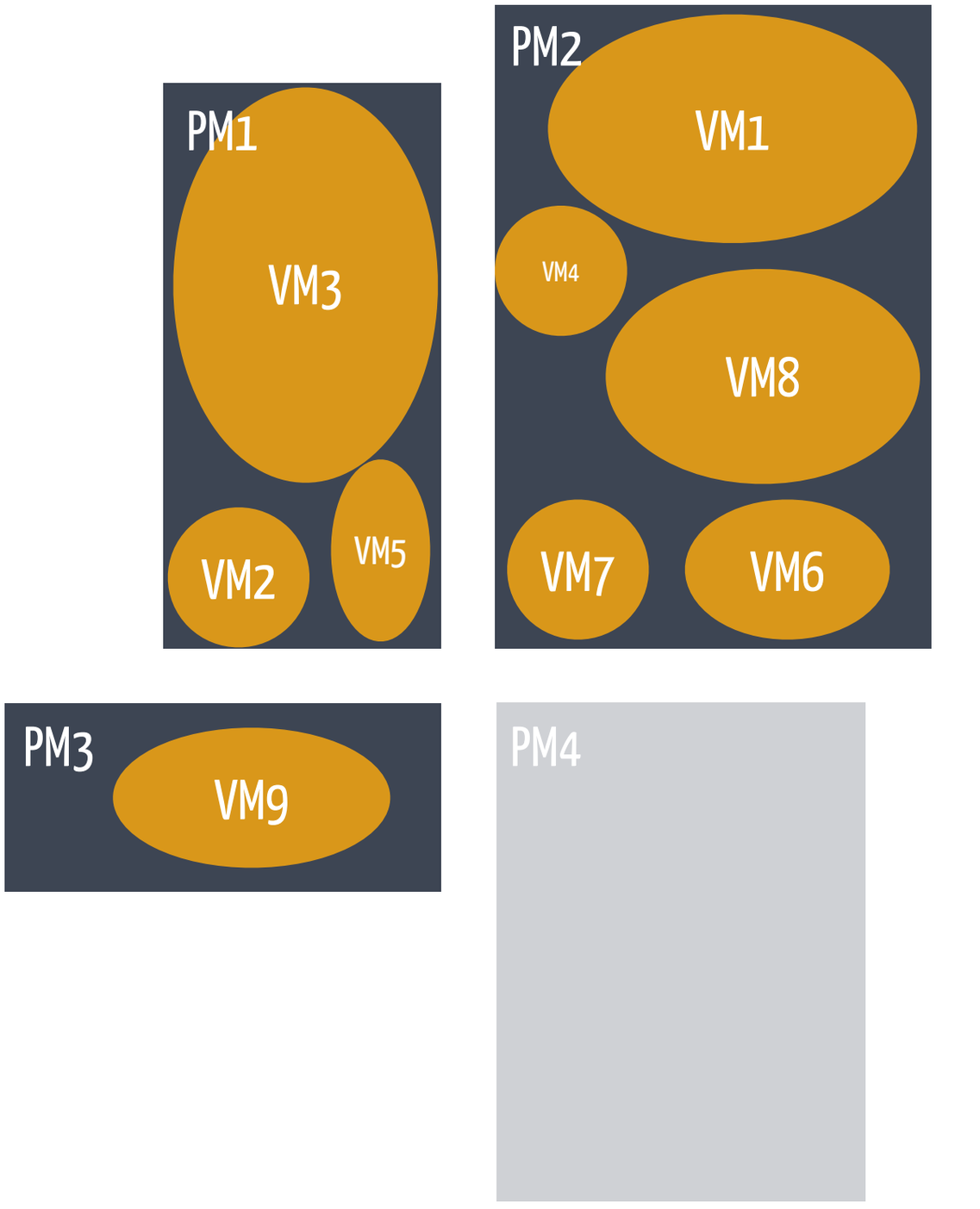
Resources



1 Heterogeneous







submissions
load change
failures

Dynamic?

