Resource management in IaaS cloud

Université Nice Sophia Antipolis

Fabien Hermenier

where to place the VMs

?

how much to allocate

1 What can a resource manager do for you

2 Models of VM scheduler

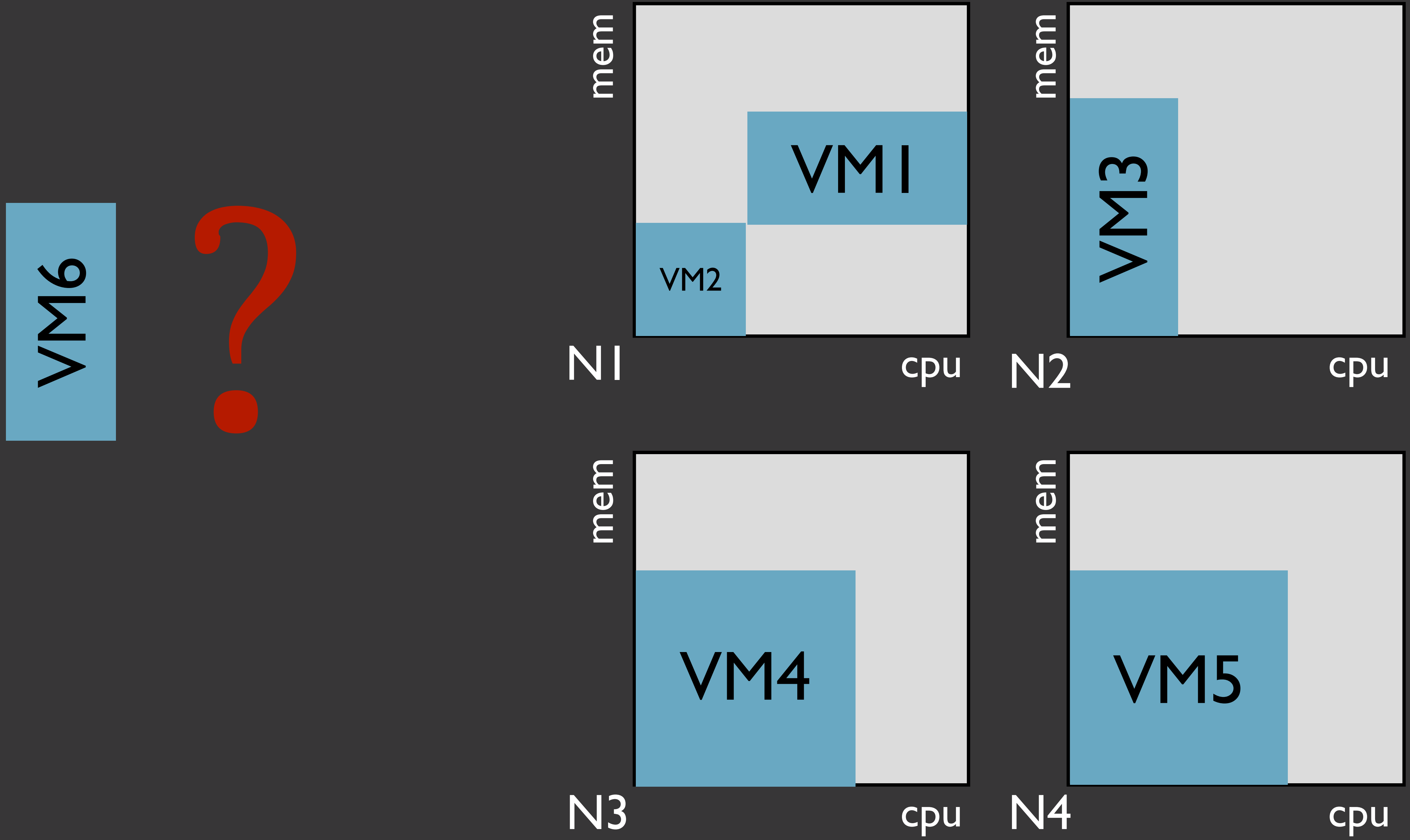
3 Implementing VM schedulers

1 What can a resource manager do for you

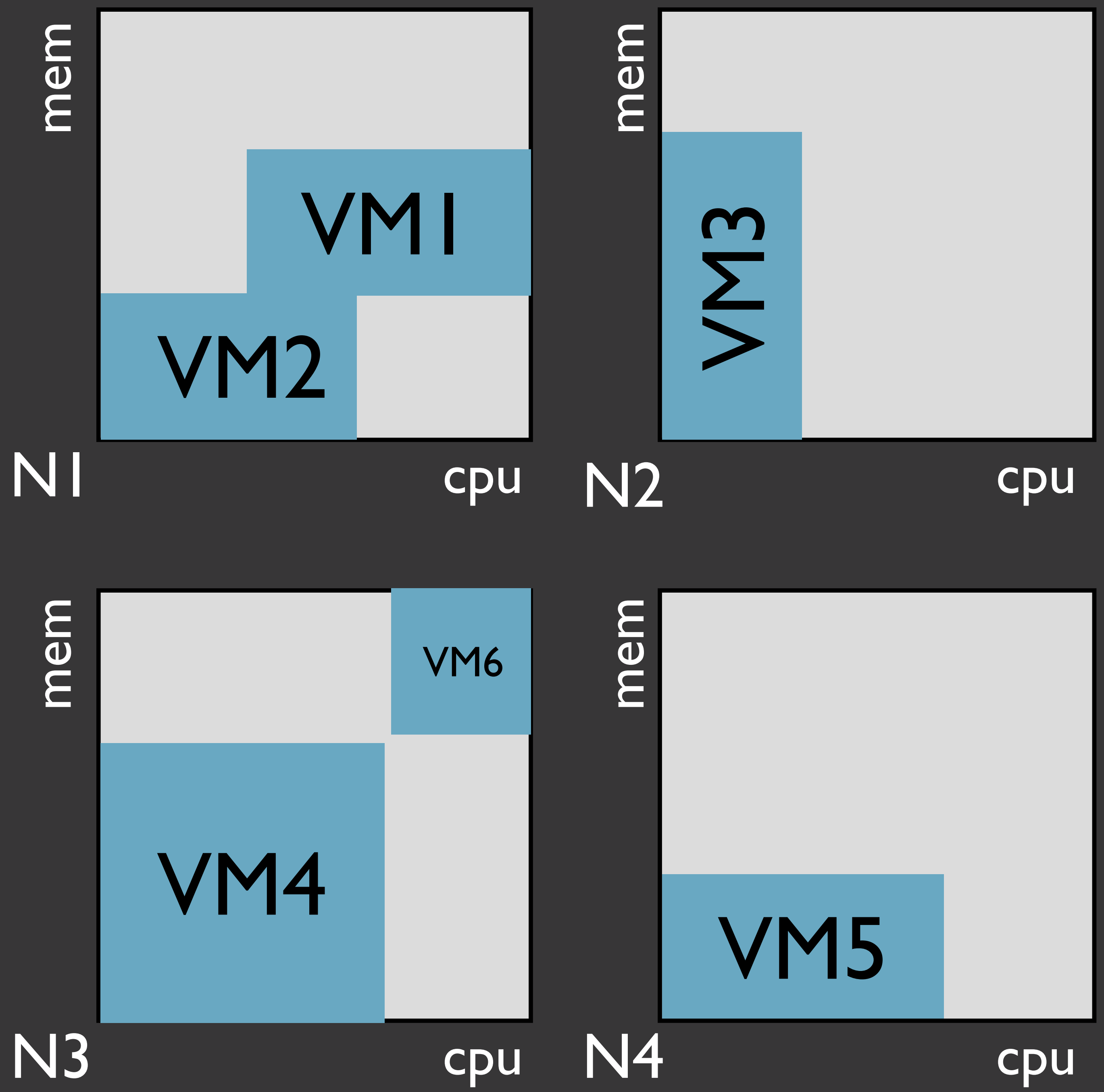
central piece of code

provide the awaited Quality of Service
address the management objectives

the RM should solve that



the RM should prevent that

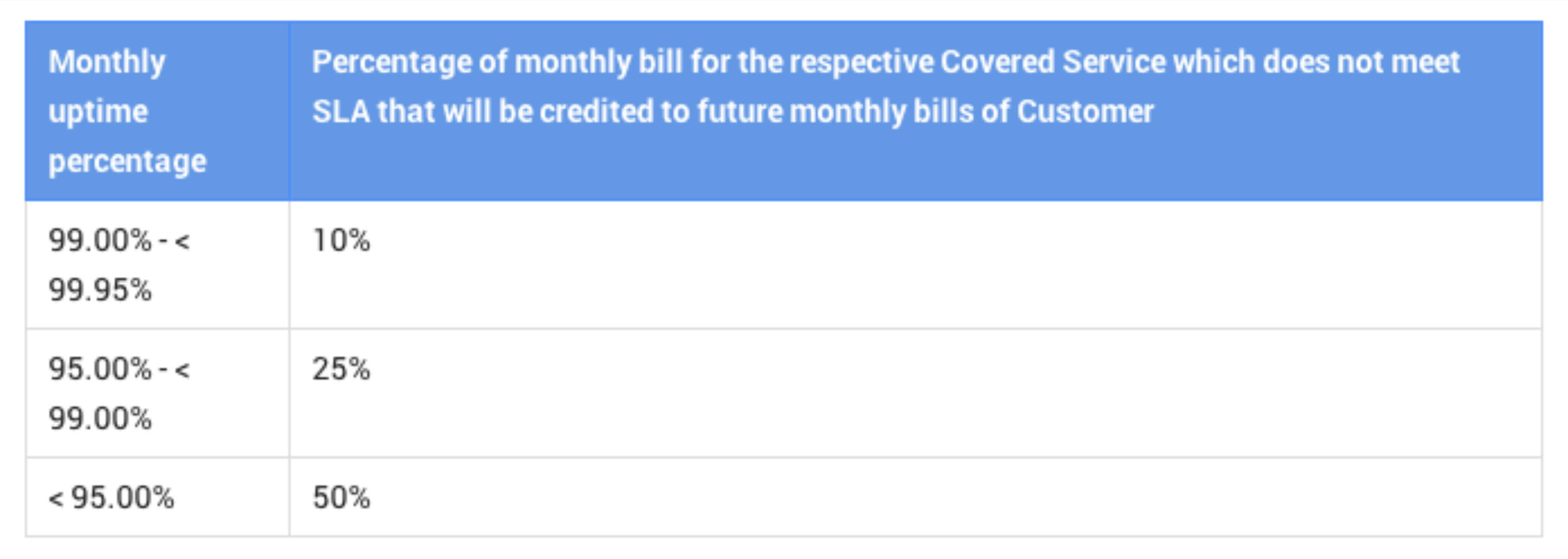


Service
Level
Agreement

a contract where a service is formally defined

performance indicator
MTTR
MTBF
availability (MTBF/MTTR + MTBF)
pricing
penalties
...

Google Cloud Platform



Monthly uptime percentage	Percentage of monthly bill for the respective Covered Service which does not meet SLA that will be credited to future monthly bills of Customer
99.00% - < 99.95%	10%
95.00% - < 99.00%	25%
< 95.00%	50%

uptime as the only
Key Performance Indicator (KPI) ?

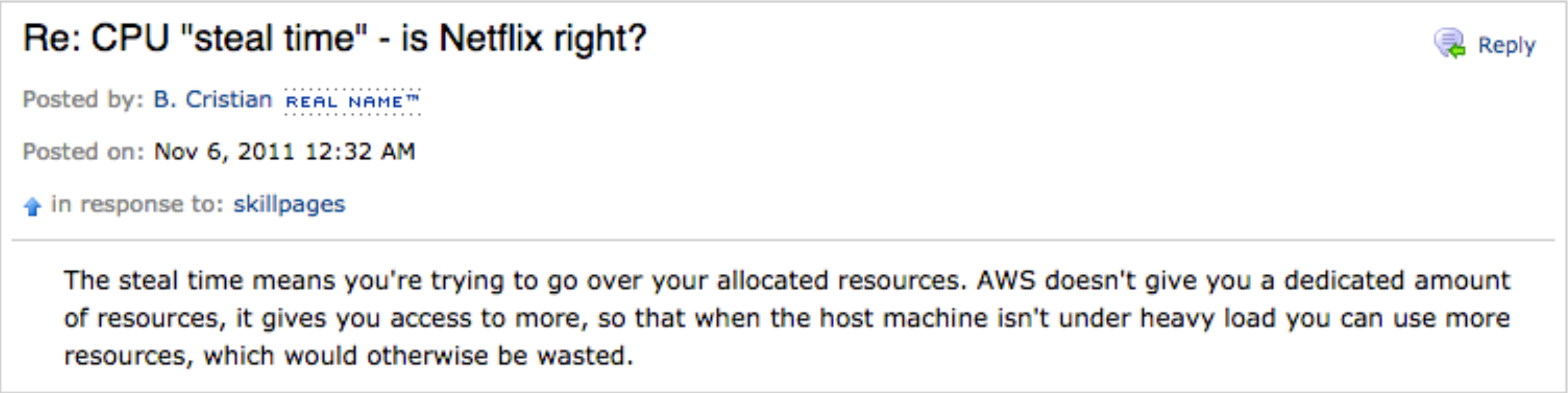
Cloud Performance SLAs

What customers want

guaranteed capacities (CPU, memory, bw)
guaranteed latencies/throughput (network, load balancer, disk)

What providers give

-



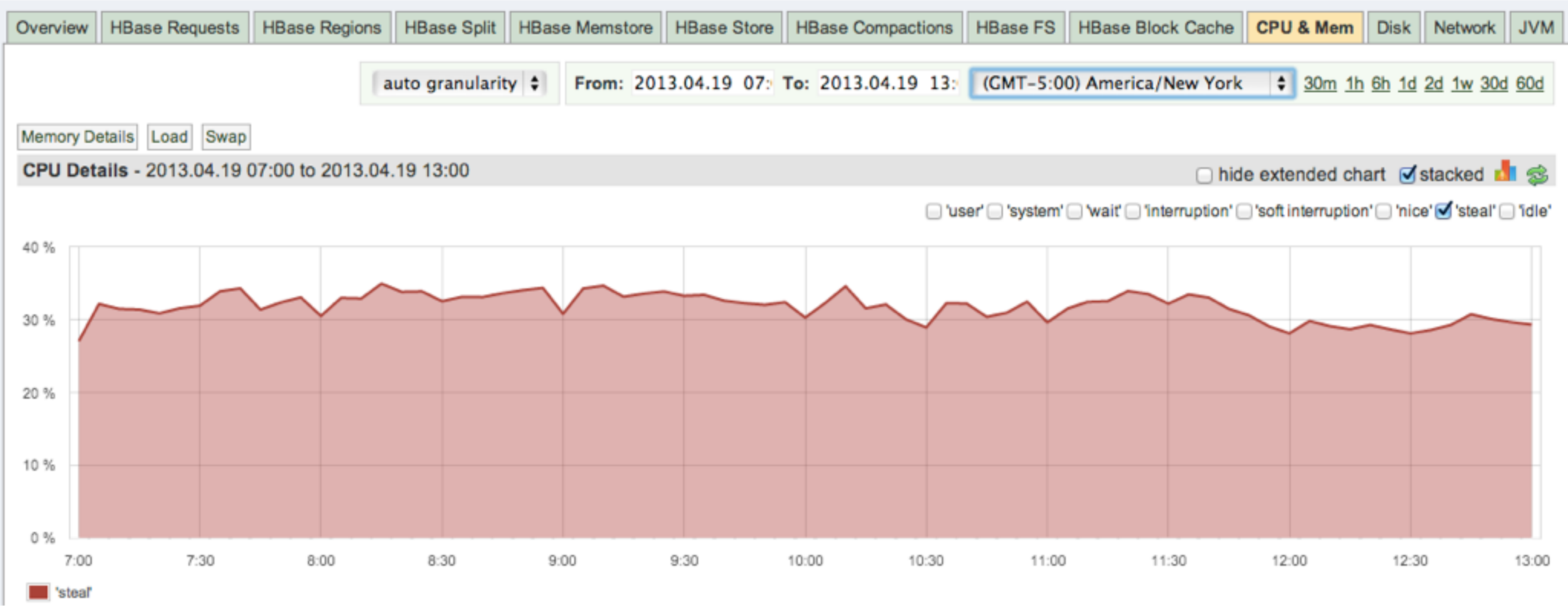
Re: CPU "steal time" - is Netflix right?

Posted by: B. Cristian REAL NAME™

Posted on: Nov 6, 2011 12:32 AM

in response to: skillpages

The steal time means you're trying to go over your allocated resources. AWS doesn't give you a dedicated amount of resources, it gives you access to more, so that when the host machine isn't under heavy load you can use more resources, which would otherwise be wasted.



auto granularity

From: 2013.04.19 07: To: 2013.04.19 13: (GMT-5:00) America/New York

CPU Details - 2013.04.19 07:00 to 2013.04.19 13:00

☐ hide extended chart ☑ stacked

☐ 'user' ☐ 'system' ☐ 'wait' ☐ 'interruption' ☐ 'soft interruption' ☐ 'nice' ☑ 'steal' ☐ 'idle'

the objective

provider side

$min(x)$ or $max(x)$

atomic objectives

$$min(\text{penalties})$$

$$min(\text{Total Cost Ownership})$$

$$min(\text{unbalance})$$

$$\ldots$$

composite objectives

using weights

$$min(\alpha x + \beta\ y)$$

How to estimate coefficients ?

useful to model sth. you don’t understand ?

$$min(\alpha\ TCO + \beta\ VIOLATIONS)$$

€ as a common quantifier:

$$max(REVENUES)$$

?

Should a client ask for an objective

?

Should a client ask for an objective

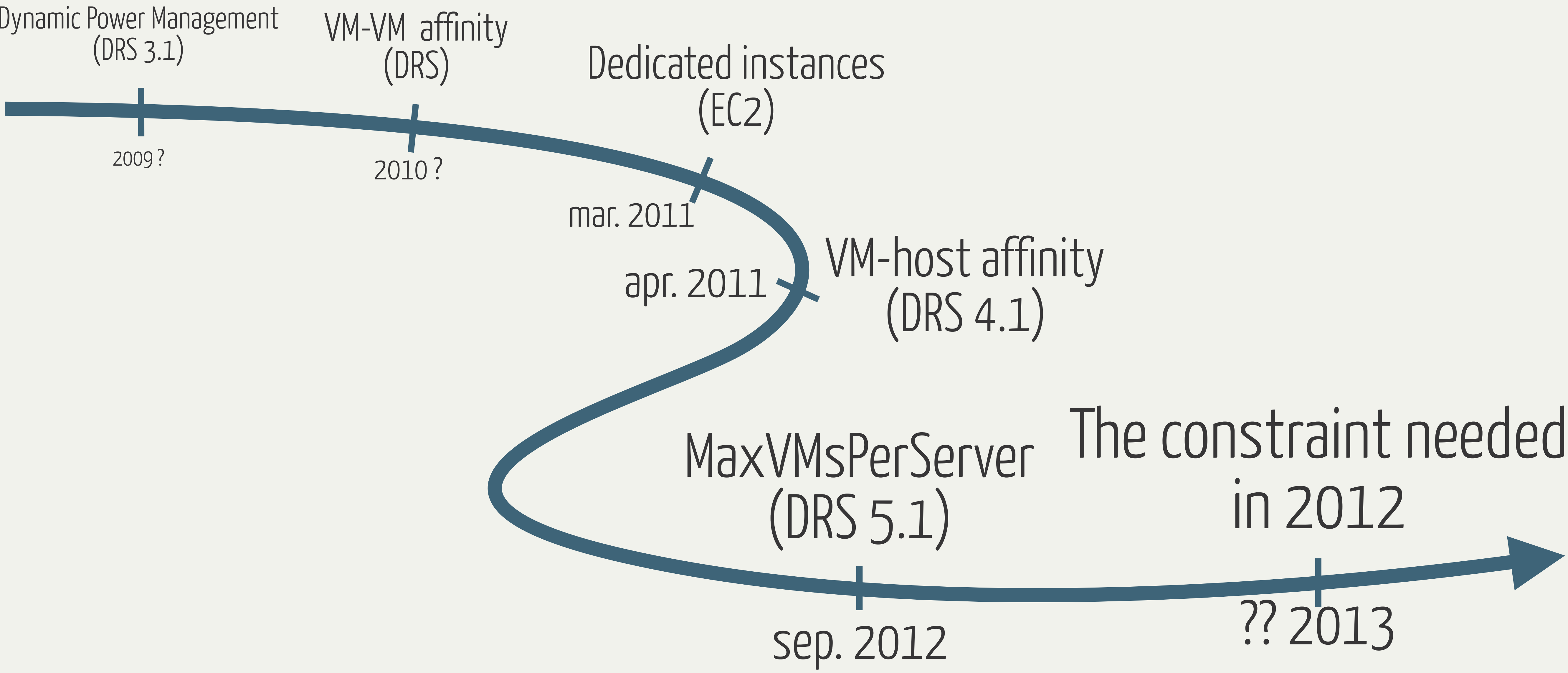
no if the provider cannot prove it is achieved

one can check the latency is below a threshold

proving a latency is minimal ...

SLAs and objectives are evolving

there is no holy grail



Dynamic Power Management
(DRS 3.1)
2009 ?

VM-VM affinity
(DRS)
2010 ?