launch
resize
stop
migrate off or live

reconfiguration actions

PM1

VM5

VM2

VM3

t

migrate

resize

relaunch

PM2

VM5

VM8

launch

PM3

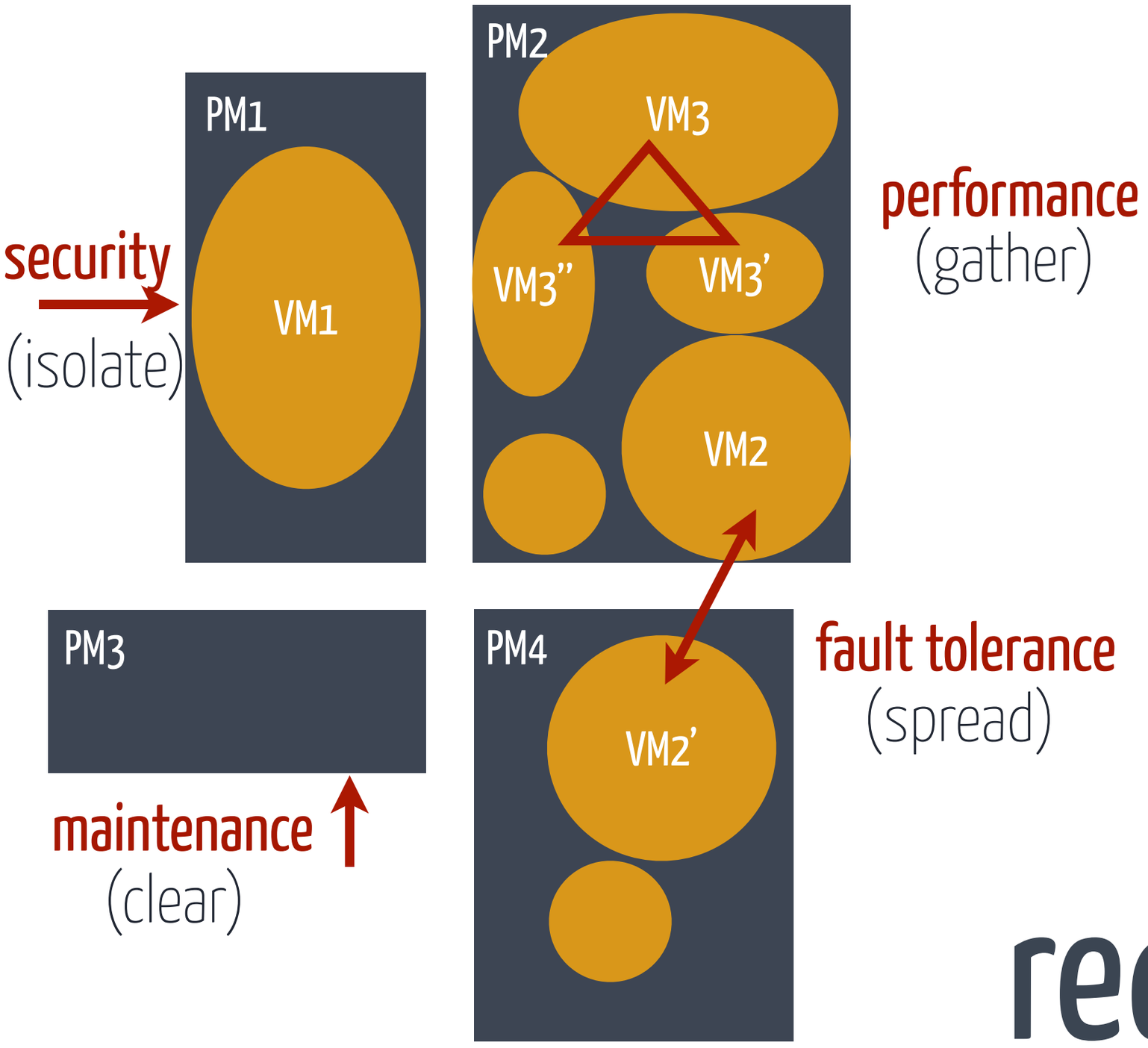
PM4

VM2

VM8

stop

Producer/Consumer Scheduling



3 User requirements

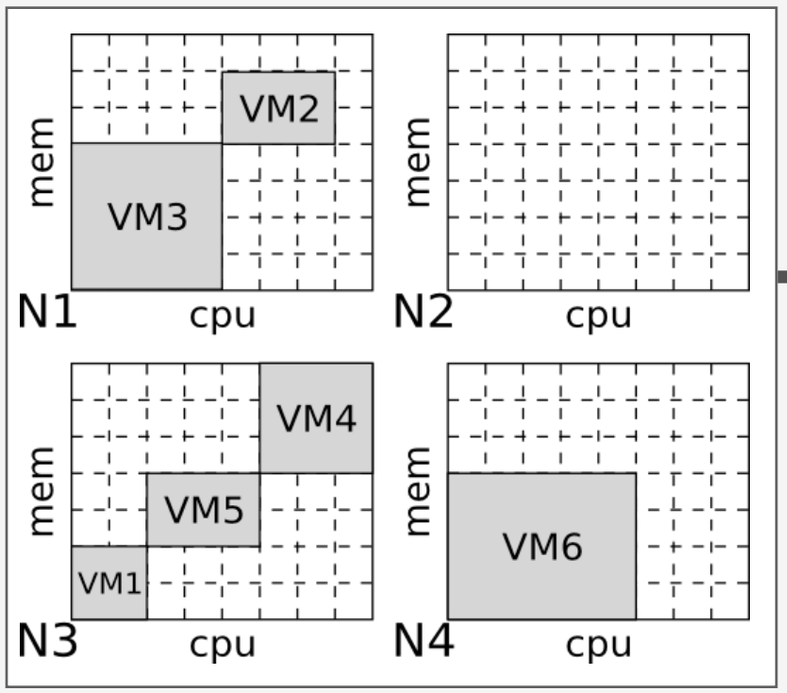
(heterogeneous & dynamic)

flexible

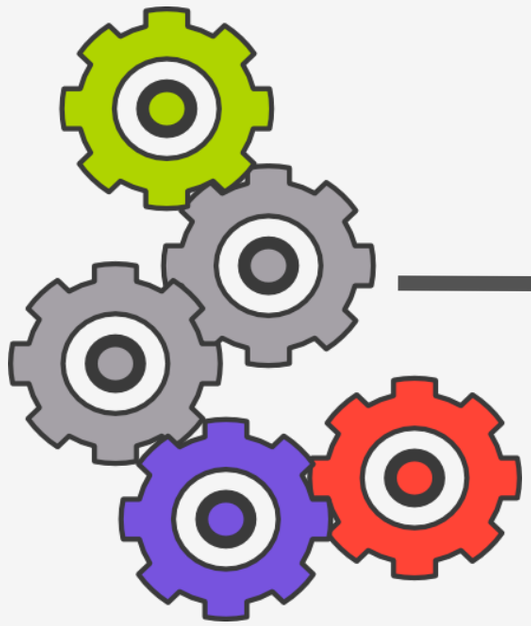
extensible (offline)

configurable (online)

An Open-Source flexible virtual machine scheduler



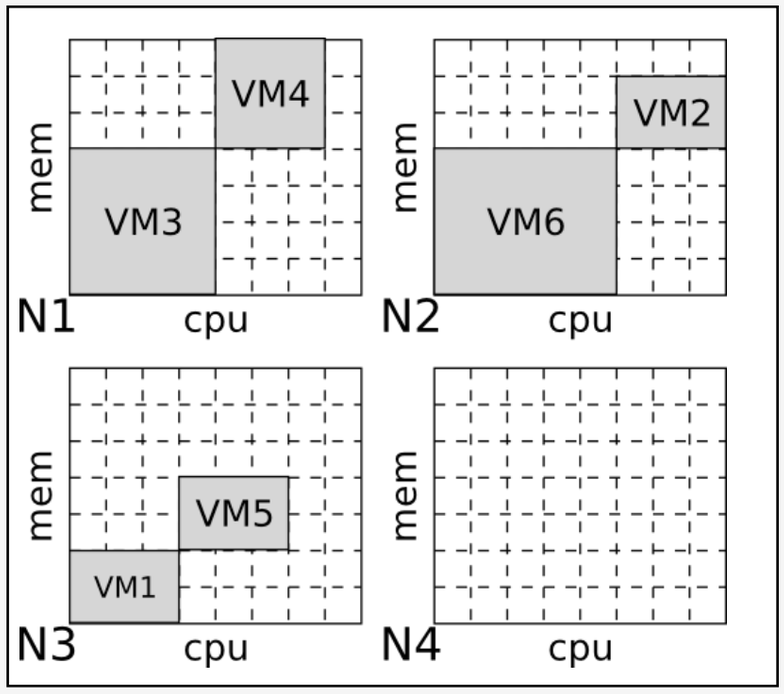

```
spread(VM[2..3]);
preserve({VM1},'ucpu', 3);
offline(@N4);
```