Dedicated instances
(EC2)
mar. 2011

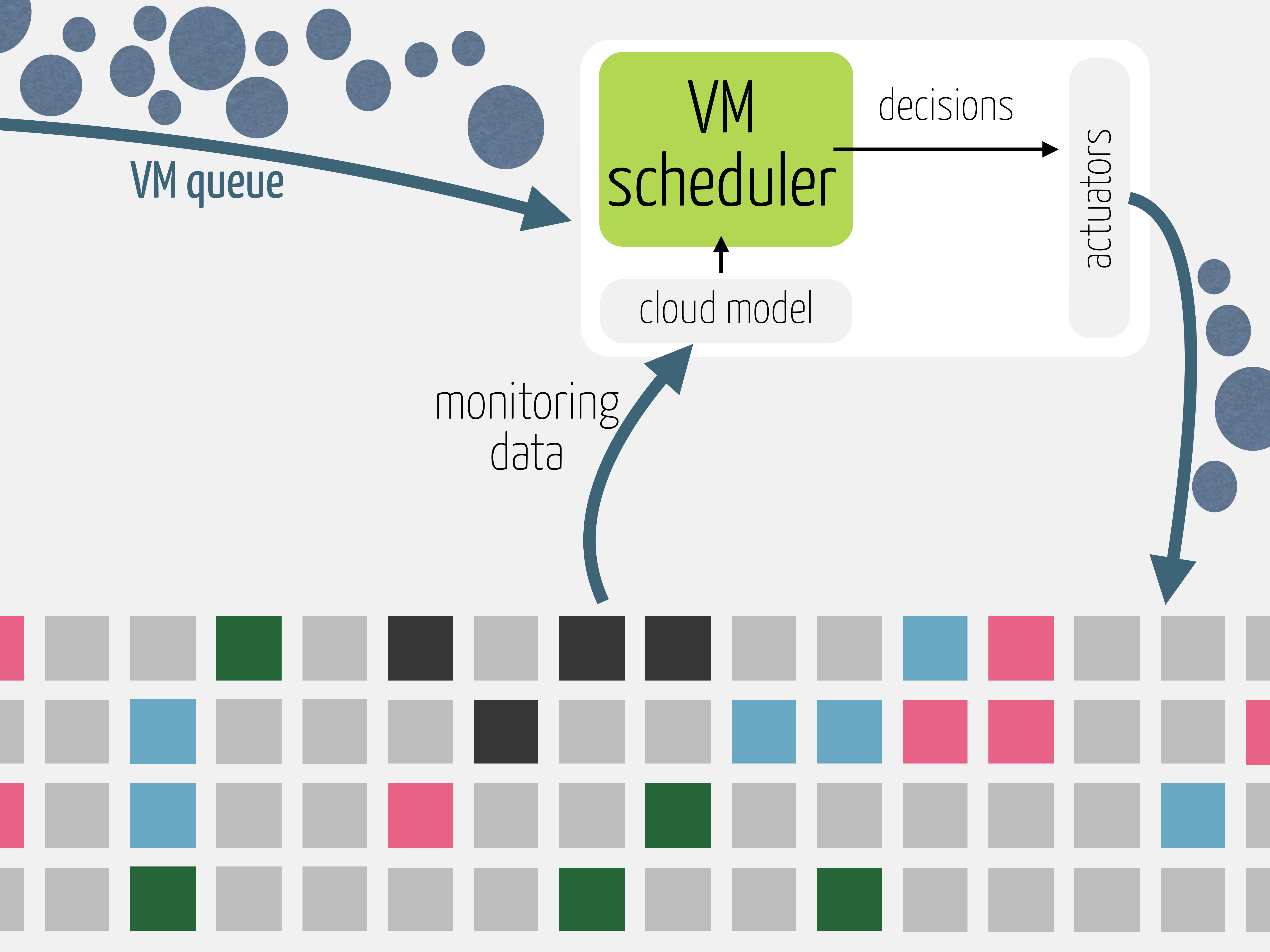
VM-host affinity
(DRS 4.1)
apr. 2011

MaxVMsPerServer
(DRS 5.1)
sep. 2012

The constraint needed
in 2012
?? 2013

2

Models of VM schedulers

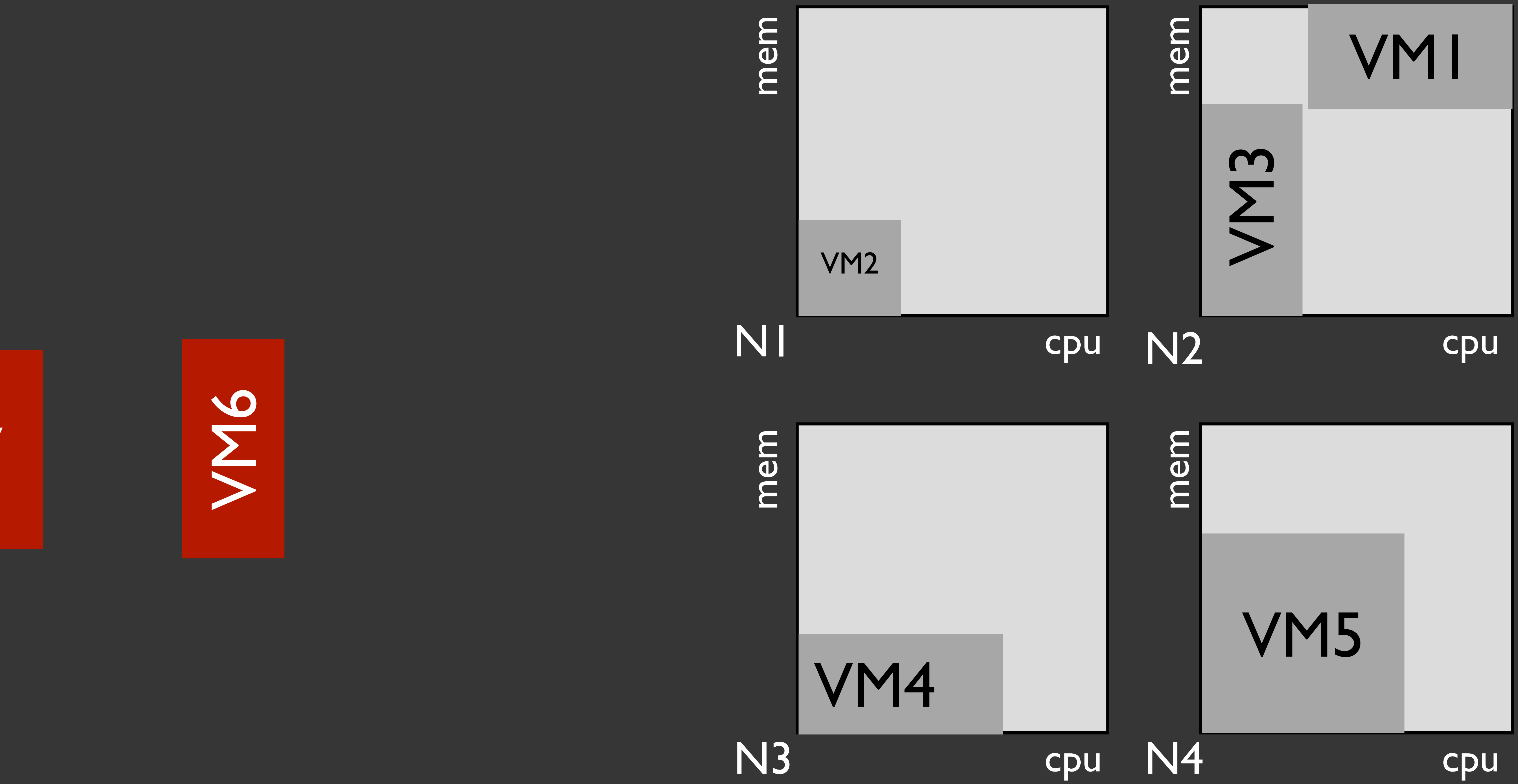

VM queue
VM scheduler
decisions
actuators
cloud model
monitoring data

scheduling models

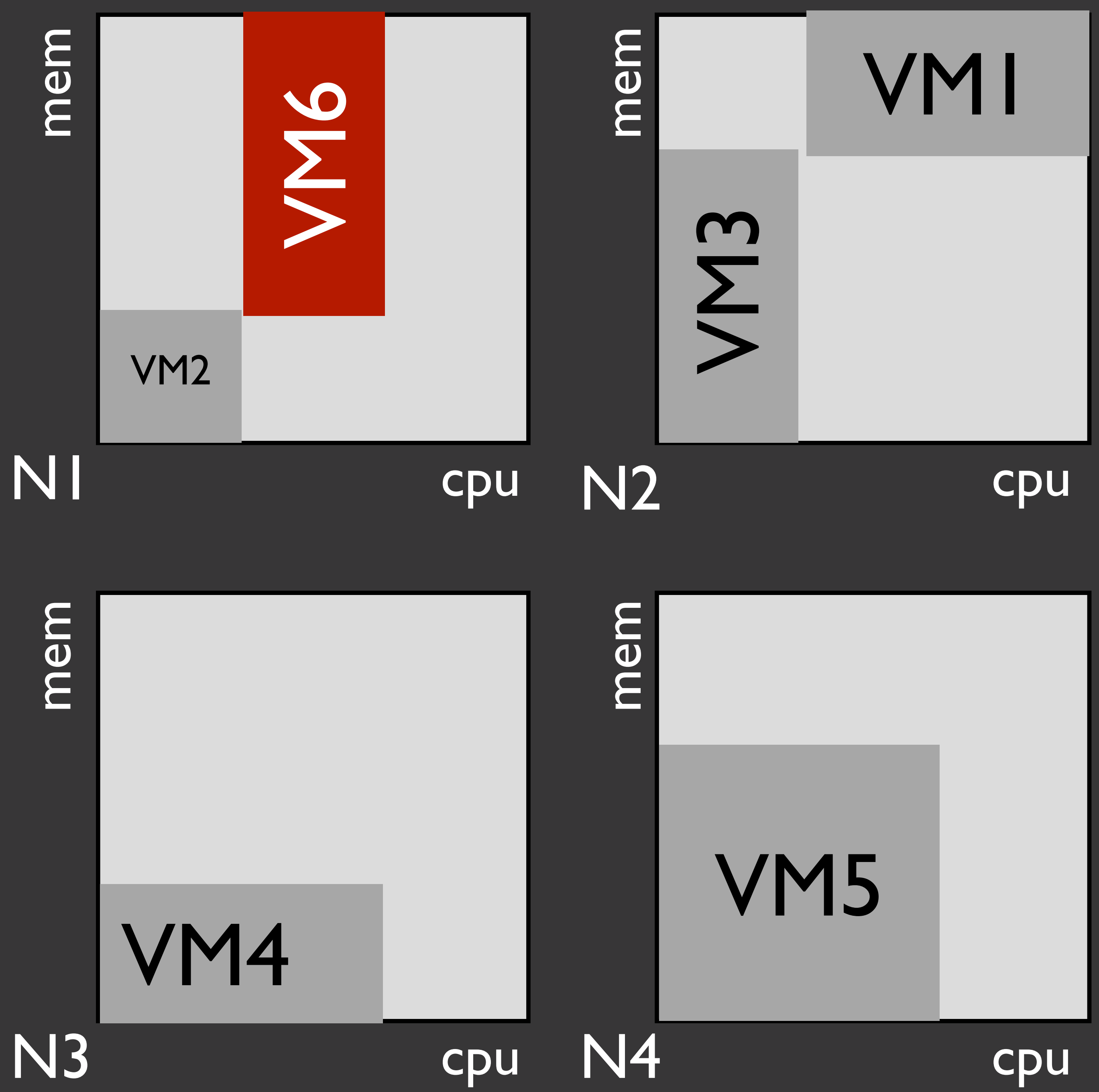
static
dynamic
schedulers

static
dynamic
resource allocation

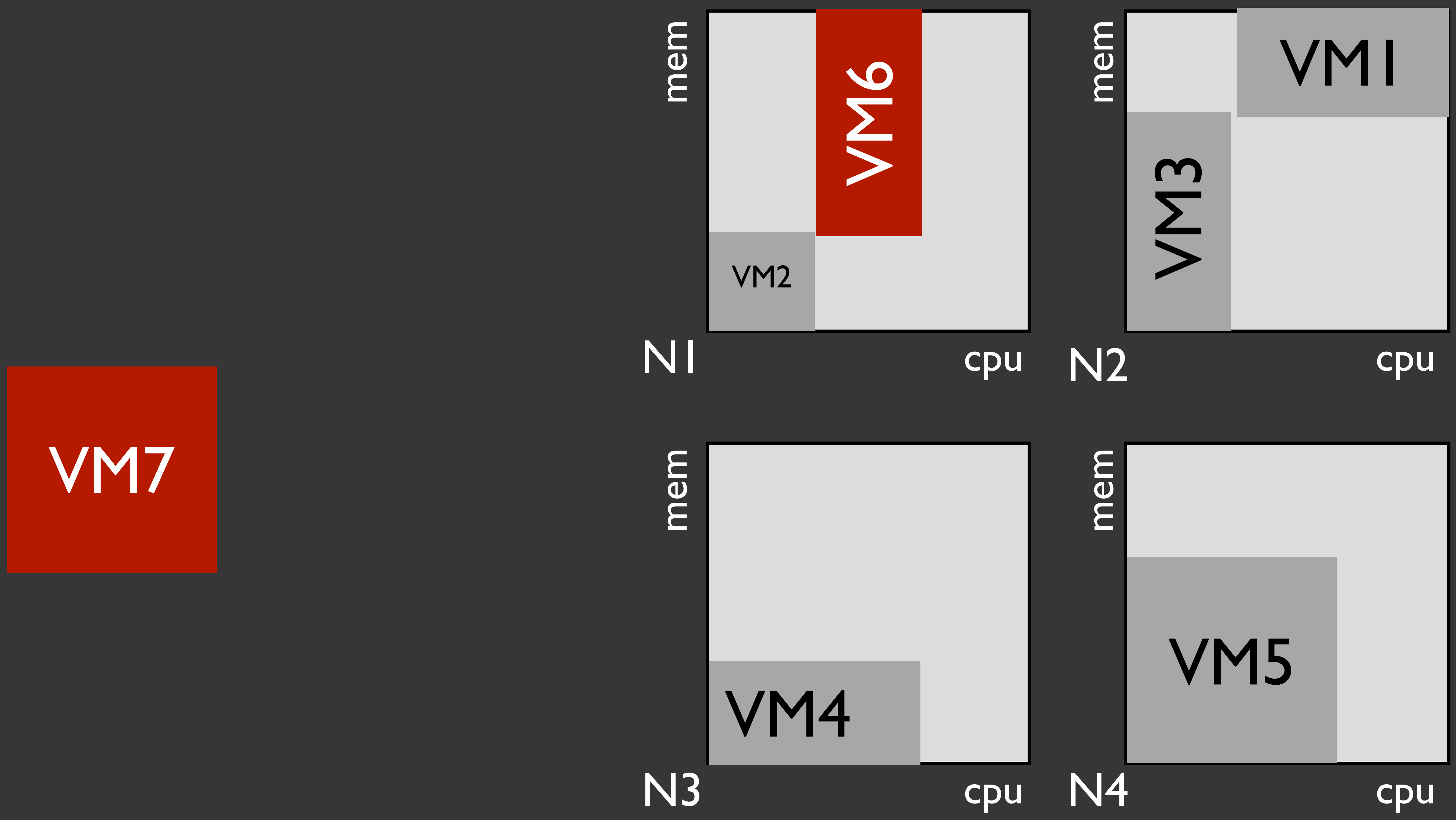
static schedulers



static schedulers



static schedulers

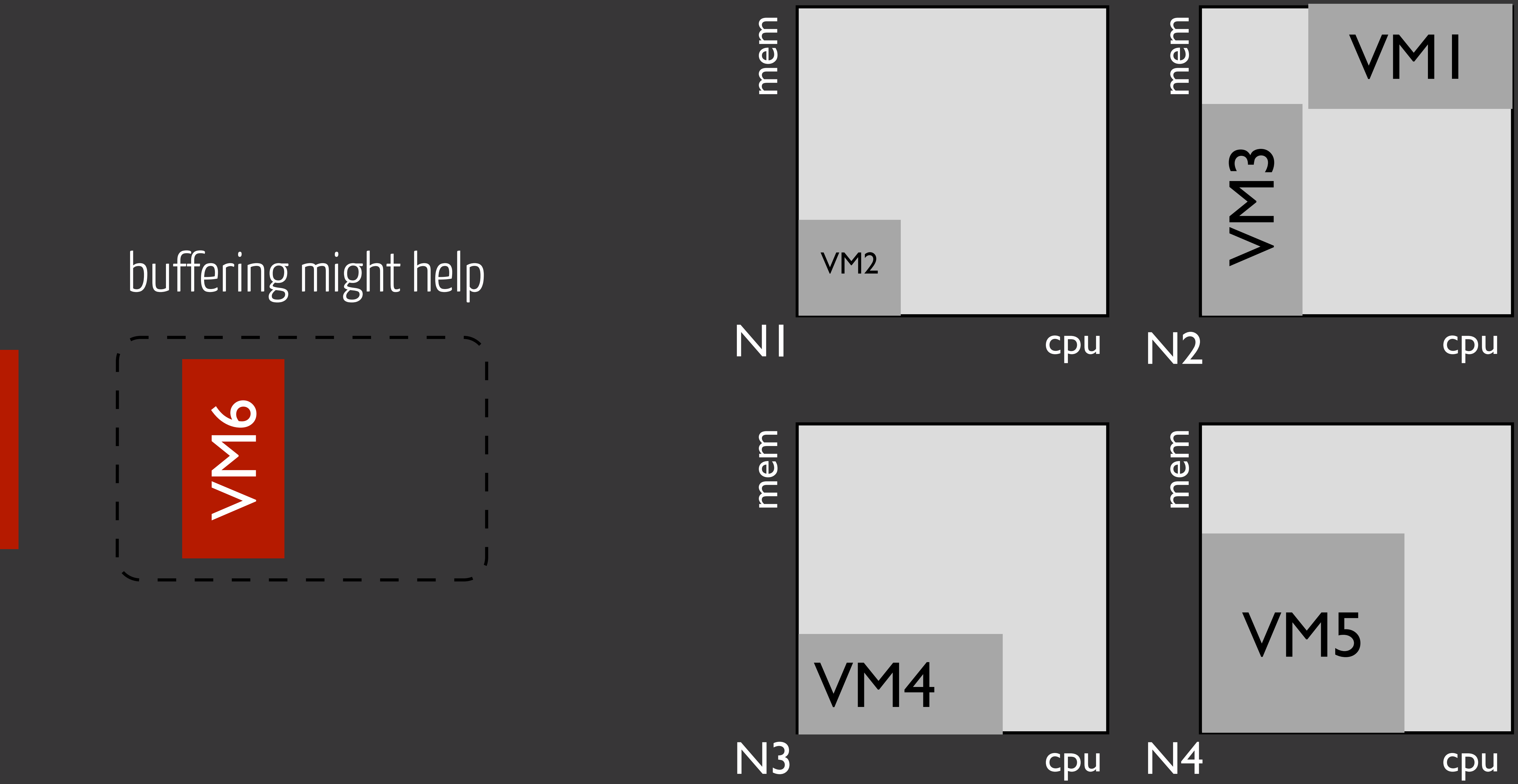


Combinatorial problem !

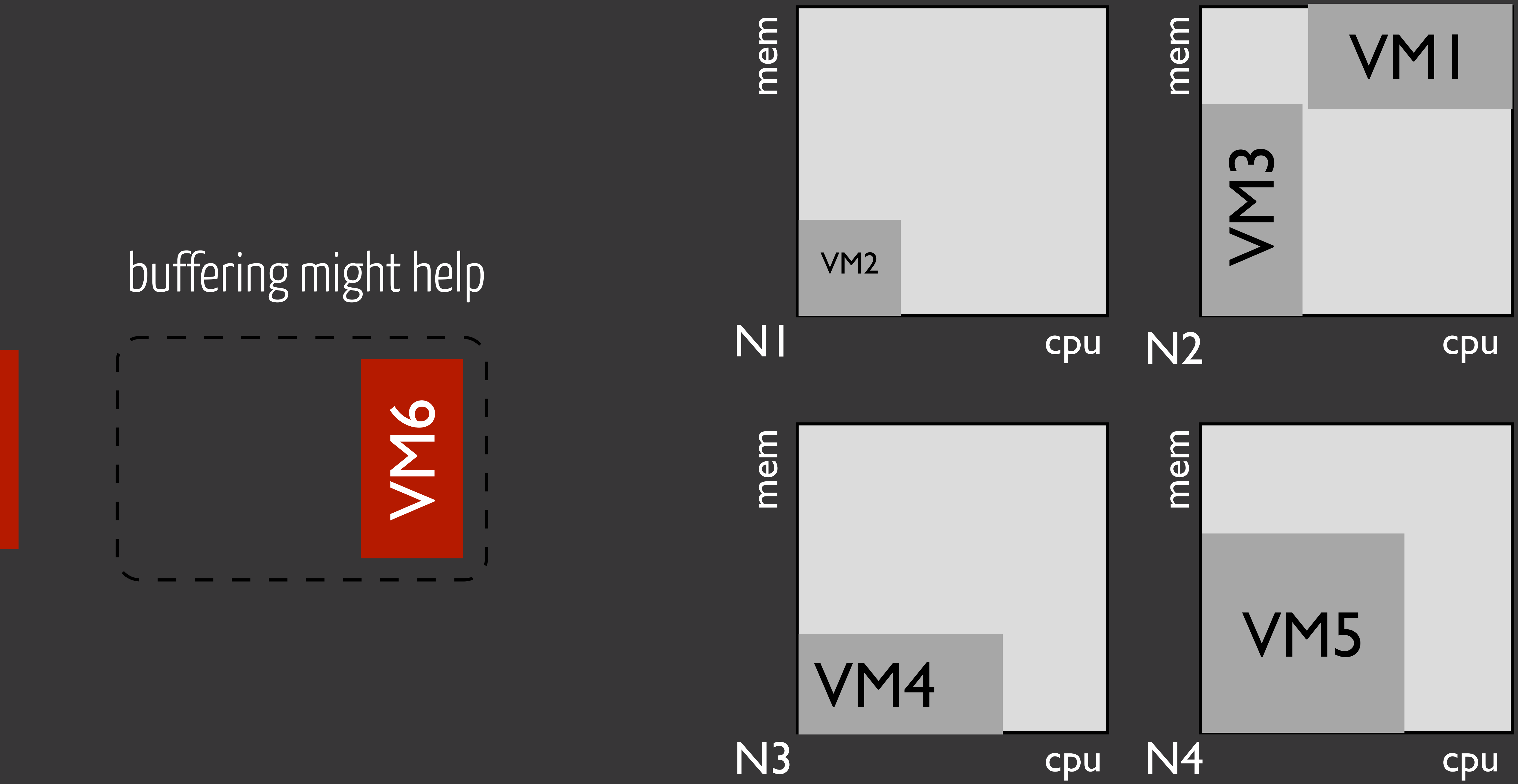
past decisions impact the future ones

static schedulers

buffering might help



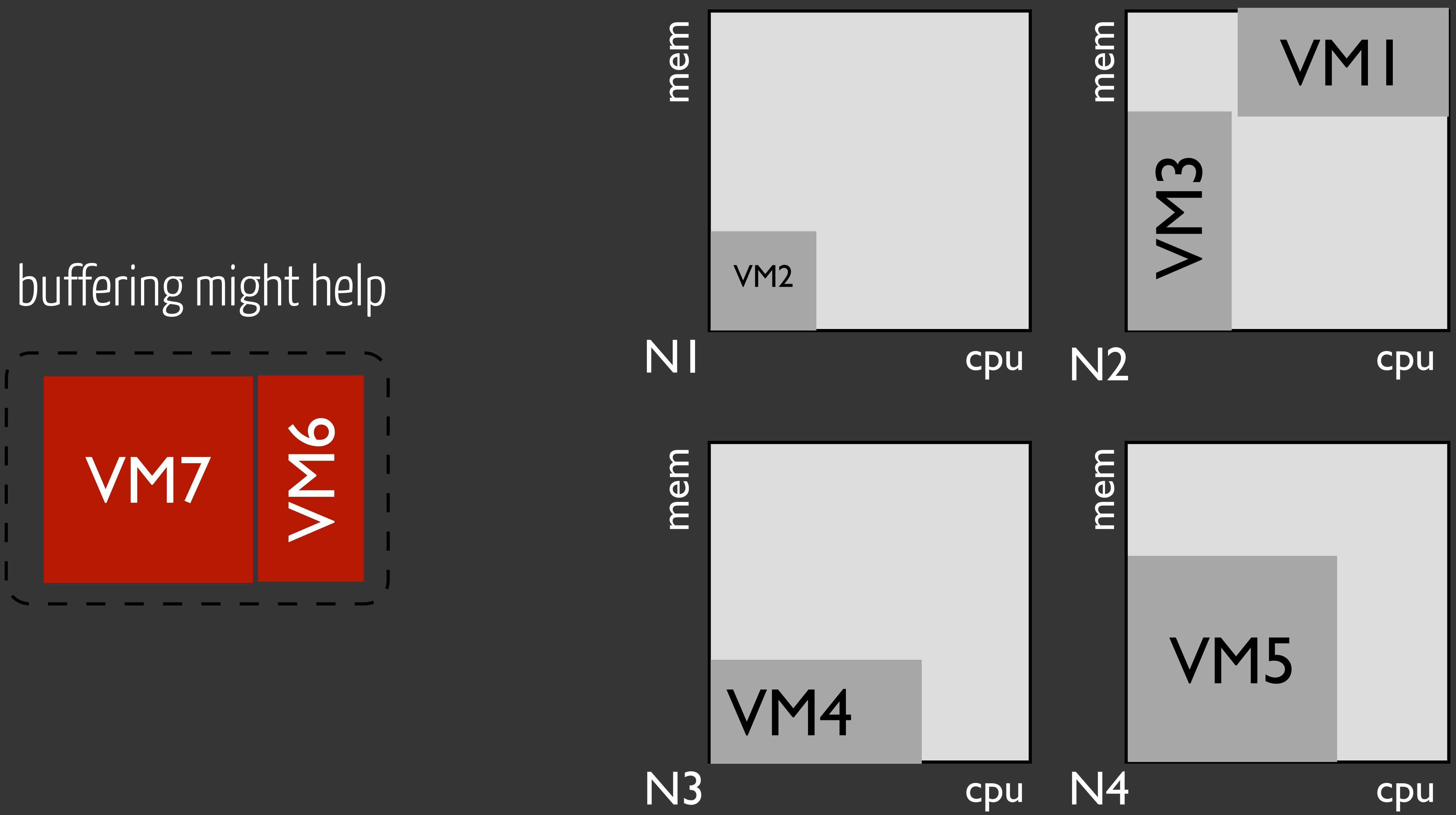
static schedulers



buffering might help

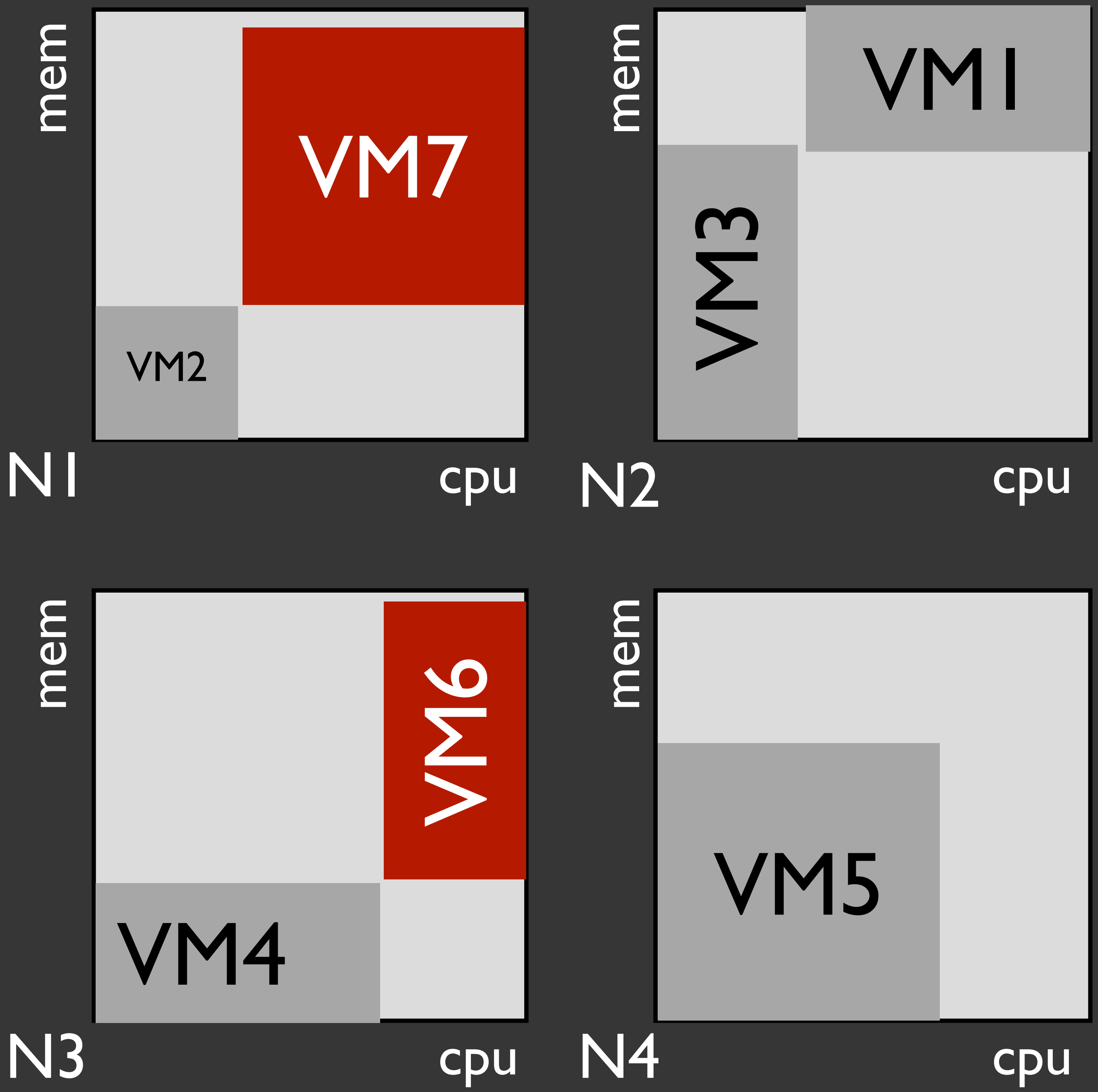
static schedulers

buffering might help



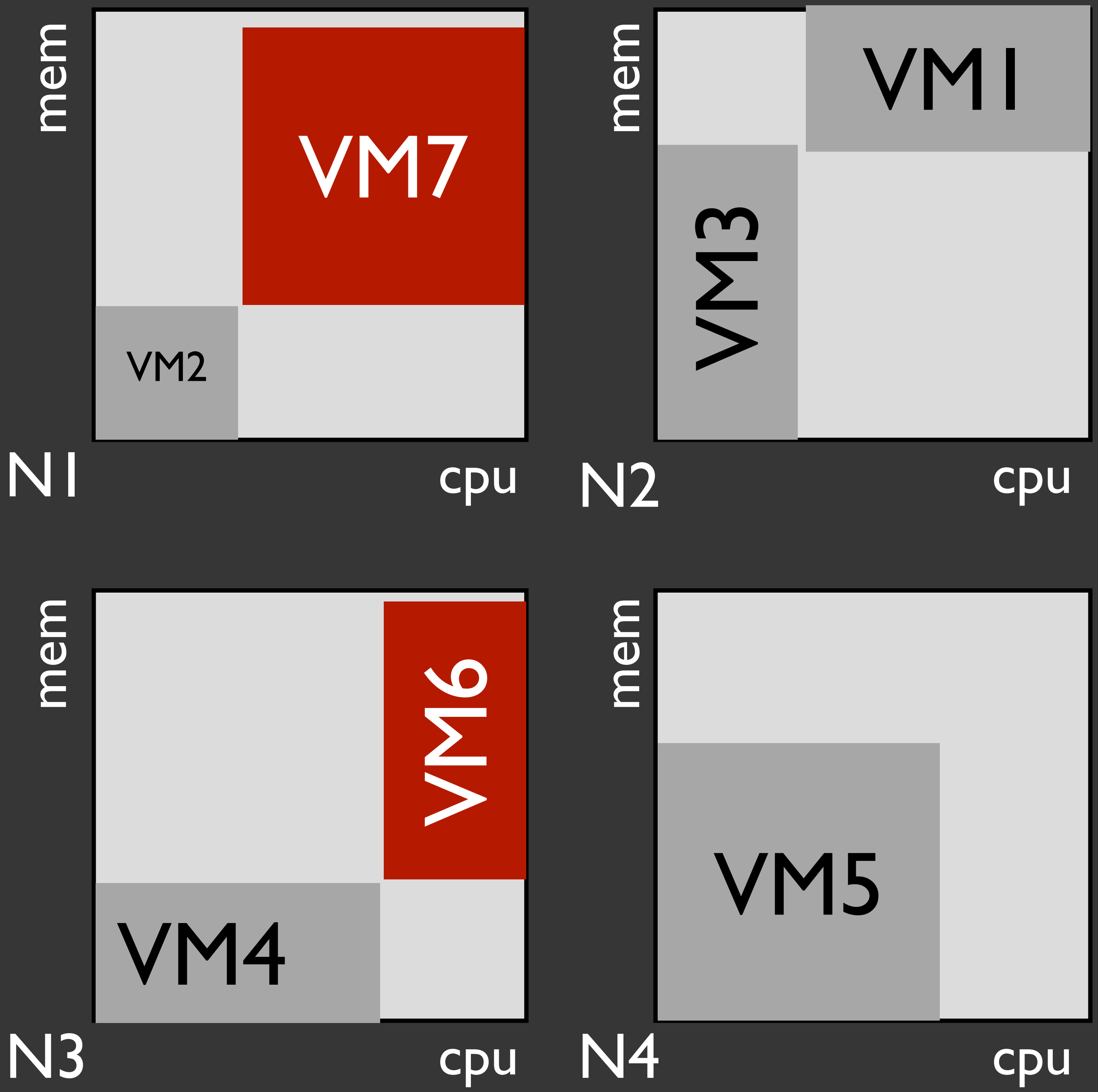
static schedulers

buffering might help



static schedulers

buffering might help



/!\ incoming rate starvation

static schedulers

manipulate the VM queue only
react on VM arrival

static schedulers

pros

easy to model
simple standard actions
manage a few elements

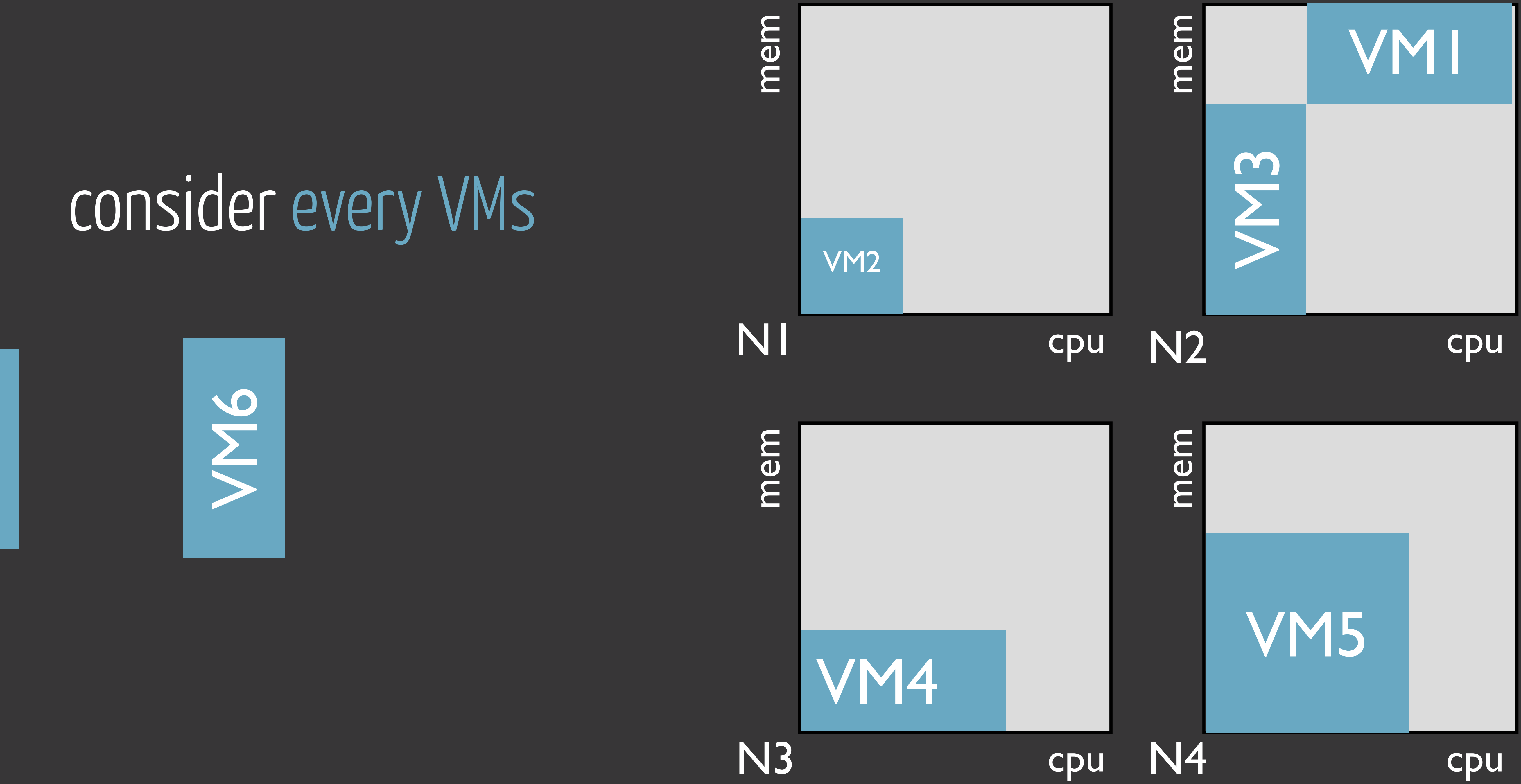
static schedulers

cons

hard to perform fine
grain optimisation

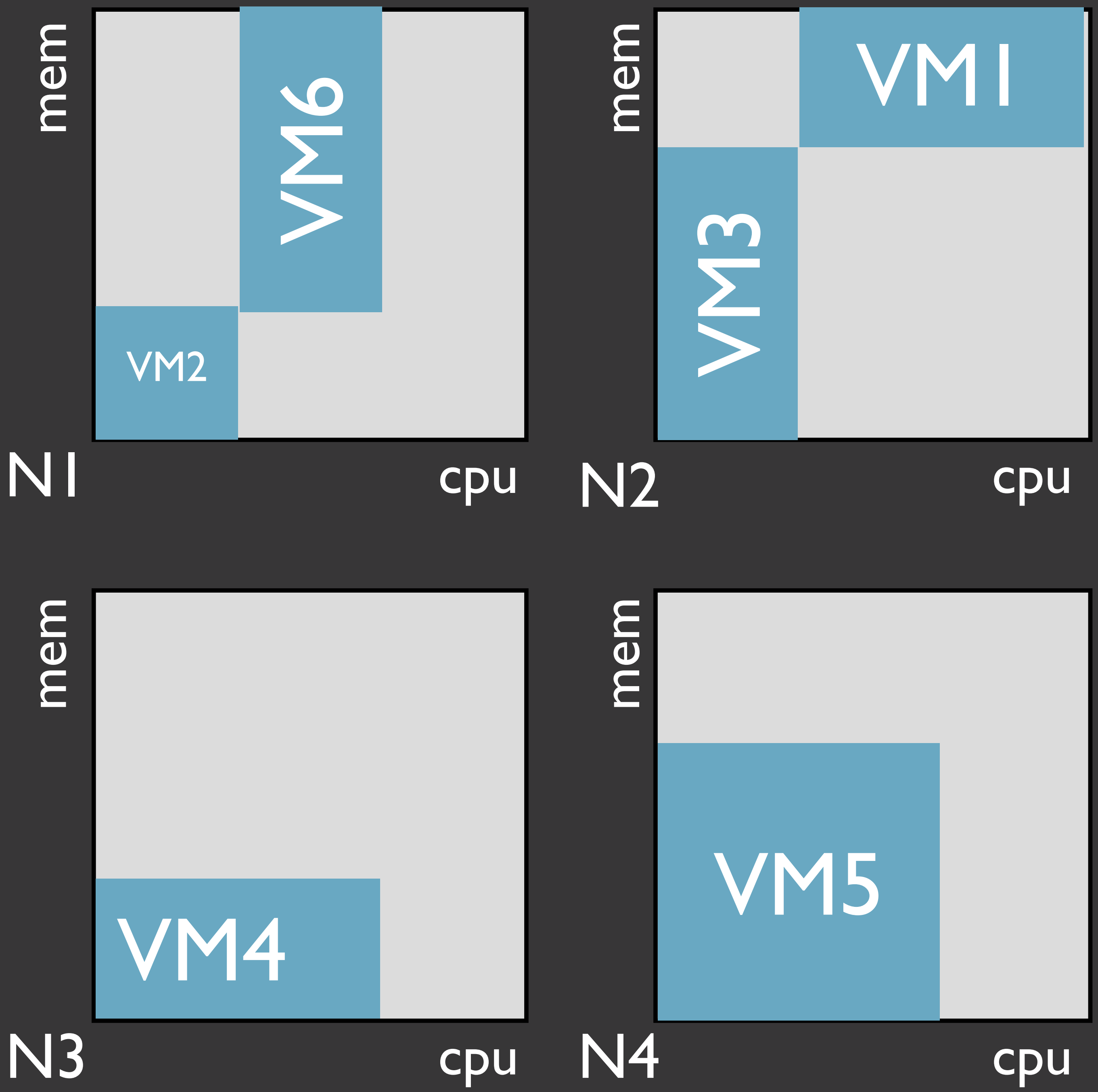
dynamic schedulers

consider every VMs



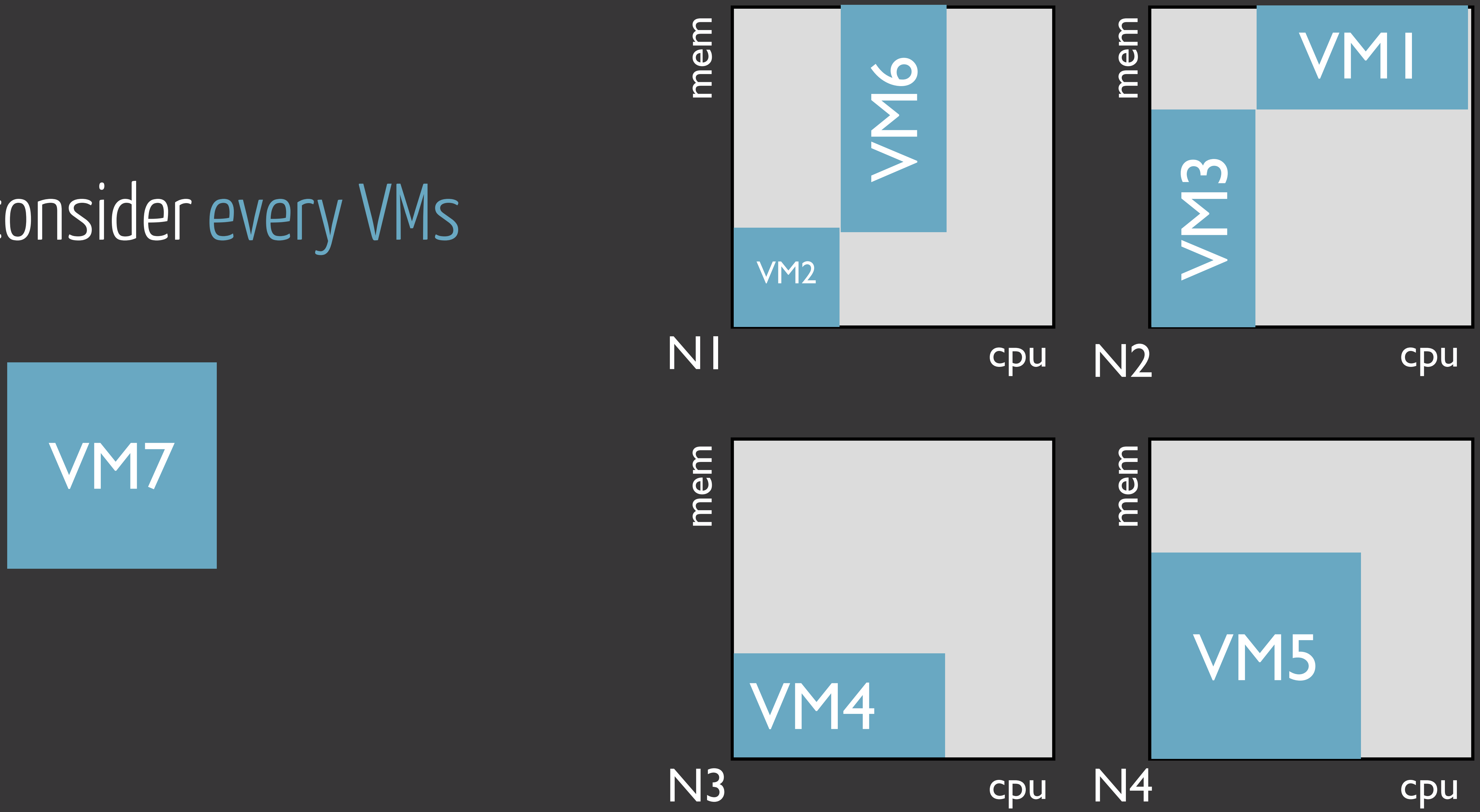
dynamic schedulers

consider every VMs