BtrPlace

The reconfiguration plan

```
0'00 to 0'02: relocate(VM2,N2)
0'00 to 0'04: relocate(VM6,N2)
0'02 to 0'05: relocate(VM4,N1)
0'04 to 0'08: shutdown(N4)
0'05 to 0'06: allocate(VM1,'cpu',3)
```

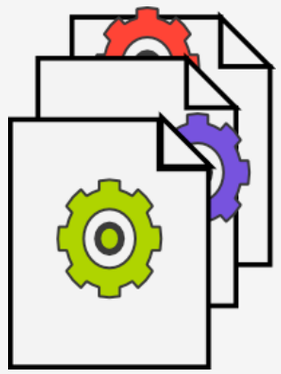


BtrPlace proposal

core reconfiguration algorithm

+

users scripts



=

specialized reconfiguration algorithm

routines library

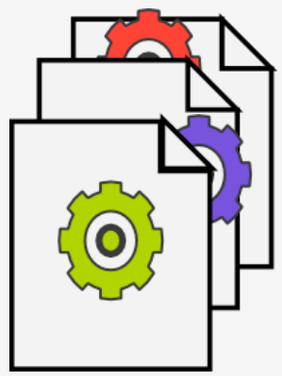
flexibility through constraint programming

packing + scheduling

core model

+

global constraints



Variables related to VM Management	
c^{host}	Current host of the VM (constant)
c^{mem}, c^{cpu}	Current amount of memory and uCPU resources allocated to the VM (constant)
c^{ed}	Time the VM may leave its current host
d^{host}	Next host of the VM
d^{mem}, d^{cpu}	Next amount of memory and uCPU resources to allocate to the VM
d^{st}	Time the VM arrives on its next host
Variables related to server management	
n^q	Next state of the server

`spread({VM1,VM2}):`

$$allDifferent(d_1^{host}, d_2^{host}) \wedge$$
$$d_1^{host} = c_2^{host} \rightarrow d_1^{st} \geq c_2^{ed} \wedge$$
$$d_2^{host} = c_1^{host} \rightarrow d_2^{st} \geq c_1^{ed}$$

constraint library

ban	unary
fence	unary
root	unary
quarantine	unary
spread	`alldifferent` (+ precedences)
lonely	`disjoint`
capacity	`gcc`
among	`element`
gather	`allequals`
mostly spread	`nvalue`
split	`disjointmultiple`
split among	`element + alldifferent`
max online	`weightedsum (cumulative)`

scalability through

light filtering packing in O(PMs)