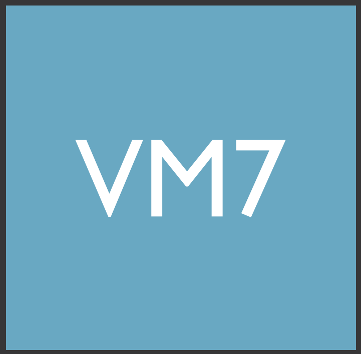
dynamic schedulers

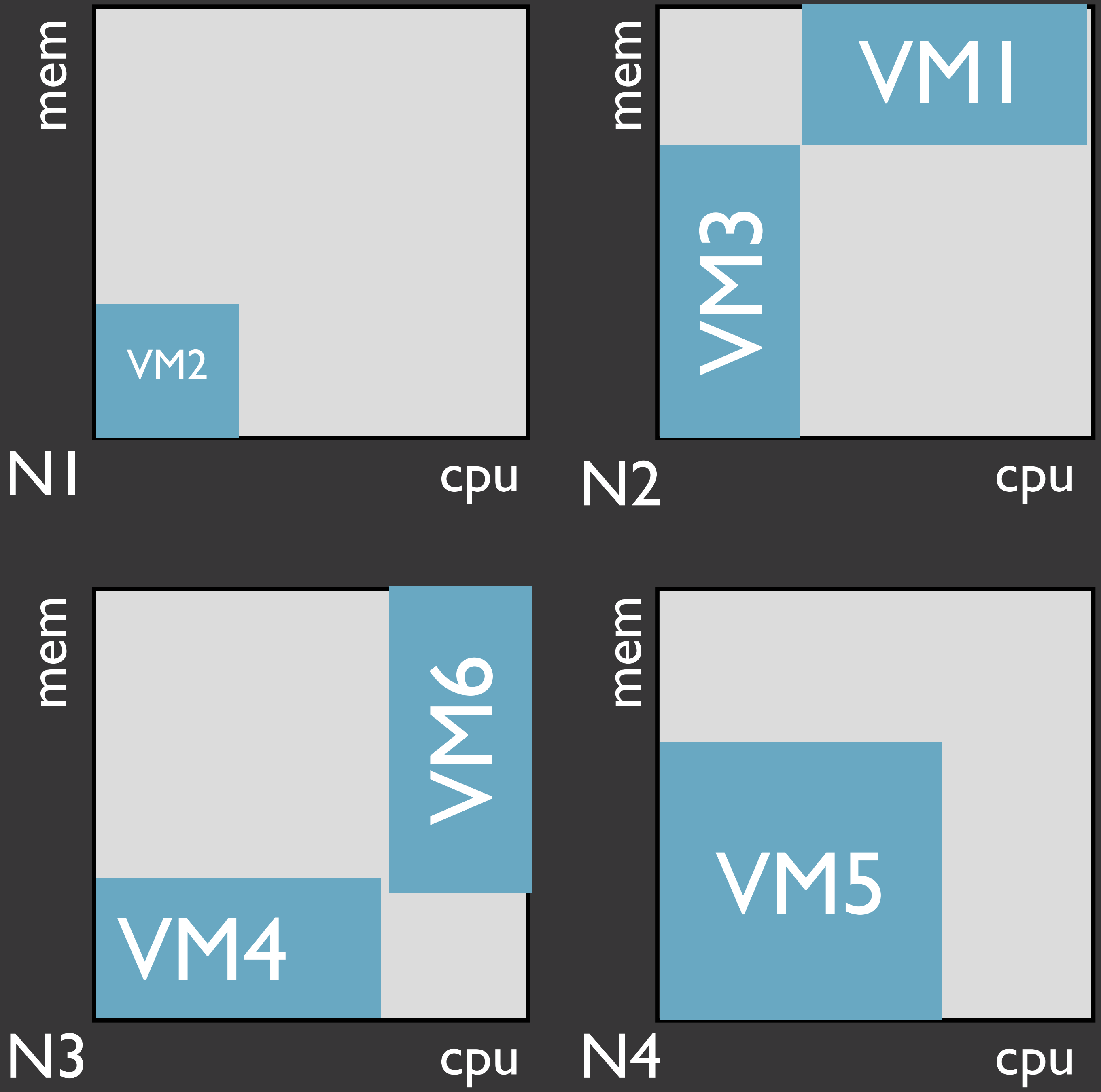
consider every VMs



dynamic schedulers

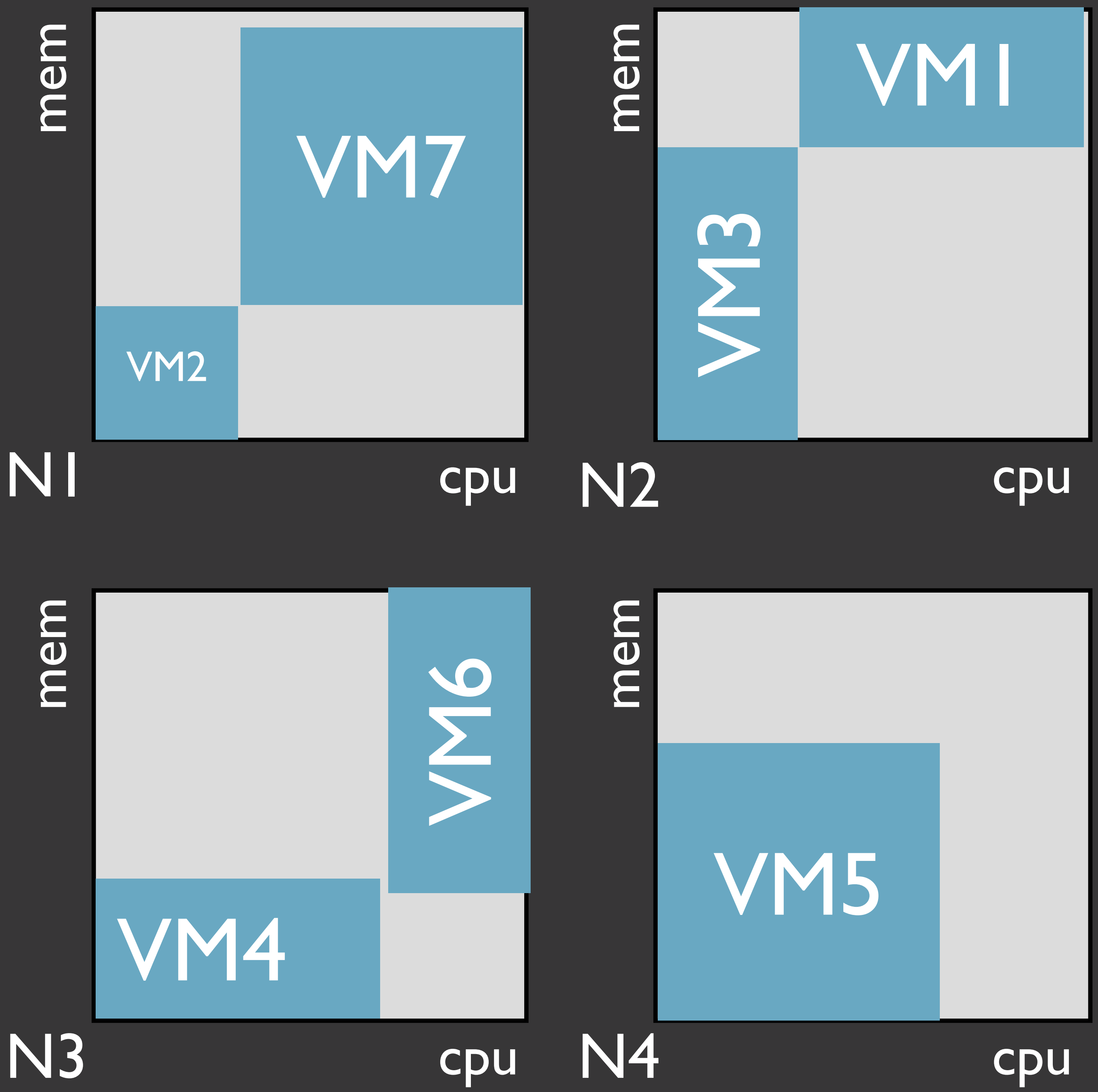
consider every VMs





dynamic schedulers

consider every VMs

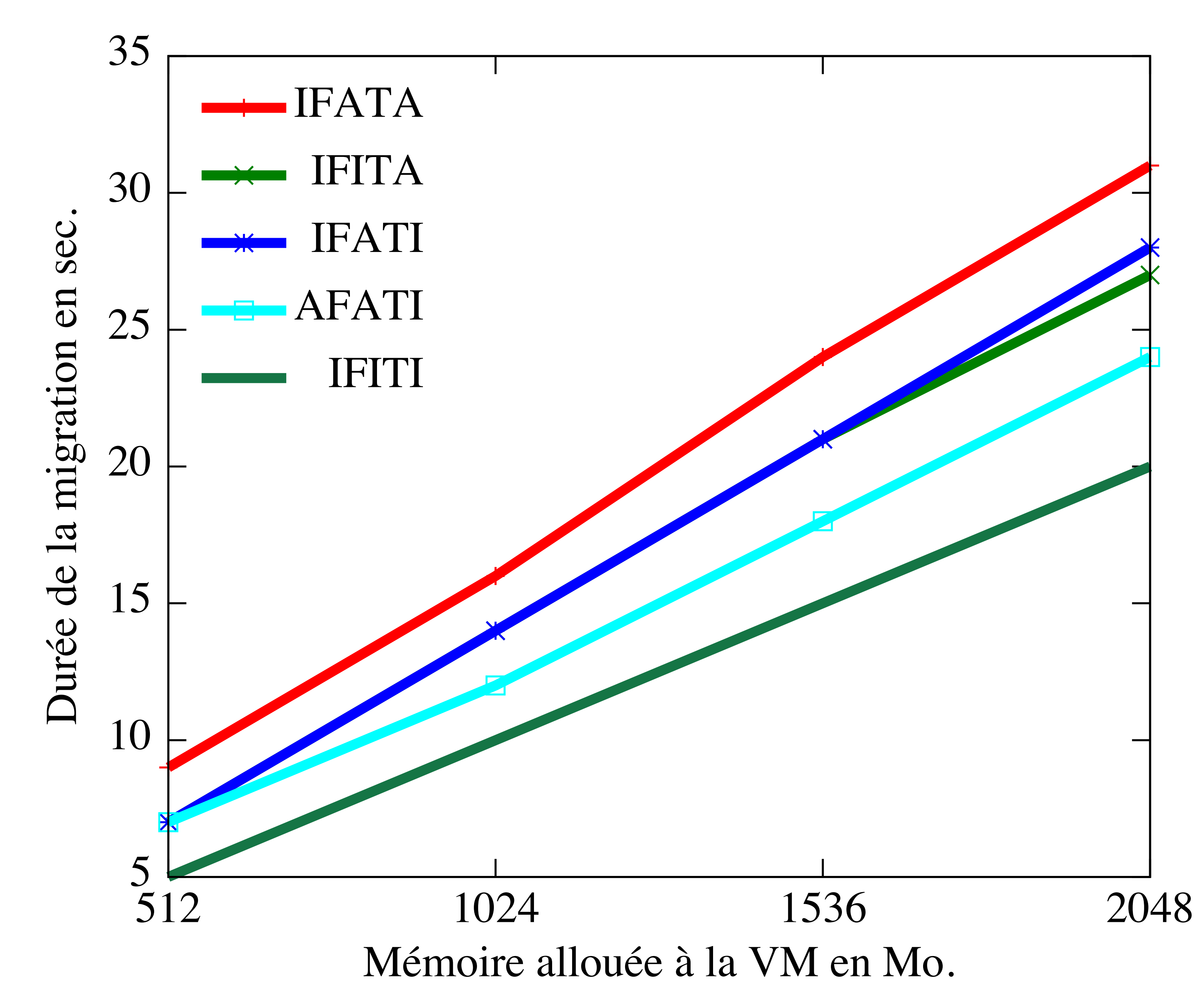


dynamic schedulers

manipulate the VM queue
reconfigure the schedule with migrations

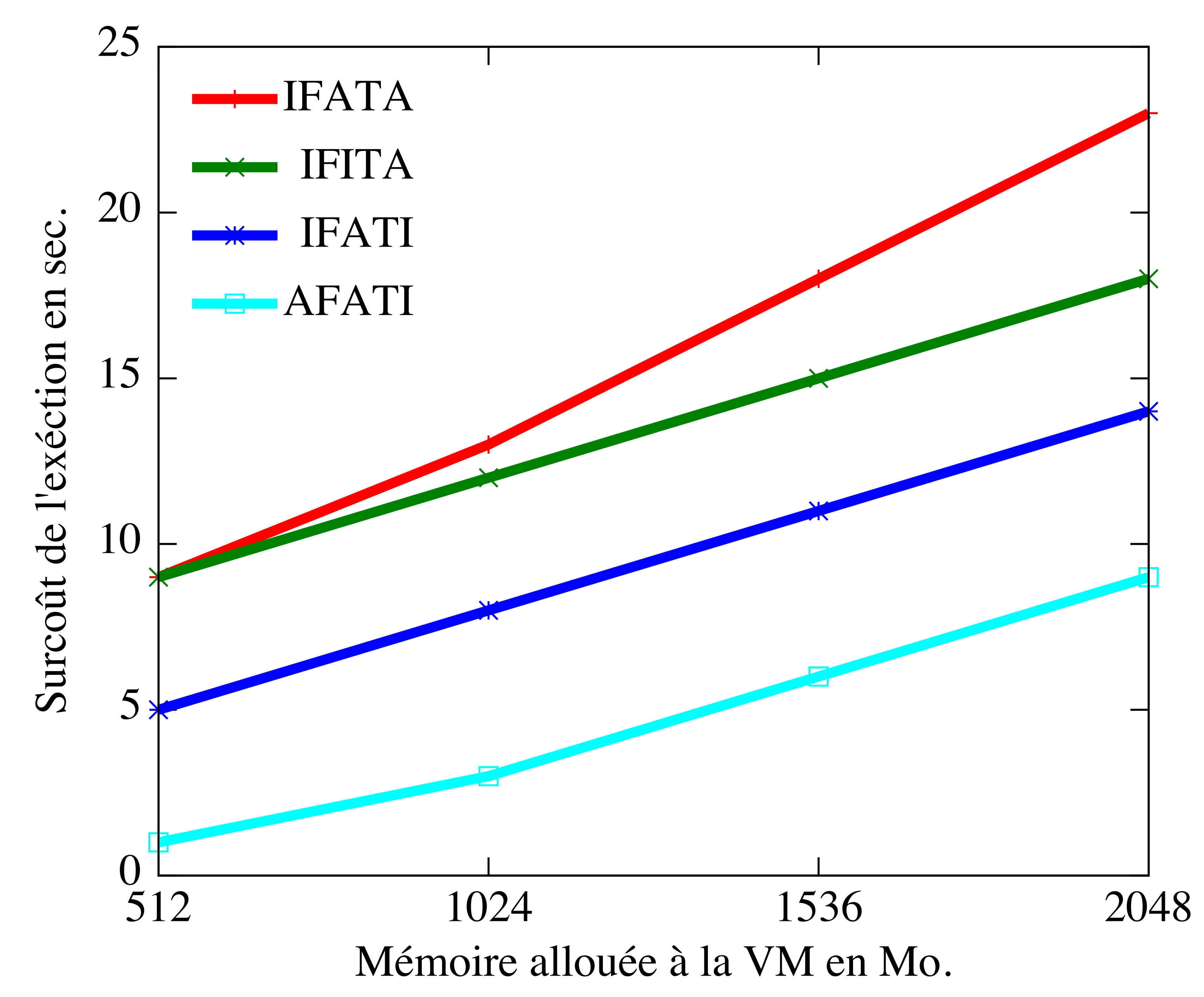
dynamic schedulers

a migration takes time



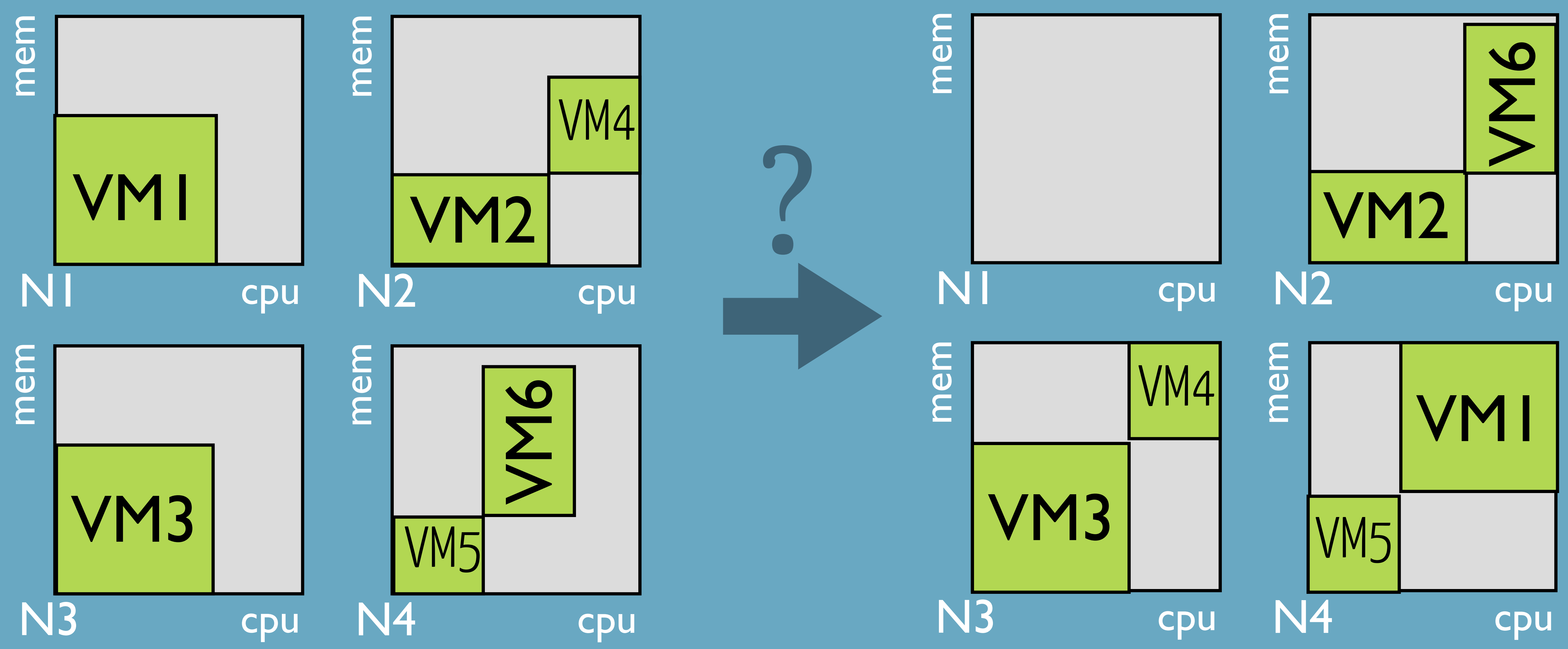
dynamic schedulers

a migration impacts performance



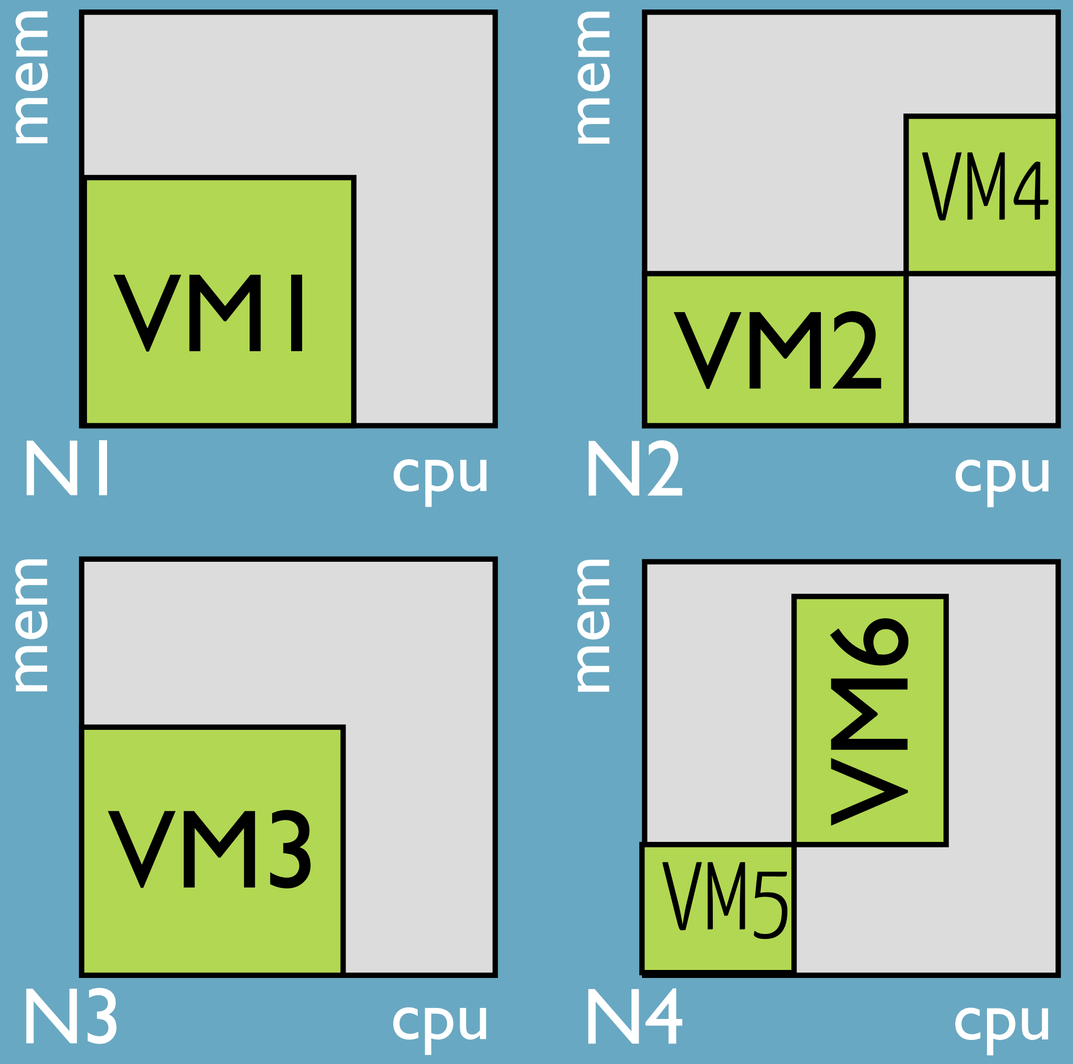
dynamic schedulers

dependency management



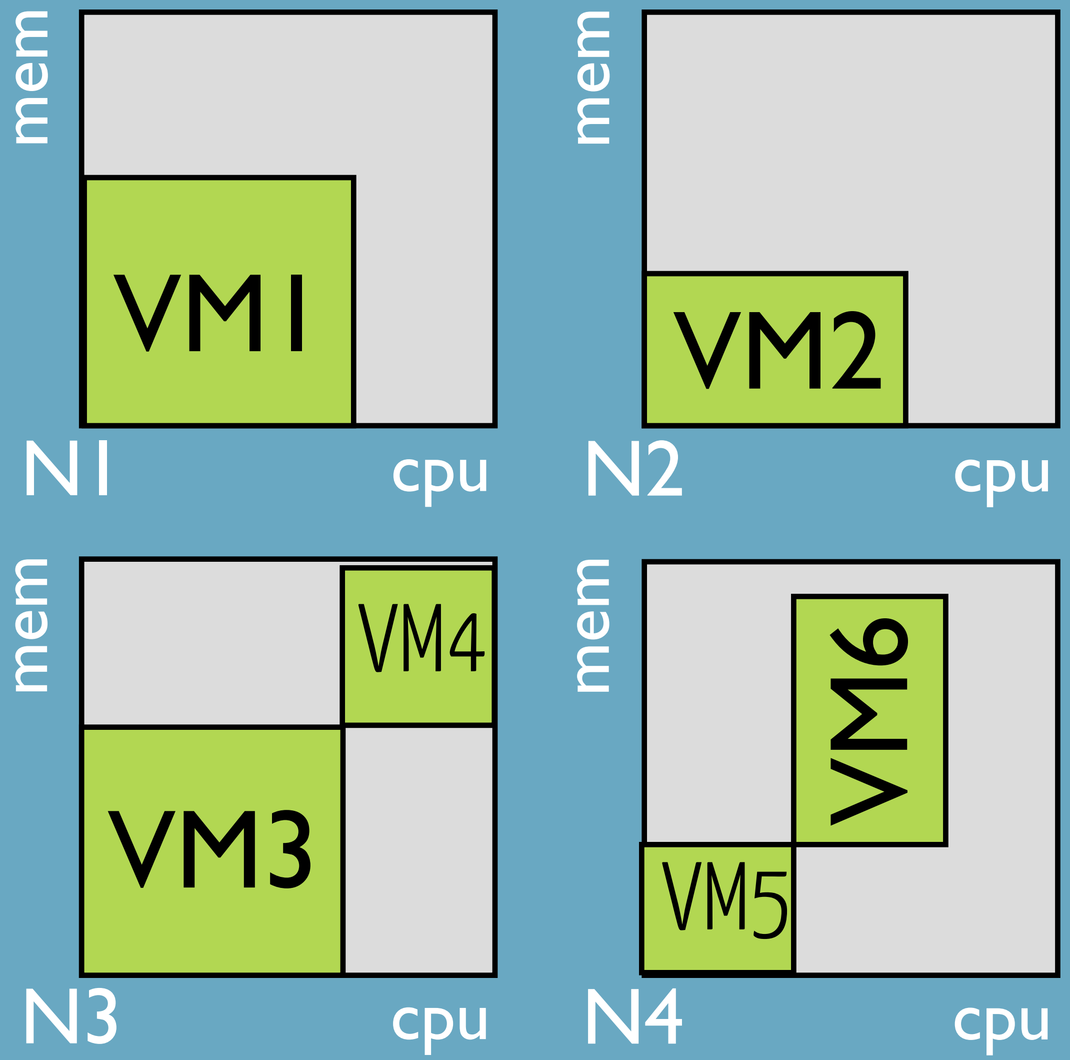
dynamic schedulers

dependency management



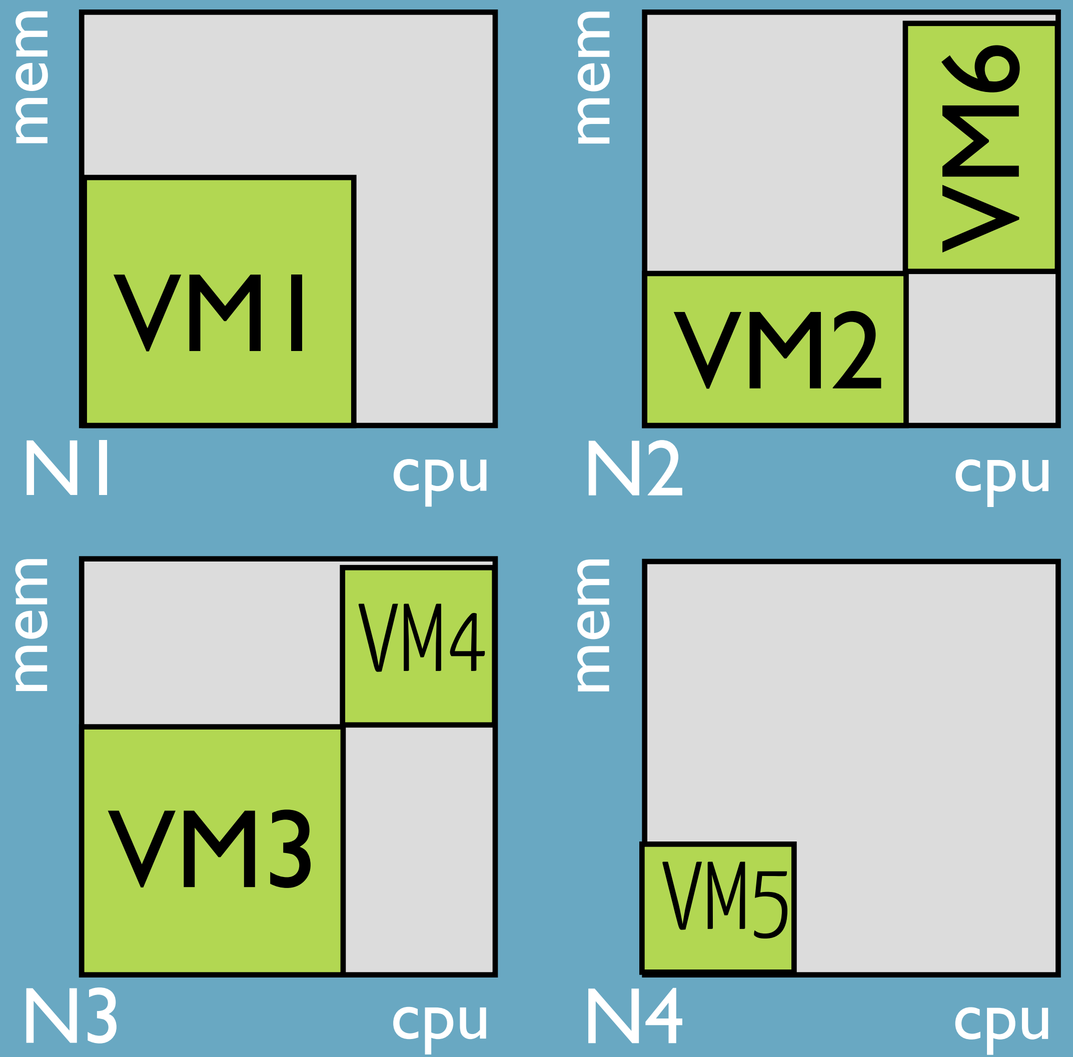
dynamic schedulers

dependency management



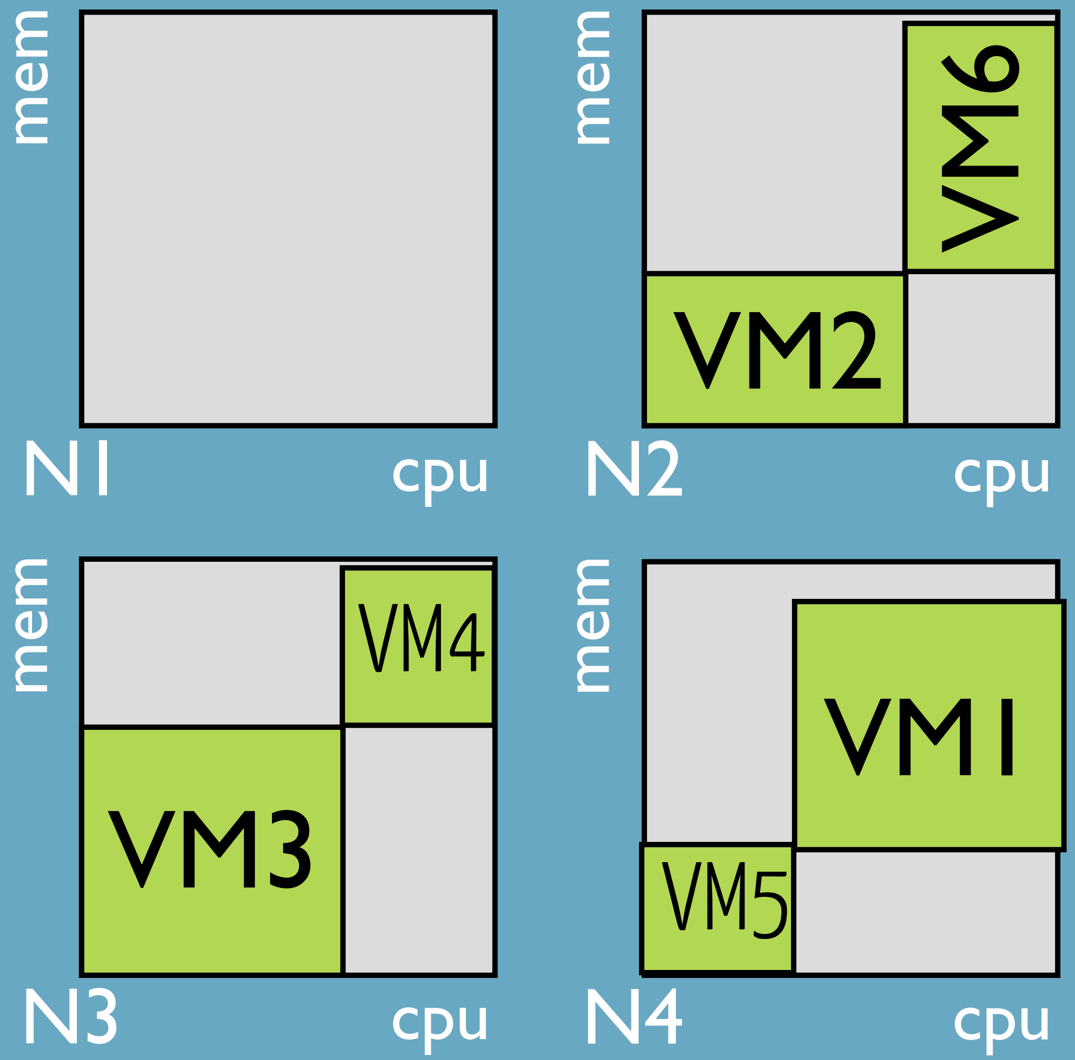
dynamic schedulers

dependency management



dynamic schedulers

dependency management

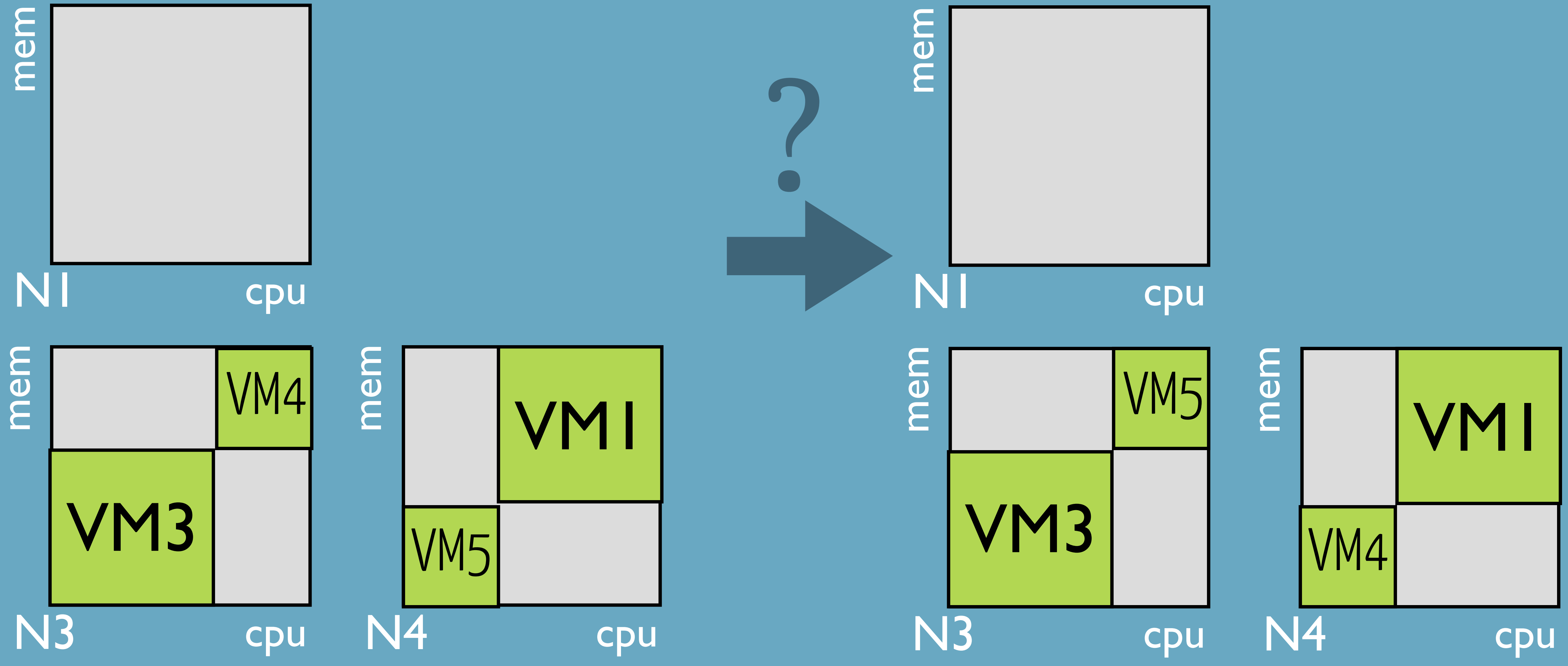


dynamic schedulers

cyclic dependencies

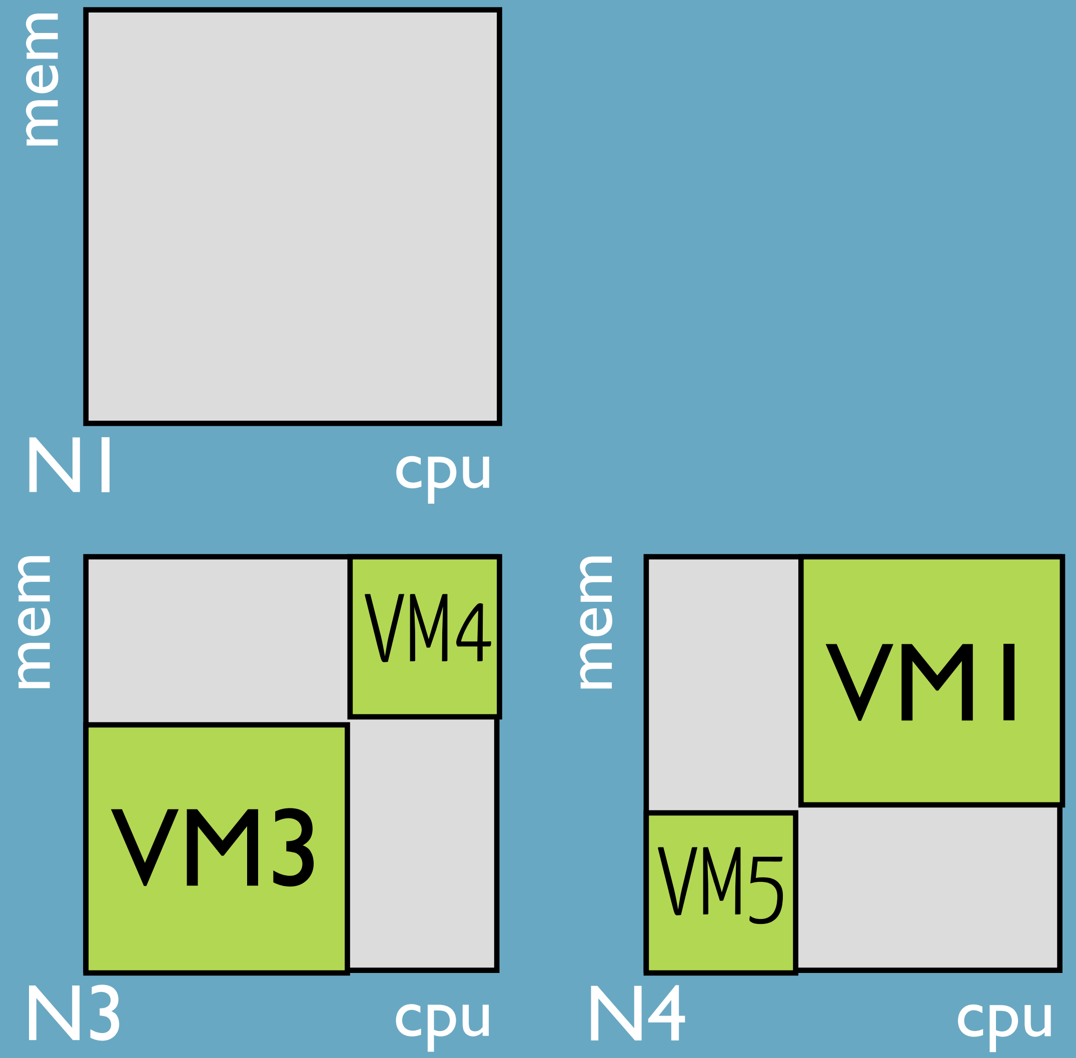
anti-affinity(VM3,VM4)
min(#onlineNodes)

anti-affinity(VM3,VM4)
min(#onlineNodes)



dynamic schedulers

cyclic dependencies

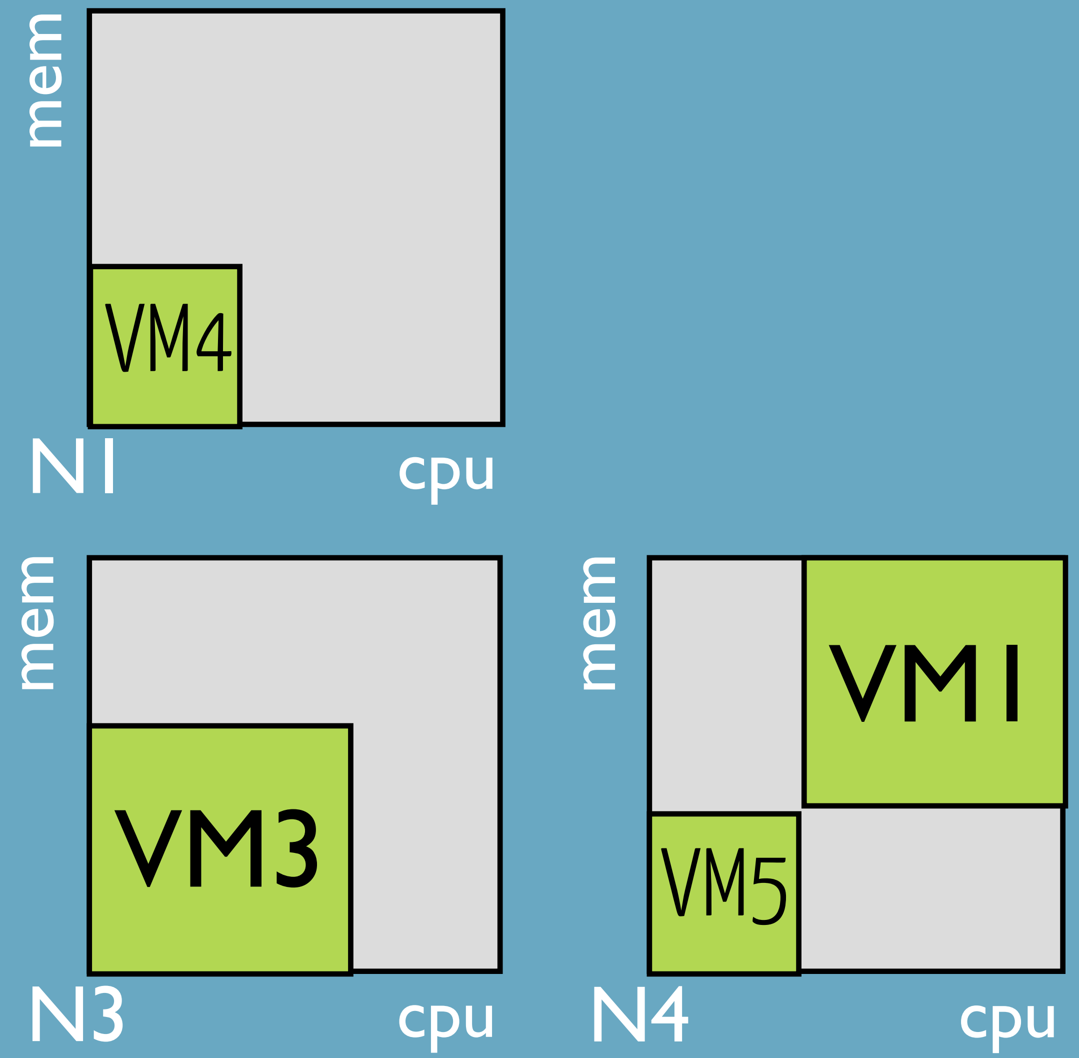


a pivot to break the cycle

fix or prevent the situation ?

dynamic schedulers

cyclic dependencies

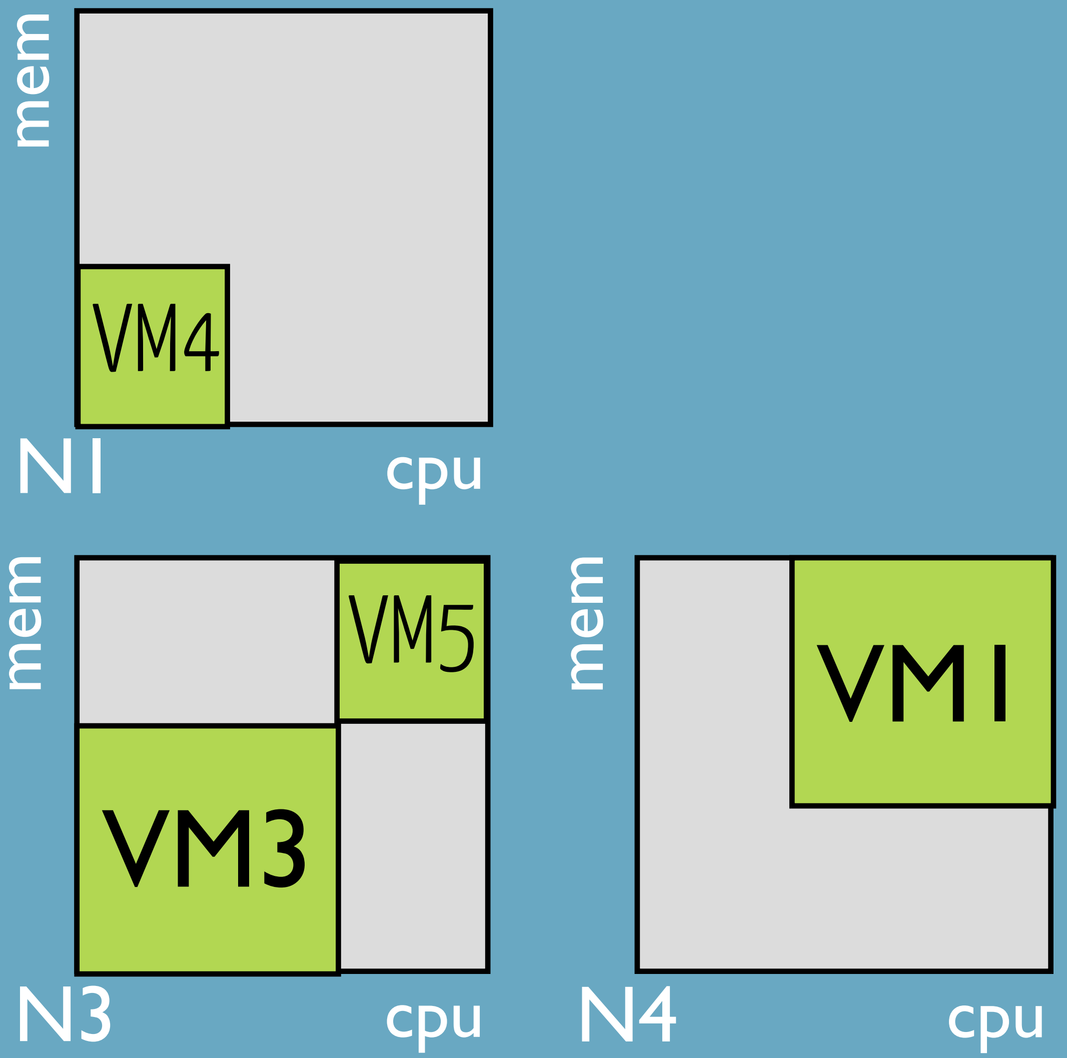


a pivot to break the cycle

fix or prevent the situation ?

dynamic schedulers

cyclic dependencies

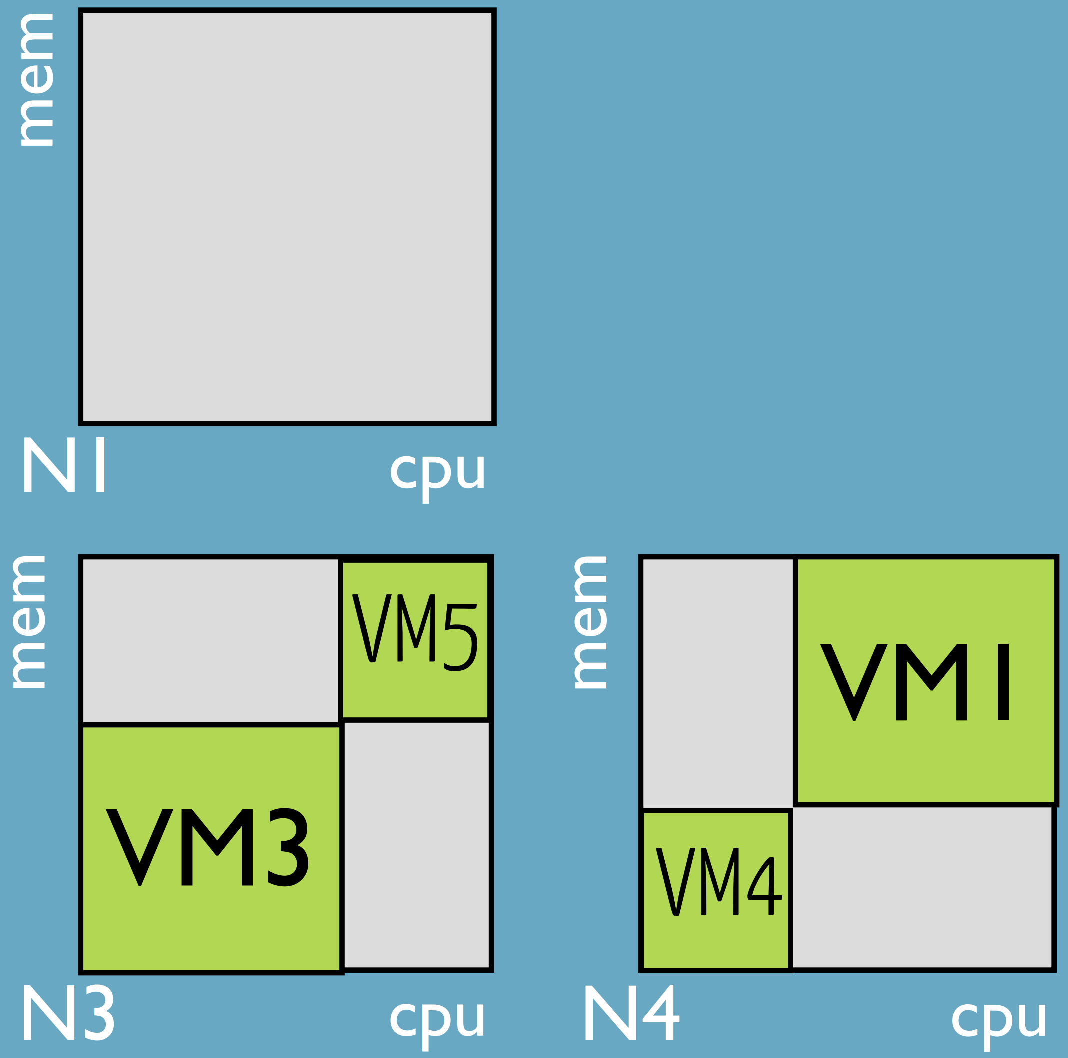


a pivot to break the cycle

fix or prevent the situation ?

dynamic schedulers

cyclic dependencies

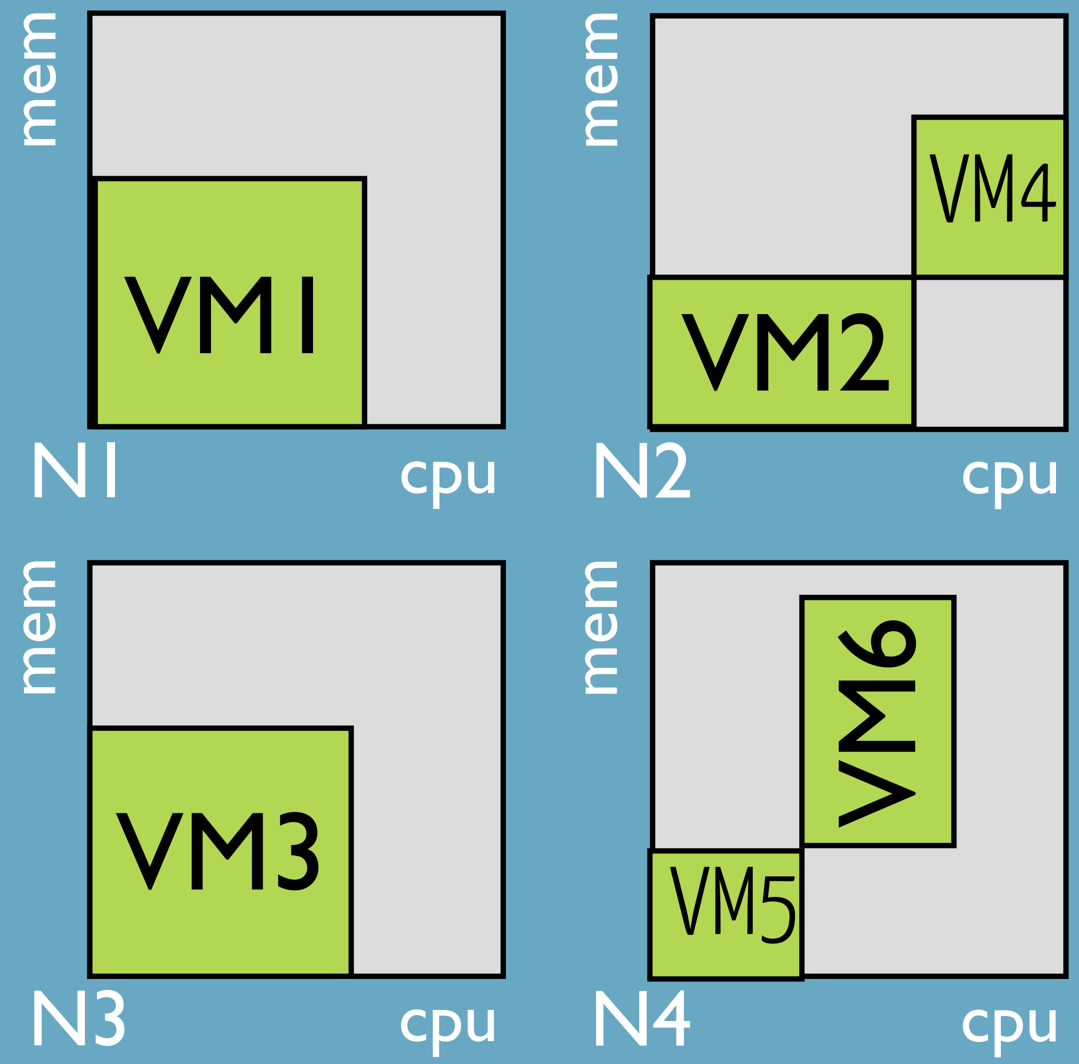


a pivot to break the cycle

fix or prevent the situation ?