local search fixing non-violations

partition split in subproblems

truncated B&B first solution

EVALUATION

PROTOCOL

5,000 PMs

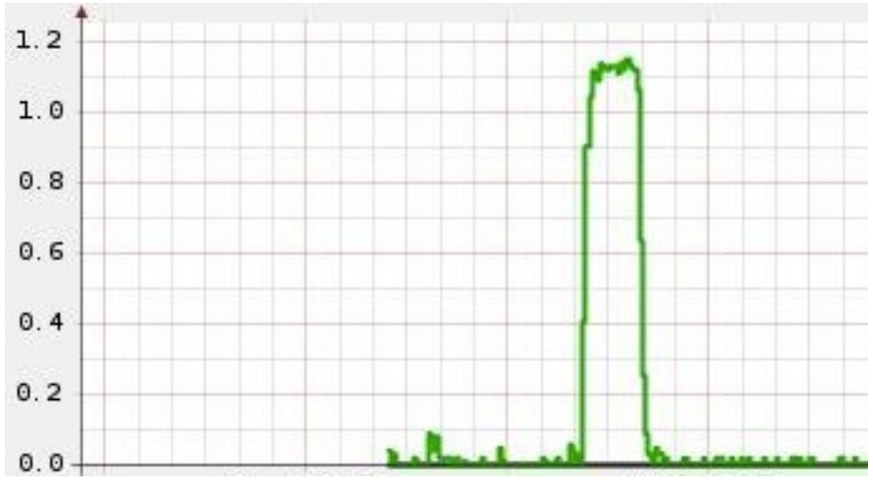
15,000 - 30,000 VMs

extra-large/high-memory EC2 instances

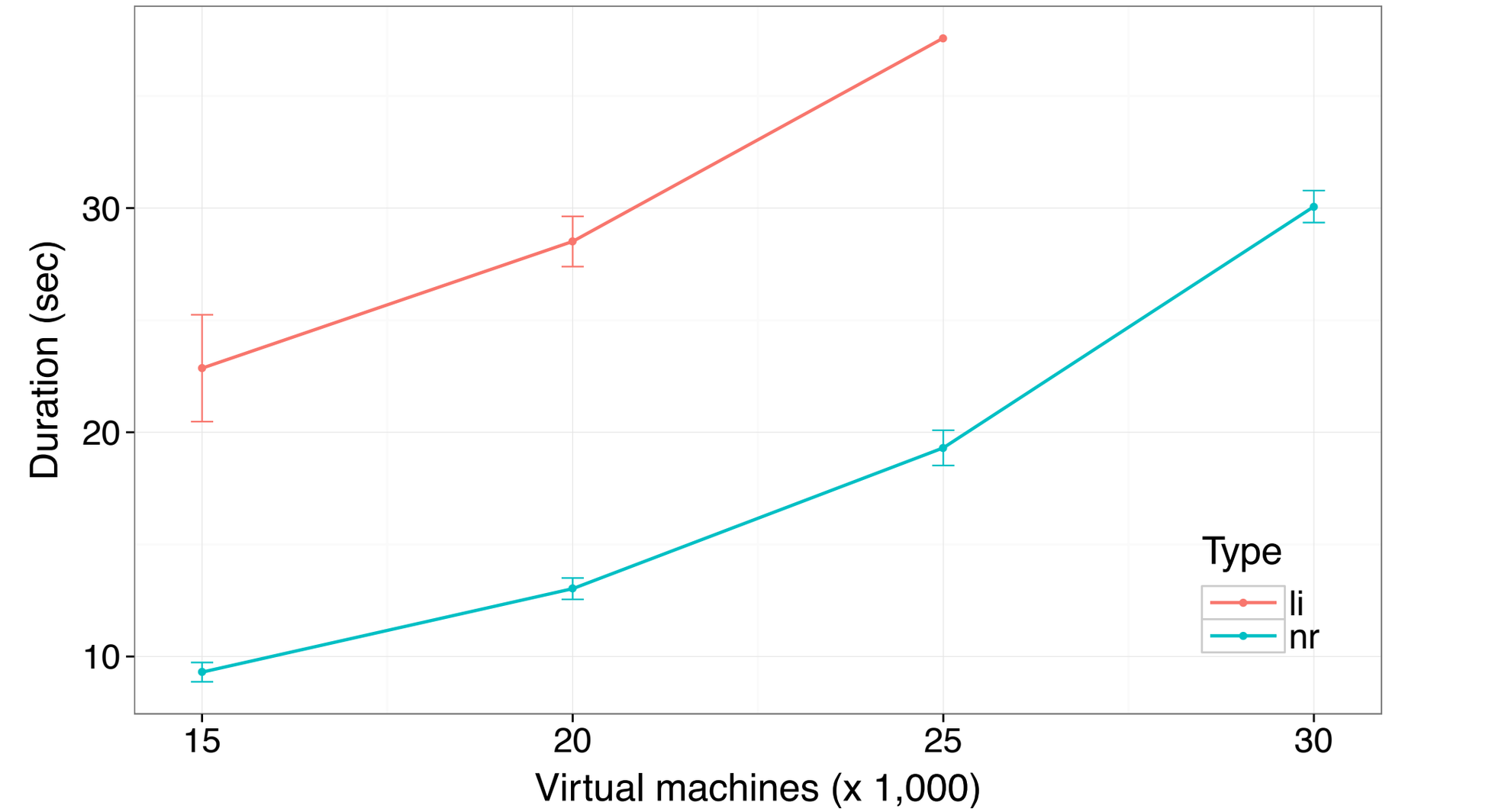
3-tiers HA web applications with replica