dynamic schedulers

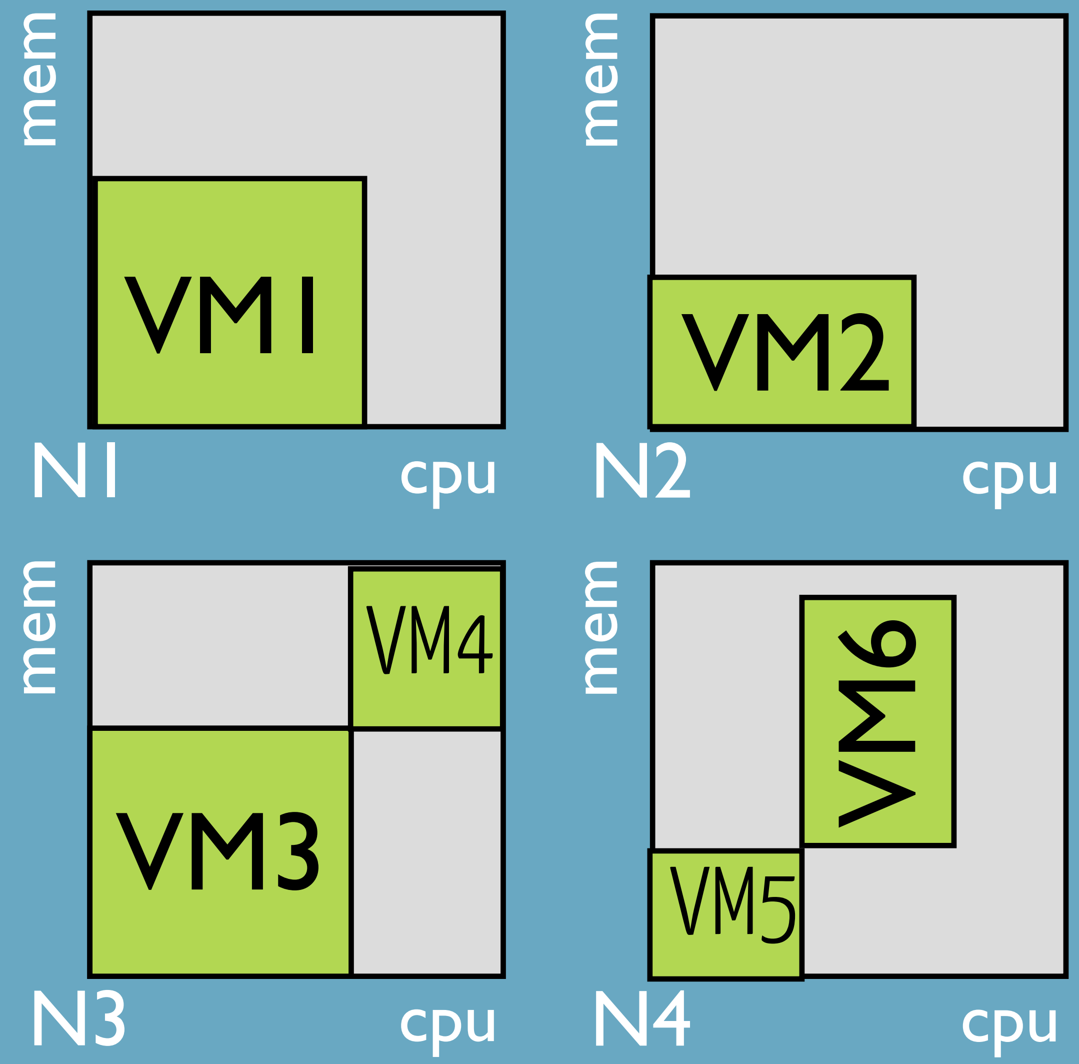
quality at a price



min(#onlineNodes) = 3

dynamic schedulers

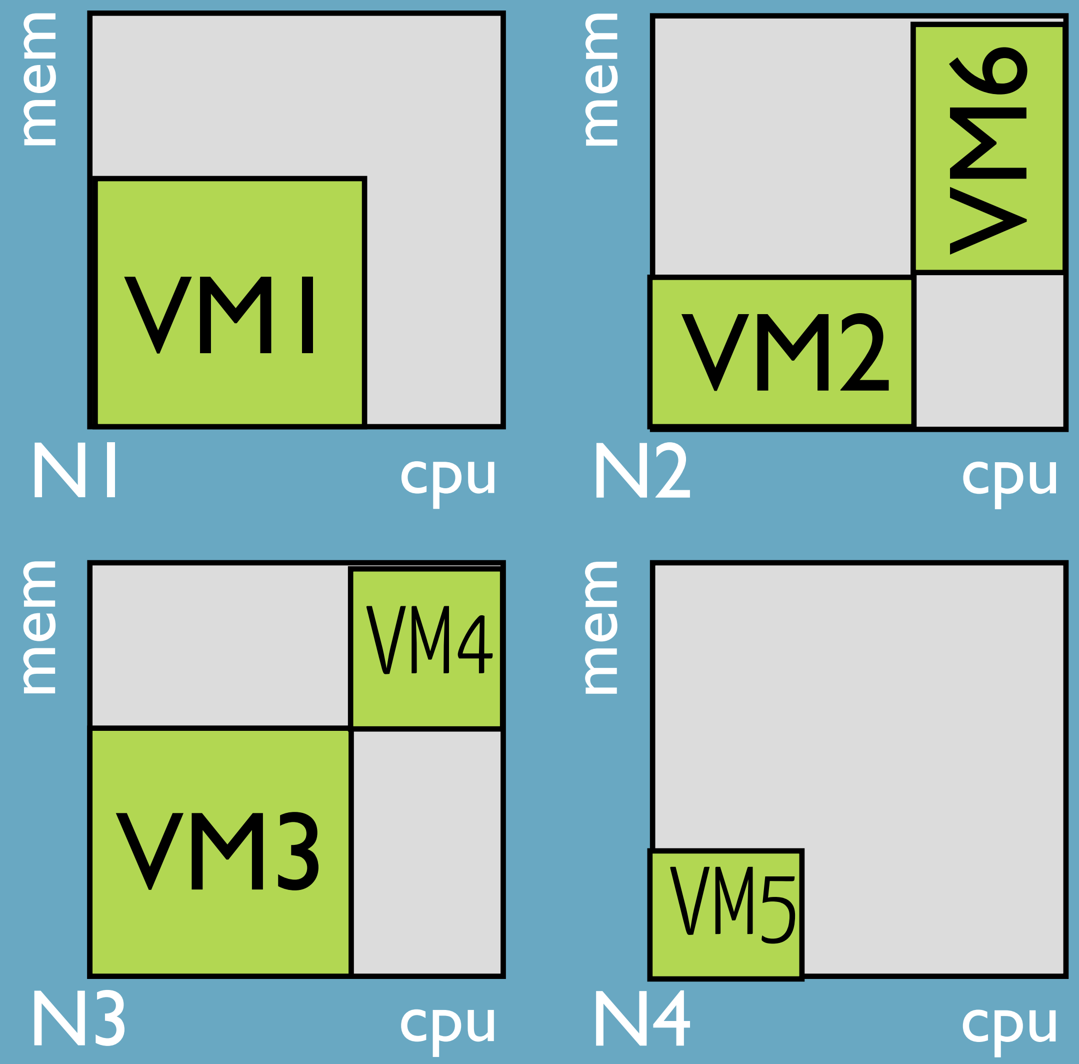
quality at a price



$min(\#onlineNodes) = 3$

dynamic schedulers

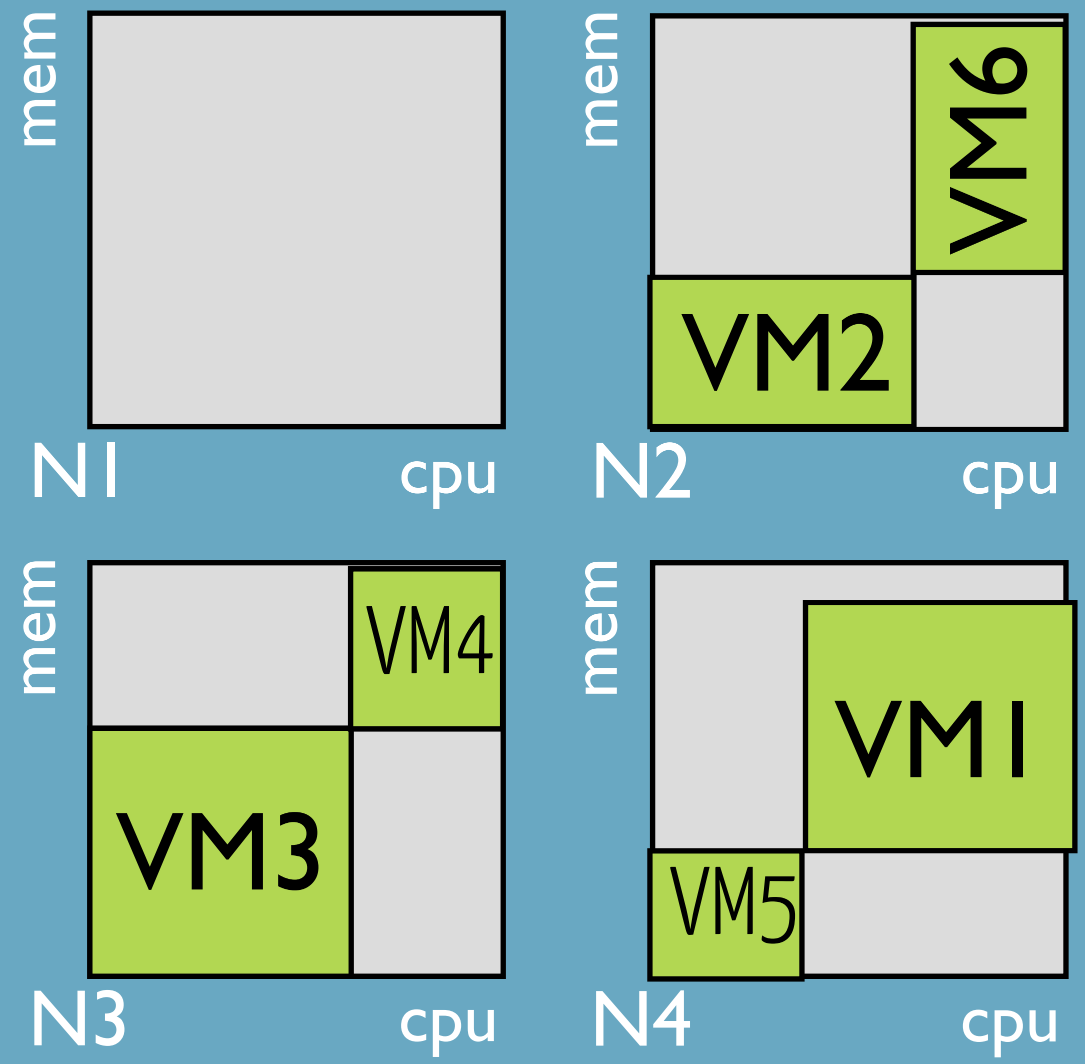
quality at a price



$min(\#onlineNodes) = 3$

dynamic schedulers

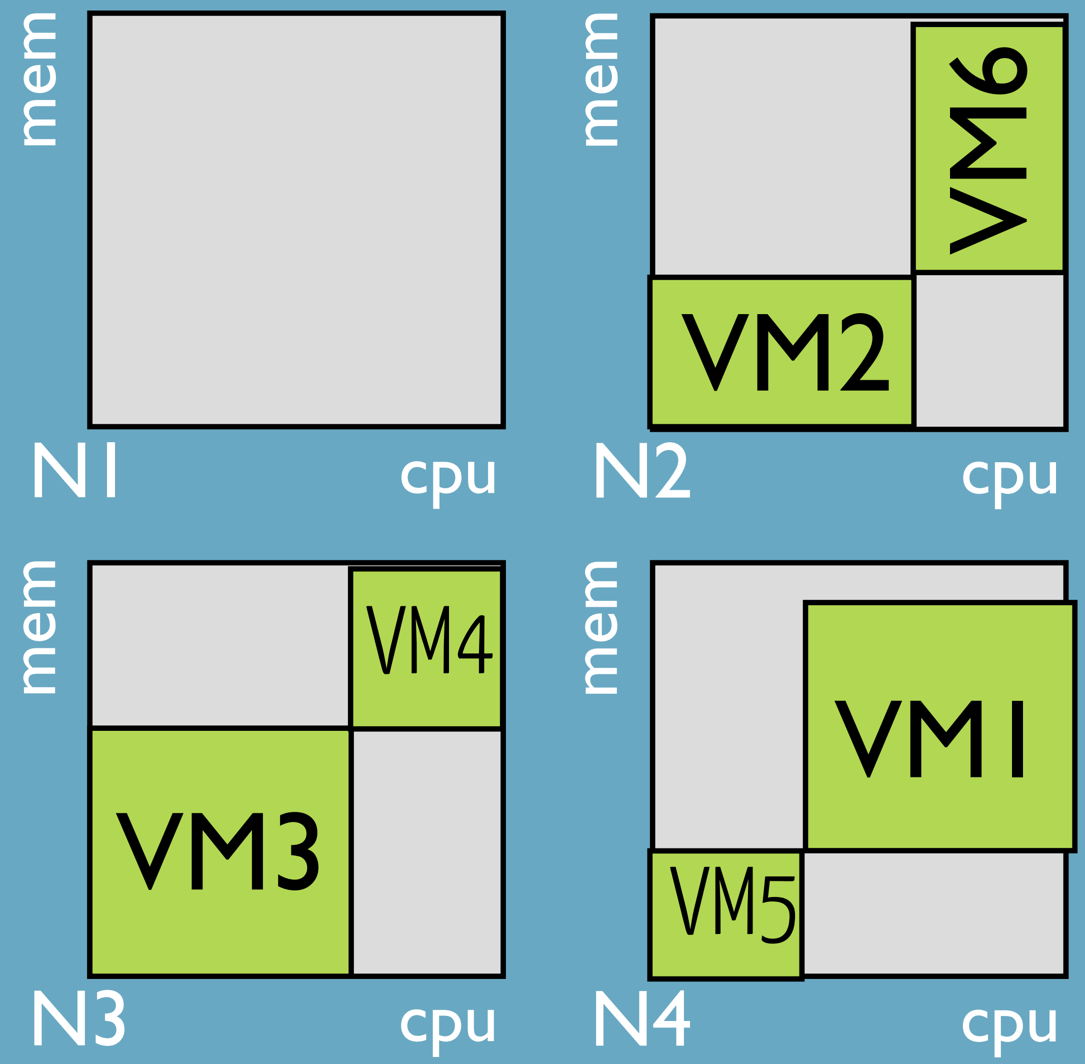
quality at a price



min(#onlineNodes) = 3

dynamic schedulers

quality at a price

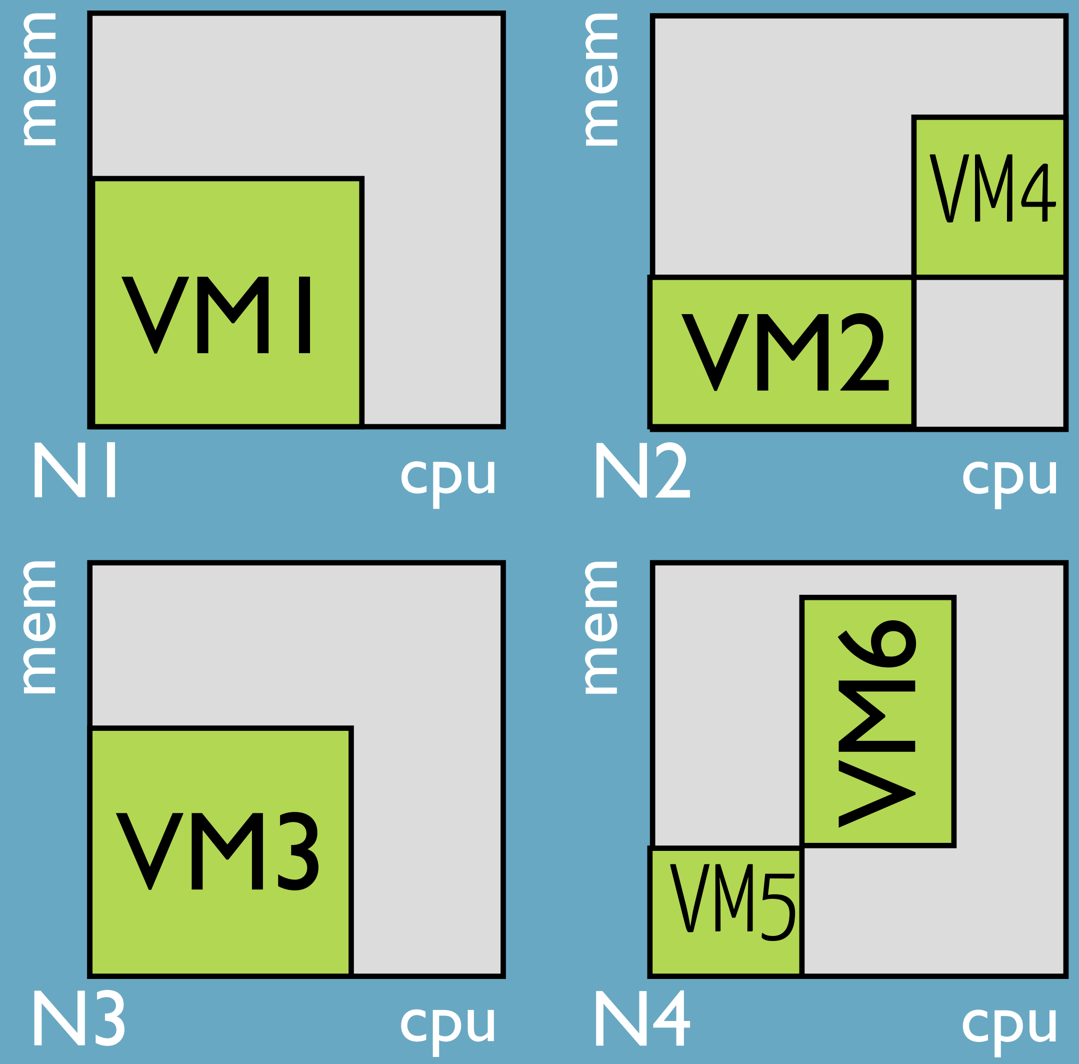


sol #1: 1m,1m,2m

$min(\#onlineNodes) = 3$

dynamic schedulers

quality at a price

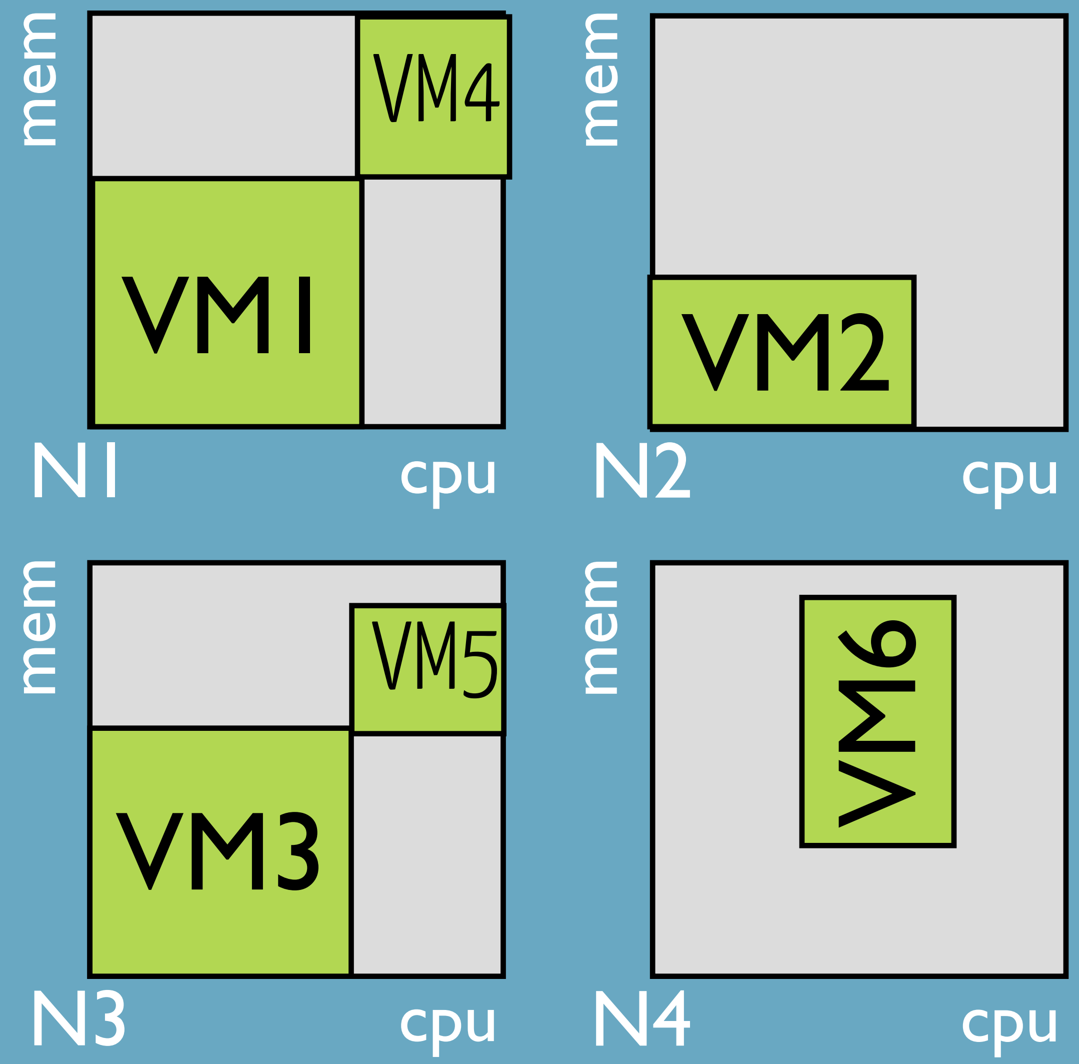


sol #1: 1m,1m,2m

$min(\#onlineNodes) = 3$

dynamic schedulers

quality at a price

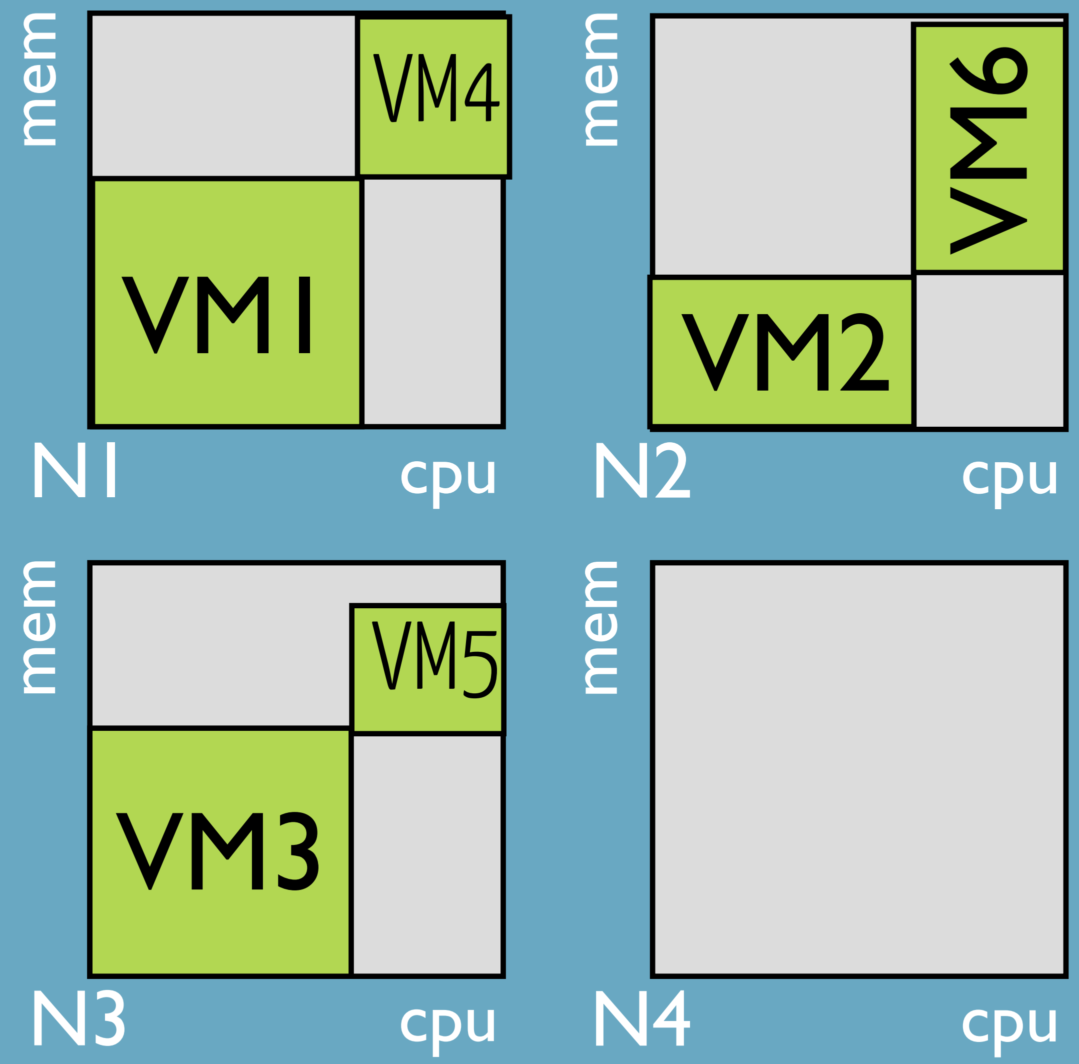


min(#onlineNodes) = 3

sol #1: 1m,1m,2m

dynamic schedulers

quality at a price

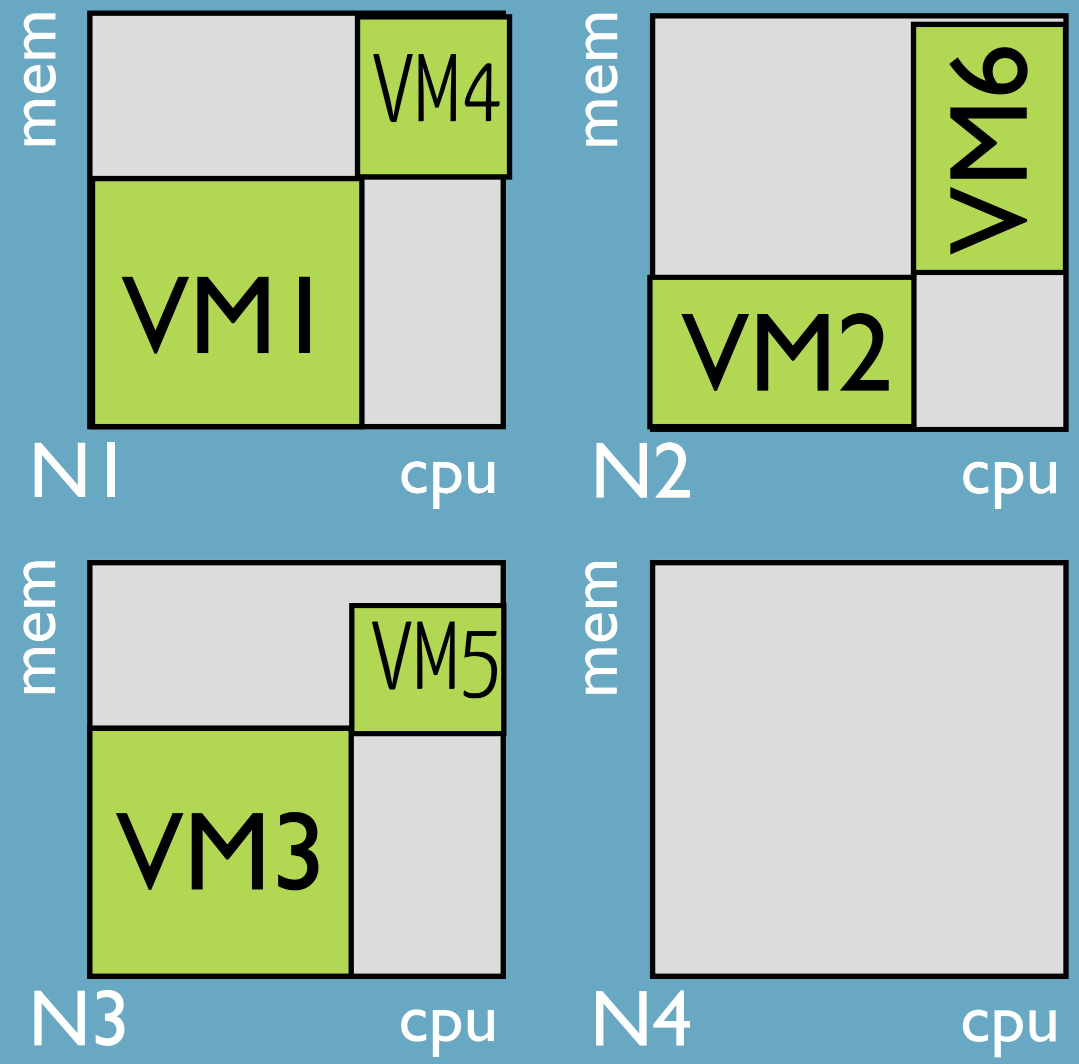


sol #1: 1m,1m,2m

min(#onlineNodes) = 3

dynamic schedulers

quality at a price



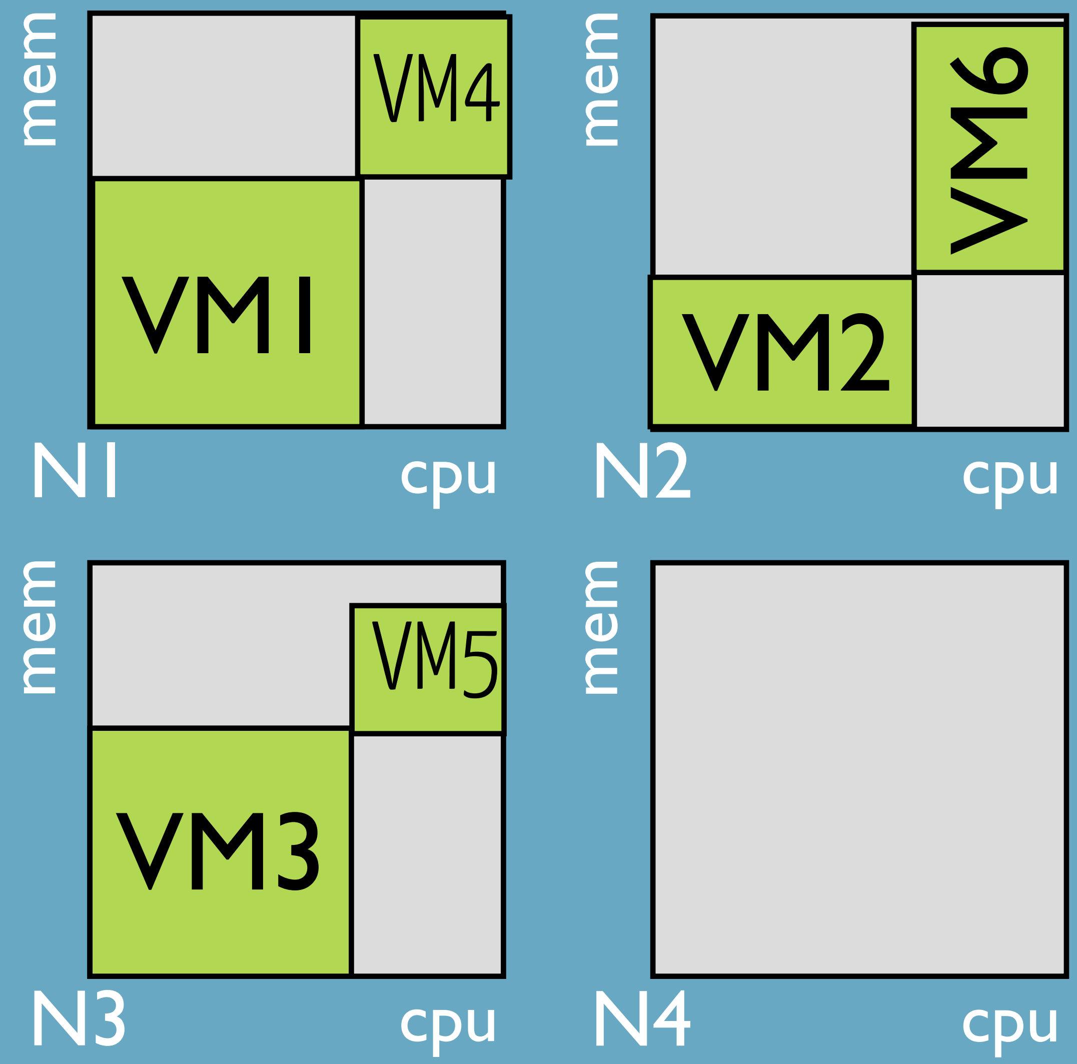
sol #1: 1m,1m,2m

sol #2: 1m,1m
1m

$min(\#onlineNodes) = 3$

dynamic schedulers

quality at a price



sol #1: 1m,1m,2m

sol #2: 1m,1m
1m

lower MTTR
(faster)

$min(\#onlineNodes) = 3$

dynamic schedulers

quality at a price

the objective should reflect reconfiguration costs

$min(MTTR), min(\#migrations), \ldots$

dynamic schedulers

pros

continuous optimisation
through reconfiguration

dynamic schedulers

cons

harder to model
harder to scale
technically expensive
costly reconfiguration

dynamic scheduling for the win?

in theory yes but that’s theory

benefits depends on

- the workload
- the objective/ SLAs
- the infrastructure

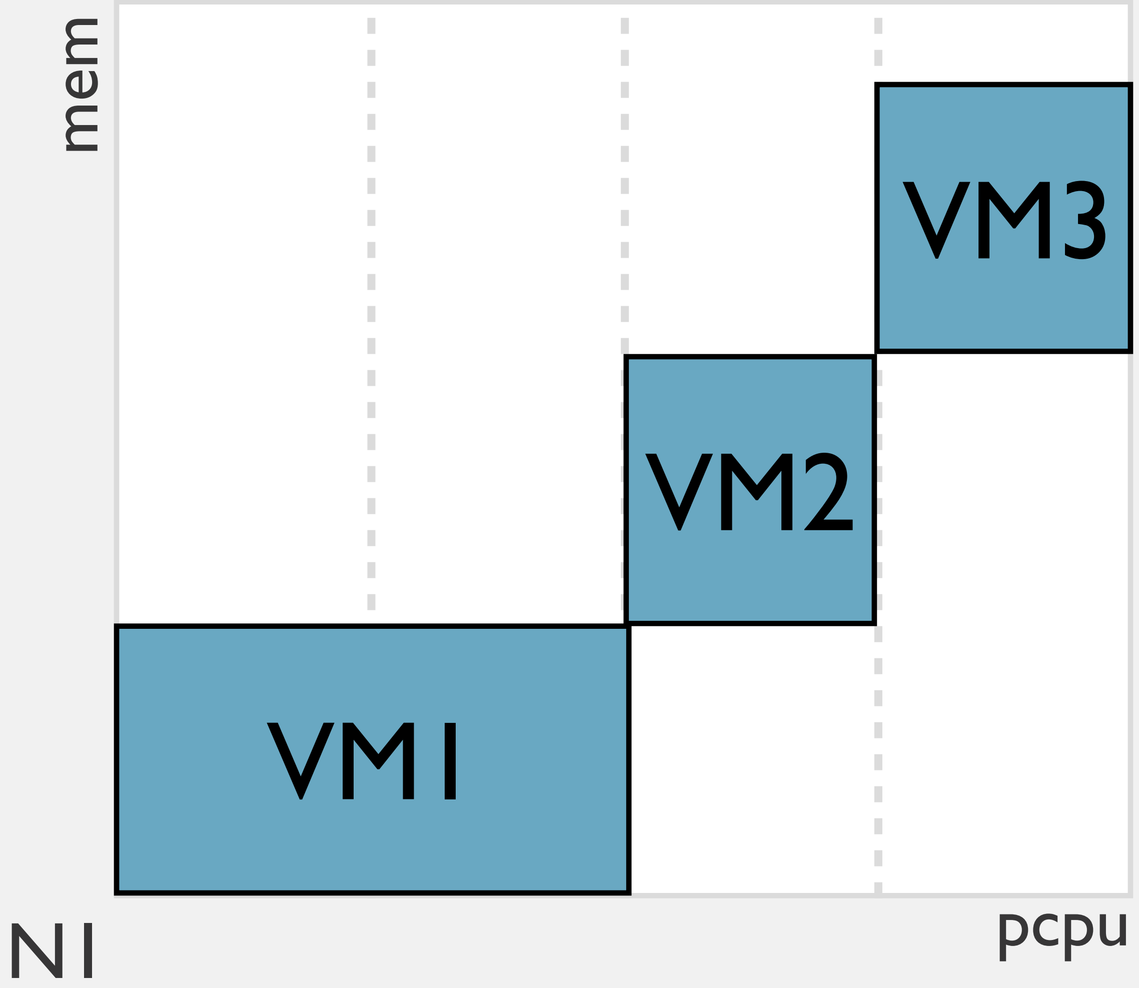
dynamic scheduling is rare
in public or large clouds

scheduling models

static
dynamic schedulers

static
dynamic resource allocation

static resource allocation



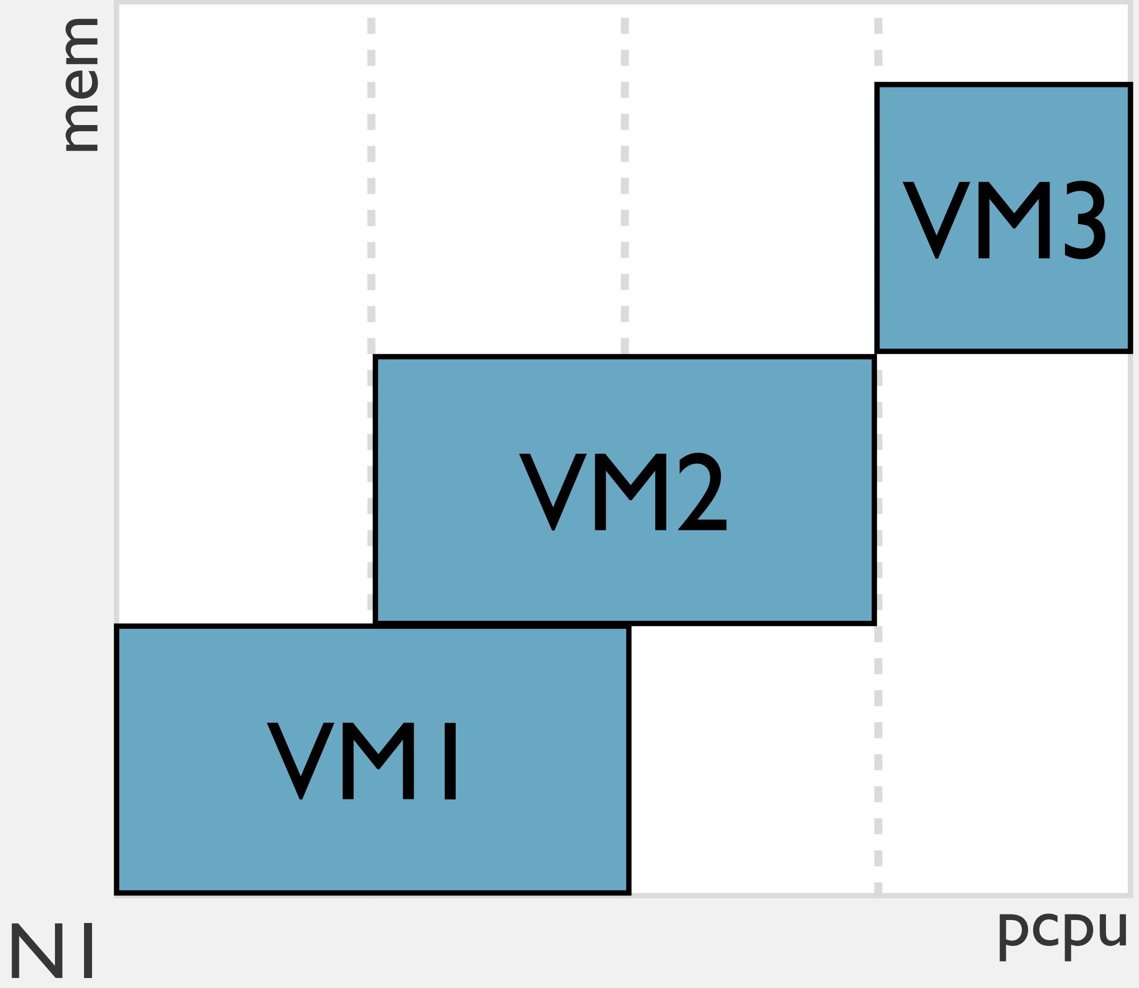
cpu, ram, i/o, bandwidth
allocated once for all

allocation != utilisation

no sharing
conservative allocation

static resource allocation

sharing (overbooking)