LI: 10% VM grow 30% uCPU

NR: 5% PM halt



solution time / **load**

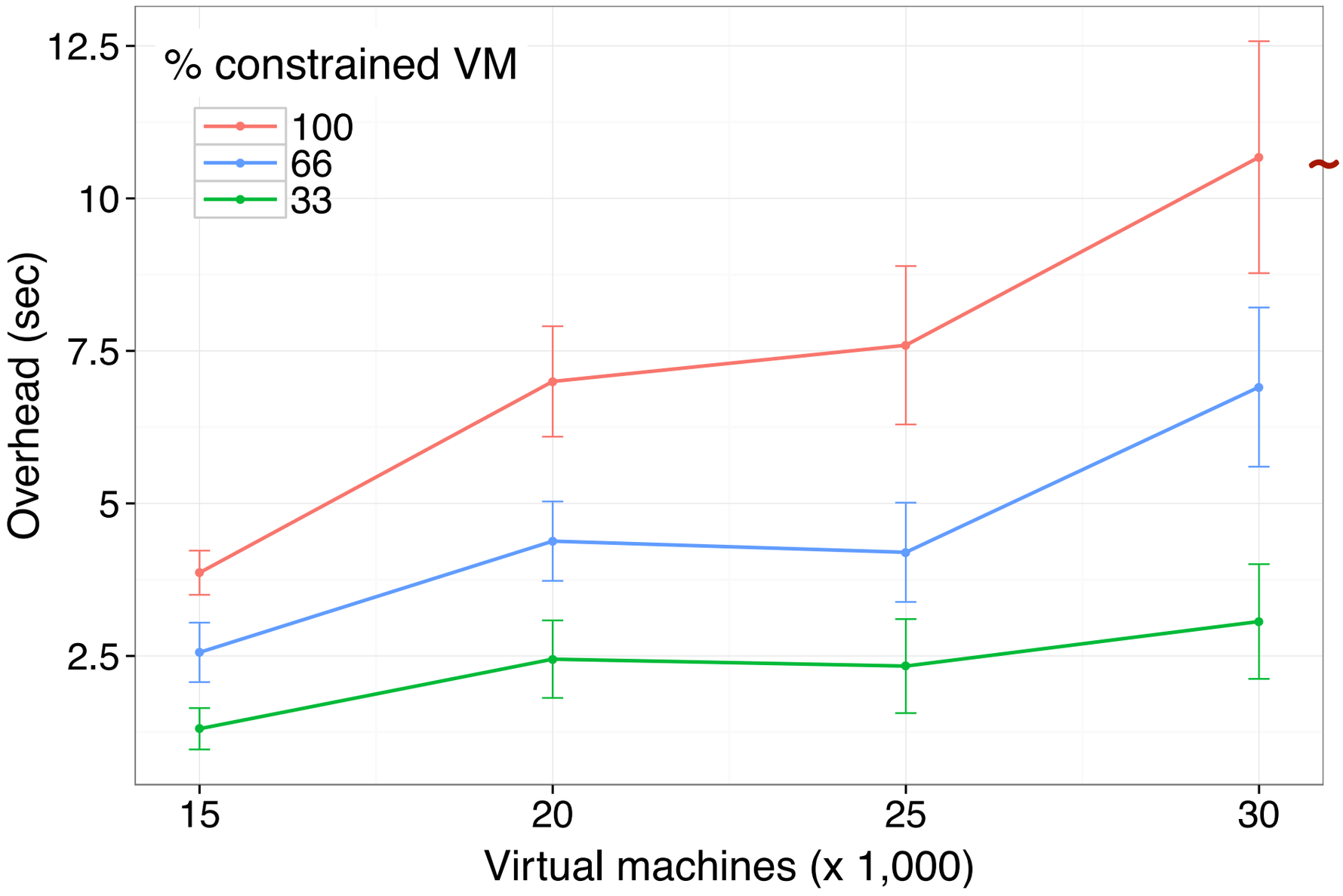


25%
EC2 average ?

70%
'ideal'

load

time overhead / user constraints



~ 30% in NR

CONCLUSION

flexibility as a key of automation

constraint programming as a solution

automated model composition

PERSPECTIVES

automated algorithm customisation
data and case studies

An Open-Source flexible virtual machine scheduler

ABOUT LIVE DEMO DOWNLOAD APIDOC