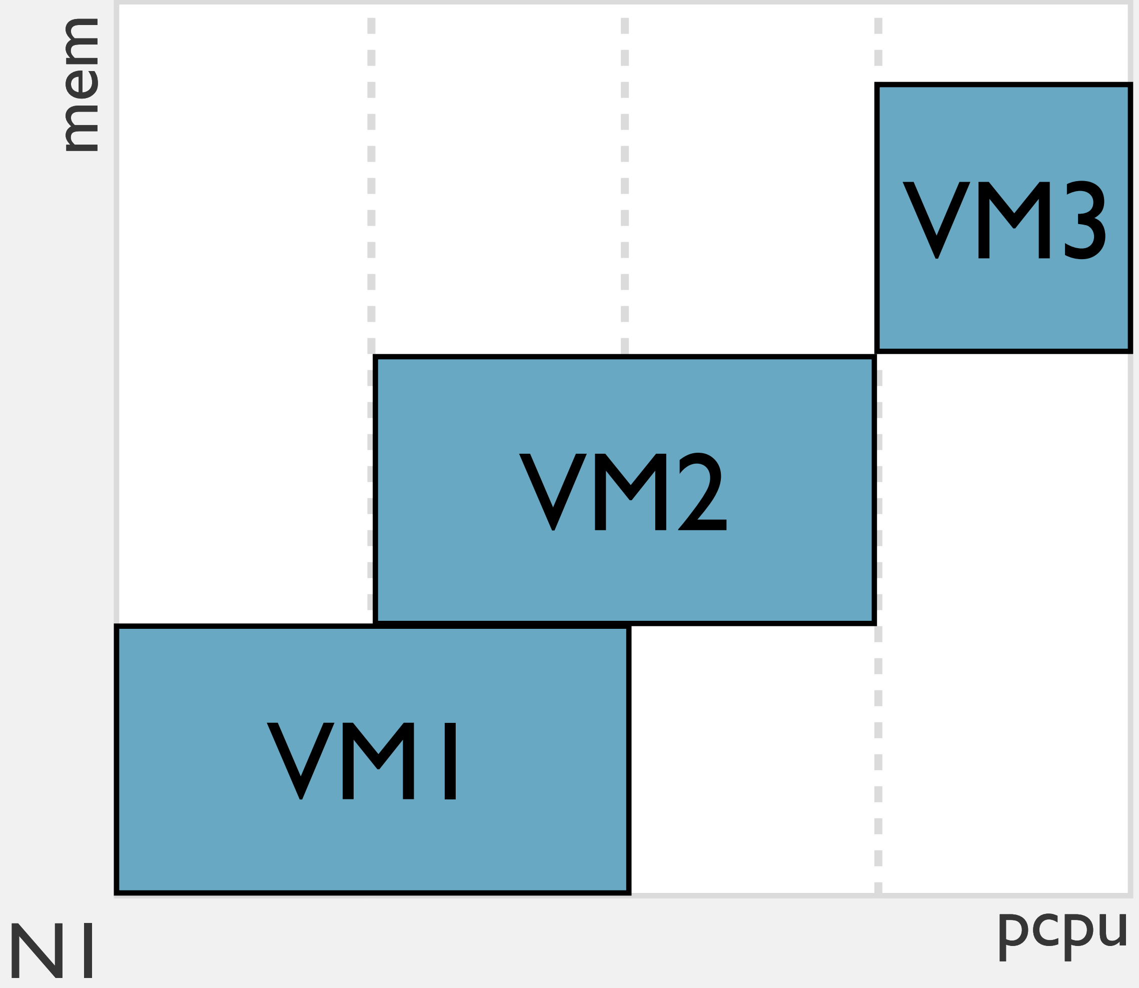


performance loss
with concurrent accesses

acceptable if stated
in the SLA

dynamic resource allocation

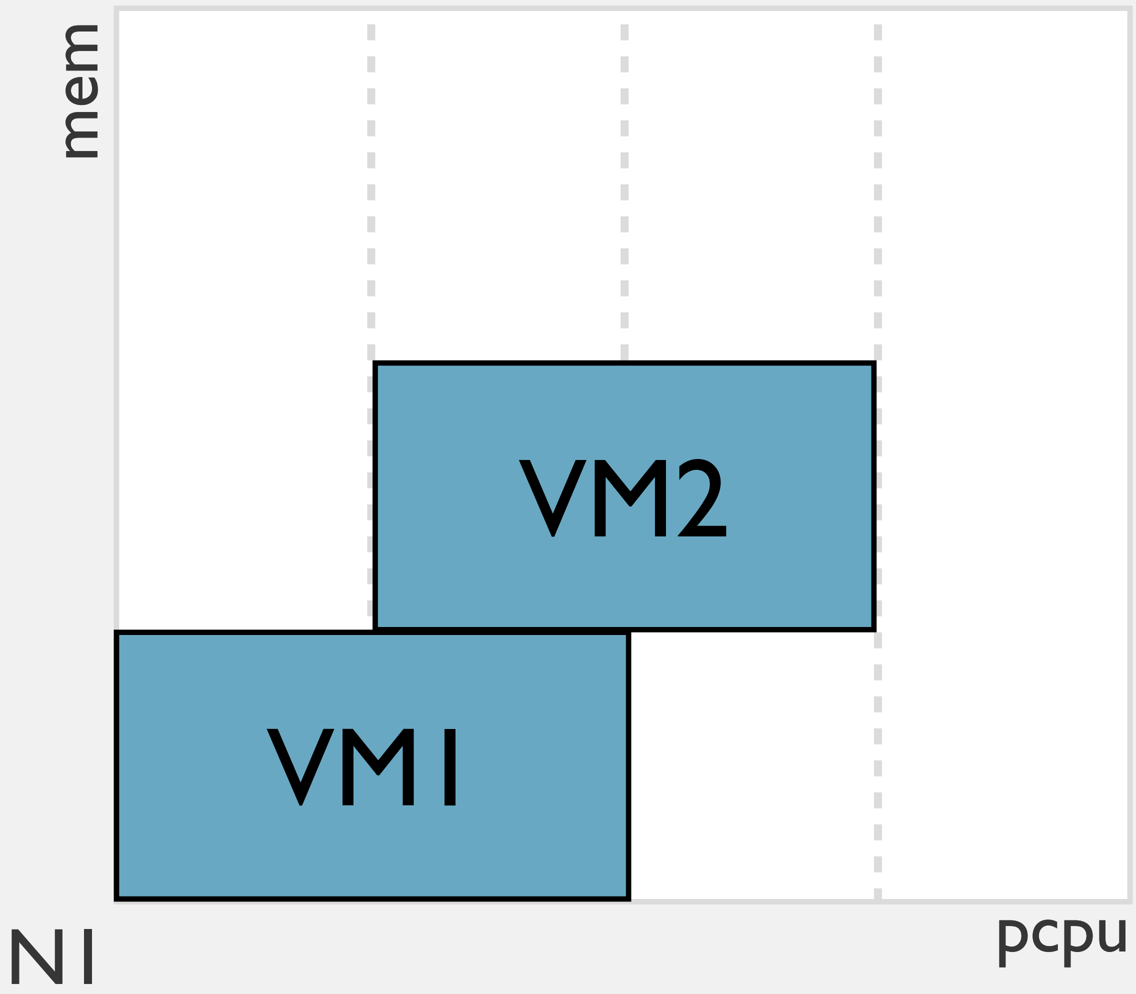
to fix violations



cpu, ram, i/o, bandwidth
allocated can be revised

dynamic resource allocation

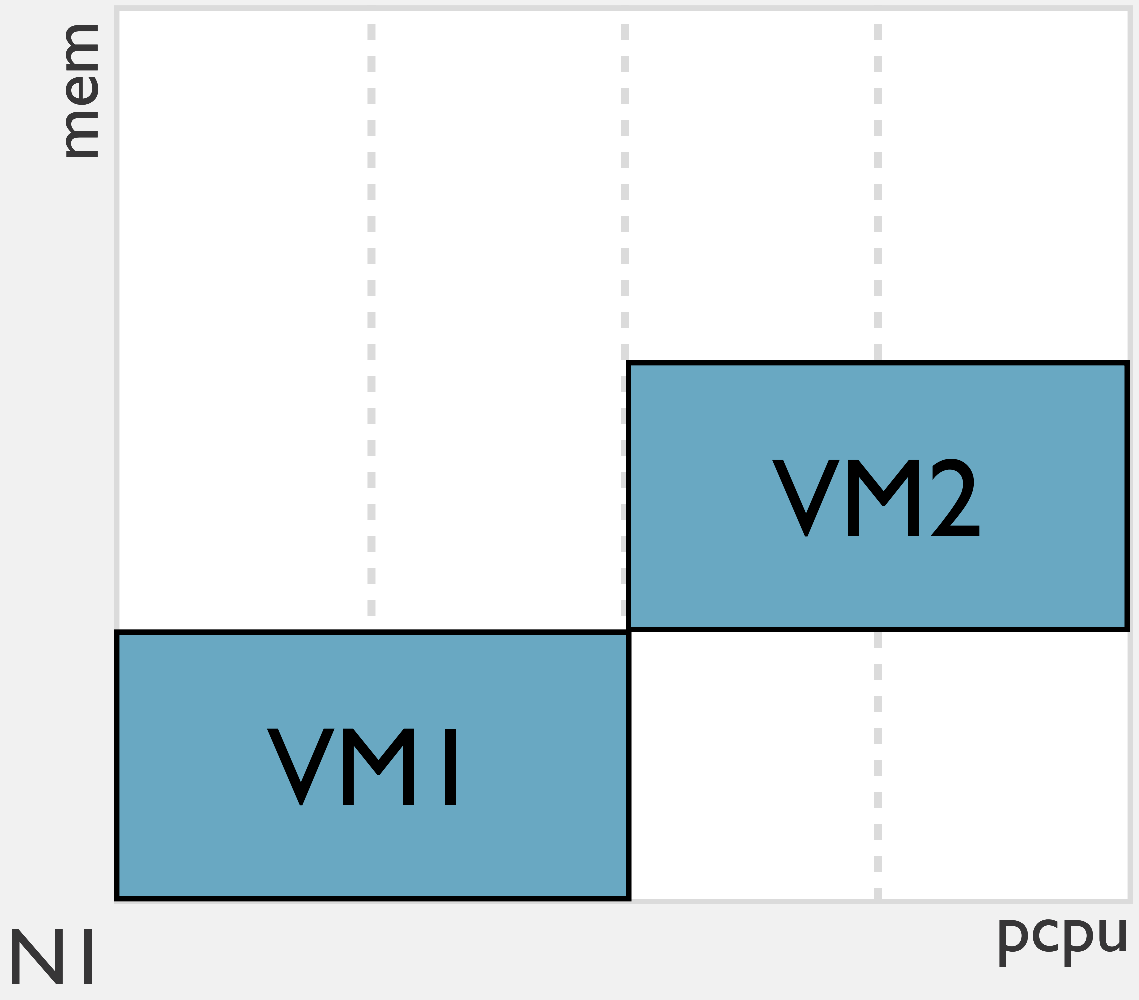
to fix violations



cpu, ram, i/o, bandwidth
allocated can be revised

dynamic resource allocation

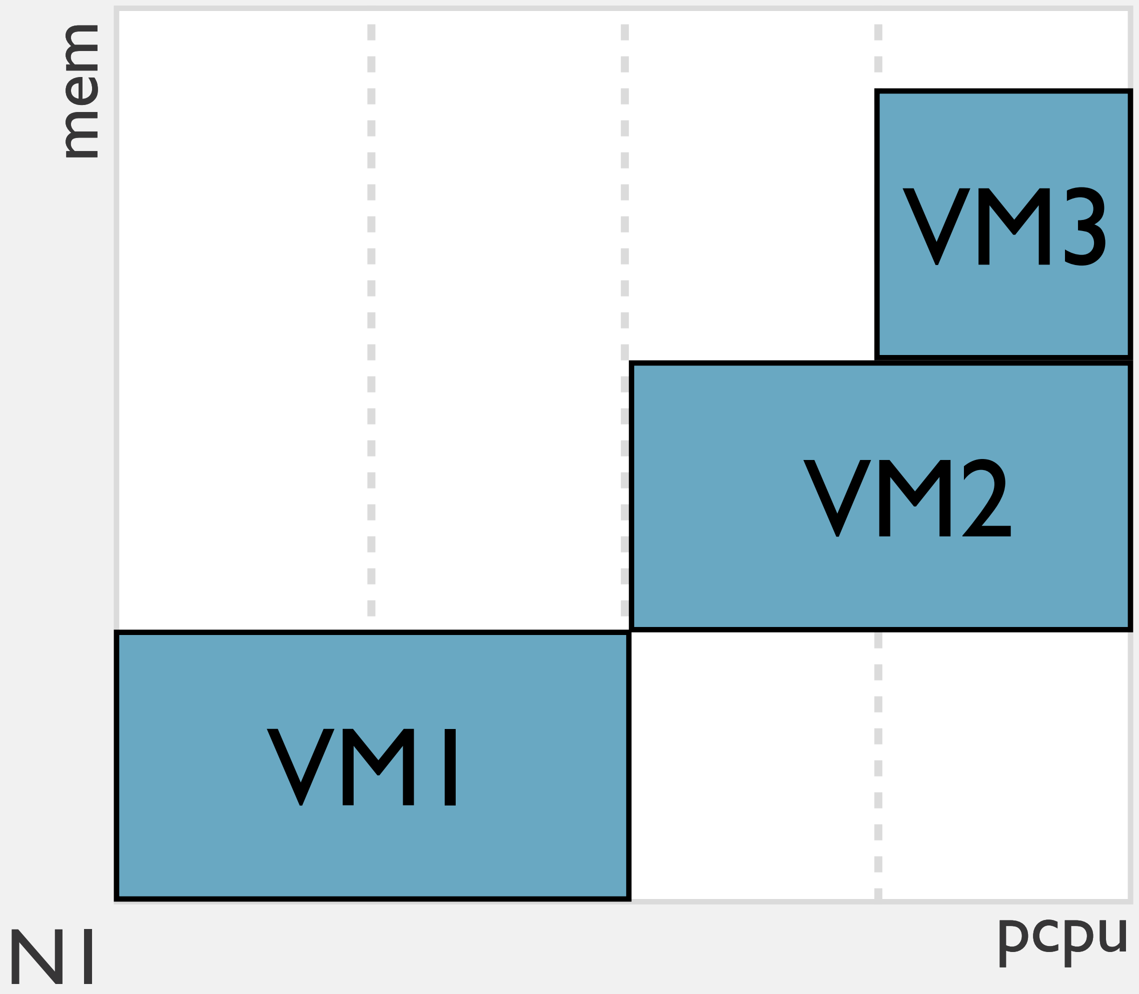
to fix violations



cpu, ram, i/o, bandwidth
allocated can be revised

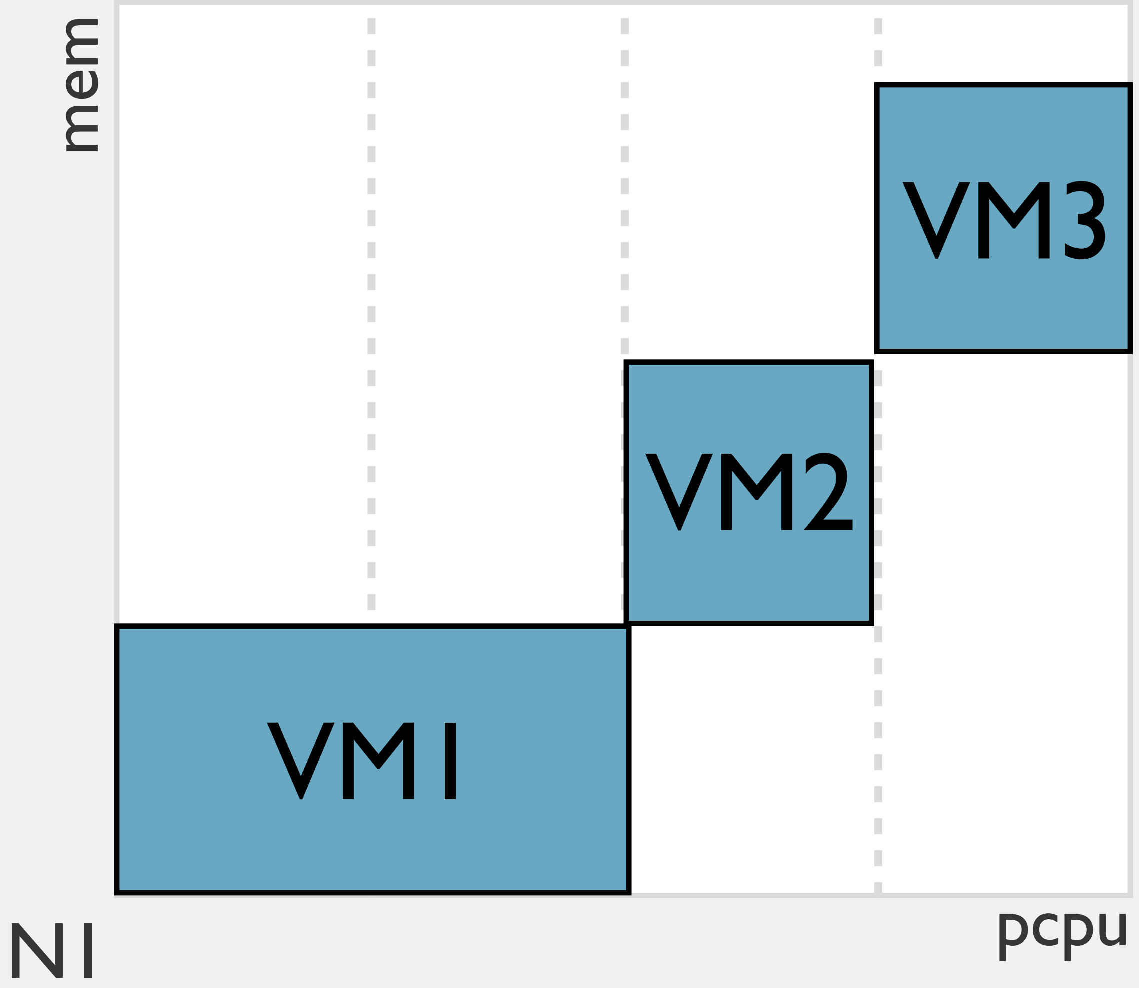
dynamic resource allocation

to support vertical elasticity



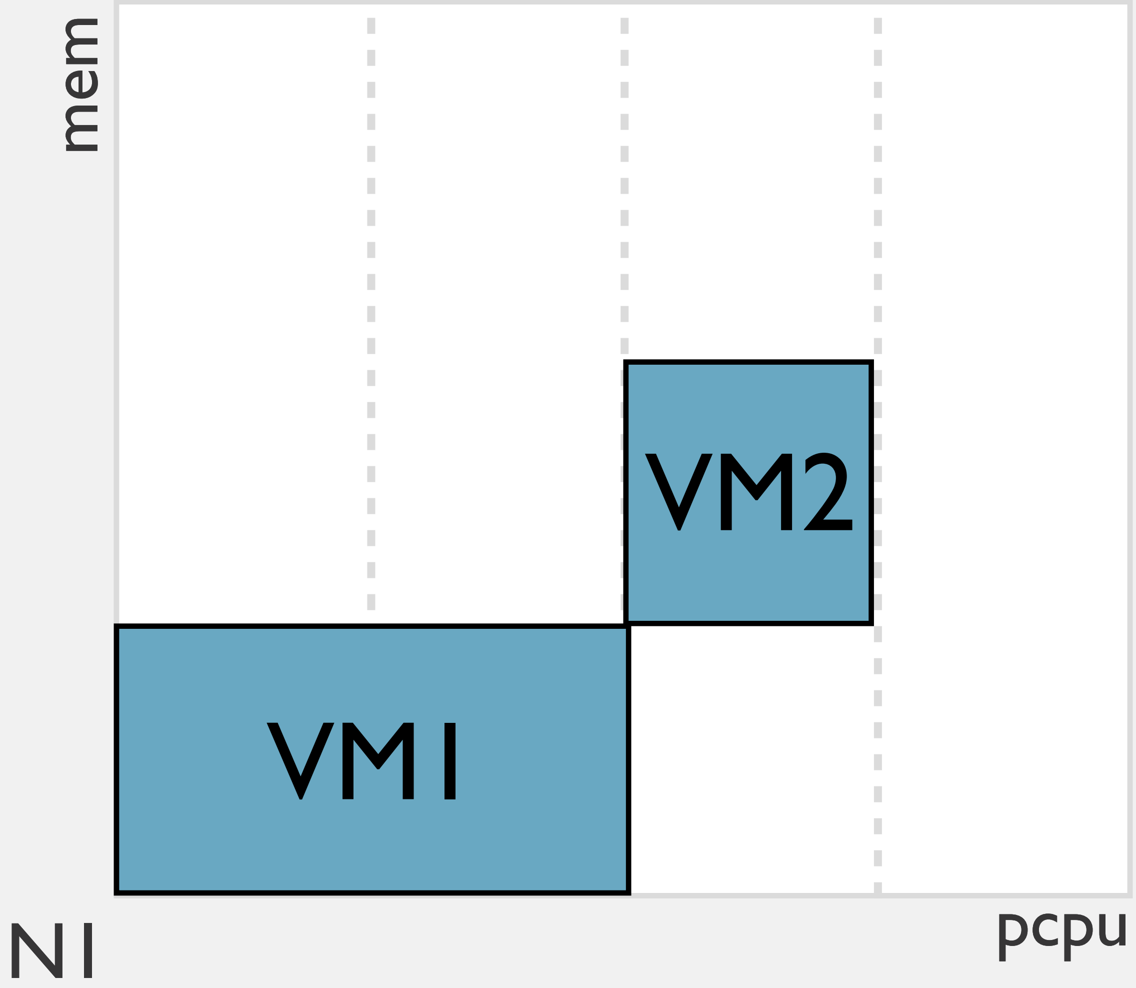
dynamic resource allocation

to support vertical elasticity



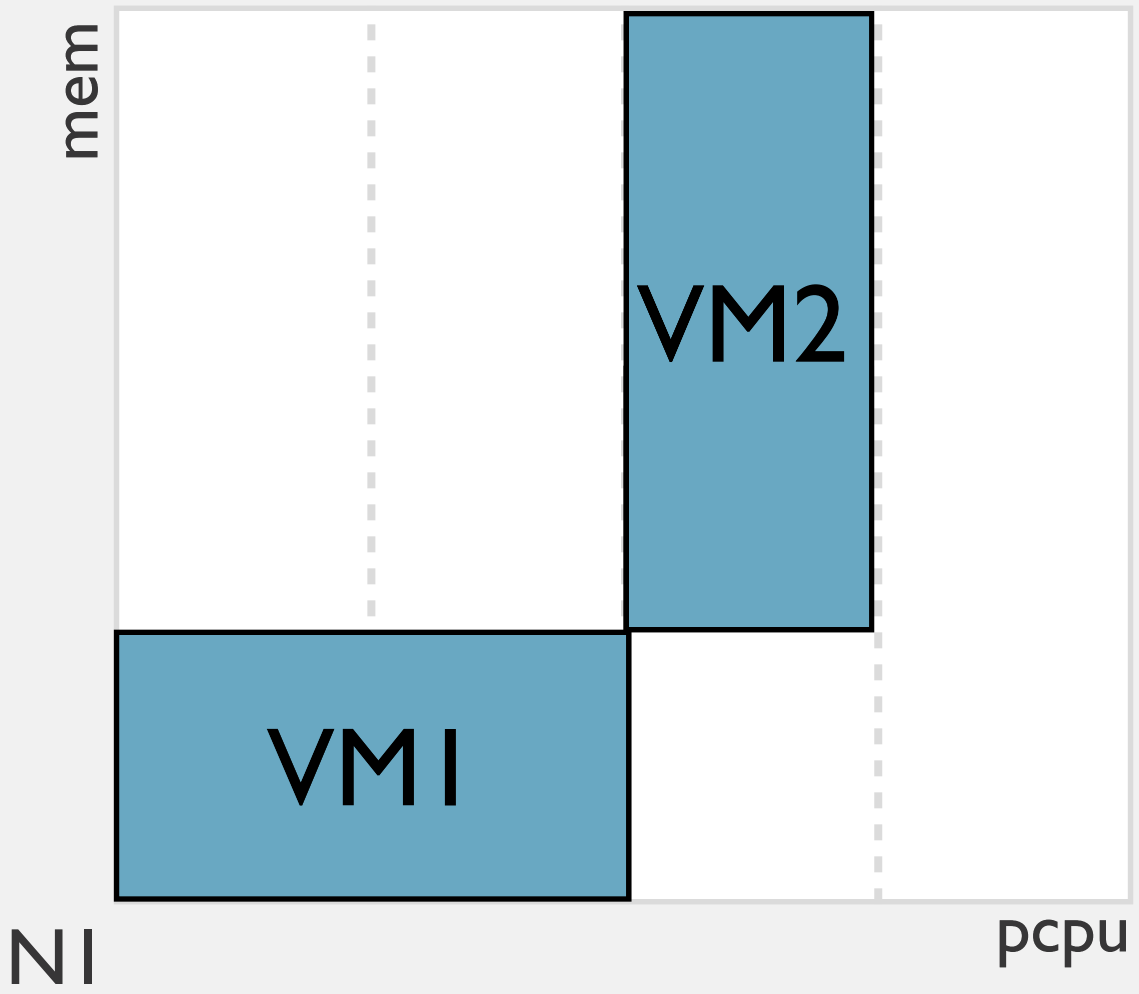
dynamic resource allocation

to support vertical elasticity



dynamic resource allocation

to support vertical elasticity



dynamic resource allocation for the win ?

common on CPU + overbooking
(inc. hosting capacity)

exceptional for memory
(huge performance loss)

benefits depends on
the workload
the objective/ SLAs
the infrastructure

RECAP

The VM scheduler makes cloud benefits real

no holy grail

think about what is costly

static scheduling for a peaceful life

dynamic scheduling to cease the day

with great power comes great responsibility

3 Coding a VM scheduler

SAFETY GLASSES

REQUIRED

VM scheduling is hard

static or dynamic scheduler
allocation ?

does the workload/SLA/objective
requires migration ?

maximum duration to schedule?

VM scheduling is NP-Hard

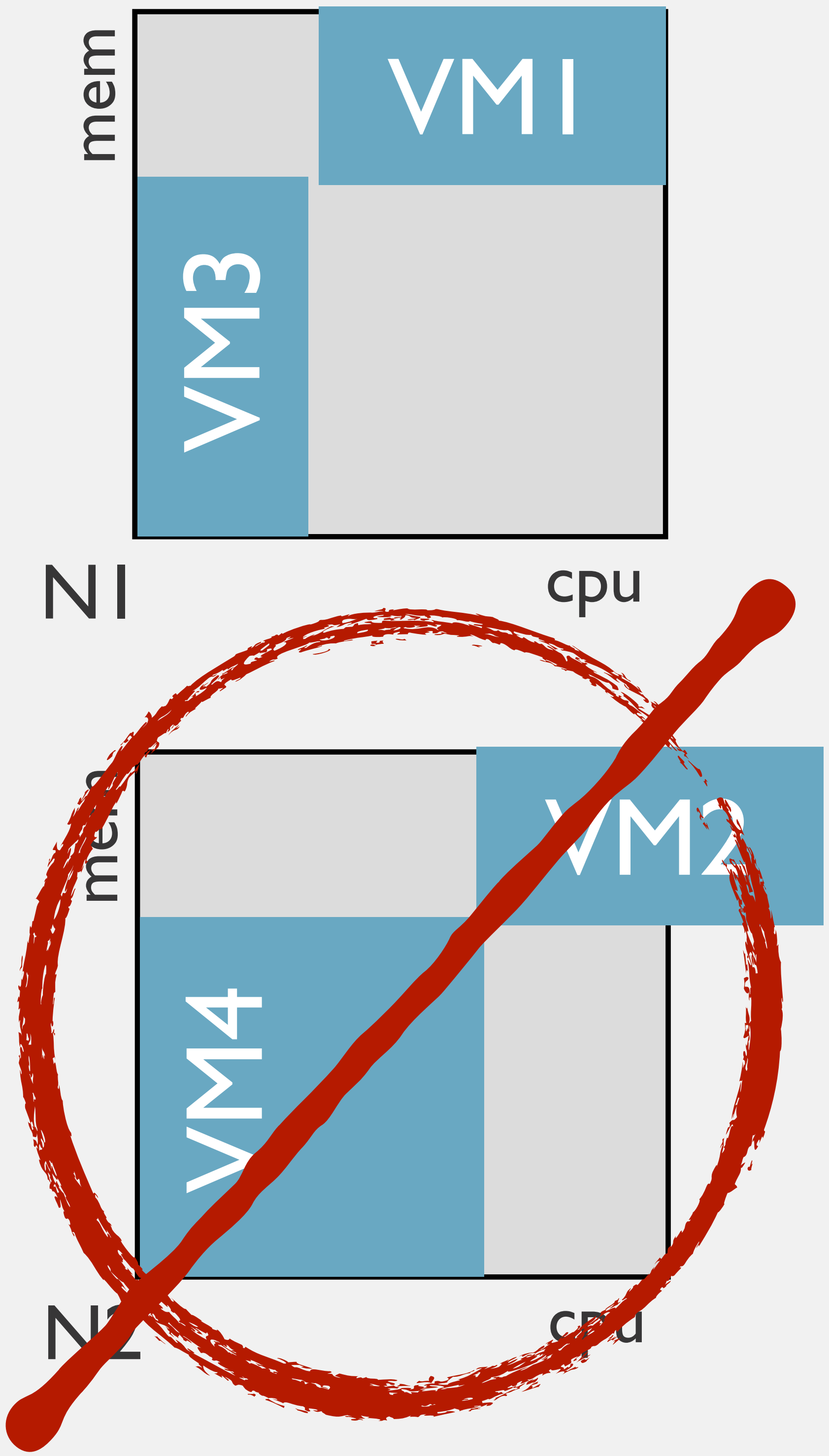
issues with large infrastructures
or hard problems

fast adhoc heuristics
despite corner cases

some use biased complete approaches
(linear programming, constraint programming)

like him

vector packing problem



items with a finite volume to
place inside finite bins

a generalisation of the bin
packing problem

the basic to model the infra.
1 dimension = 1 resource

?

Which resource can be modeled as a packing dimension

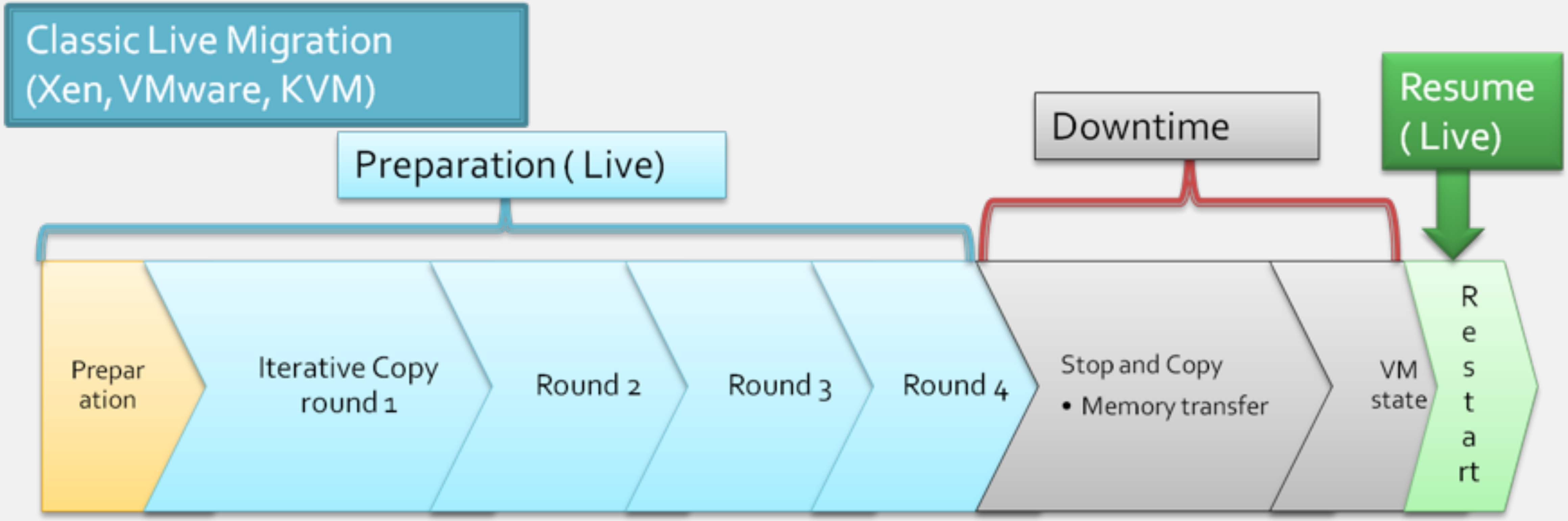
?

Which resource can be modeled as a packing dimension

CPU, memory, disk IO, licences cardinality, network boundaries, …

~~end to end network~~

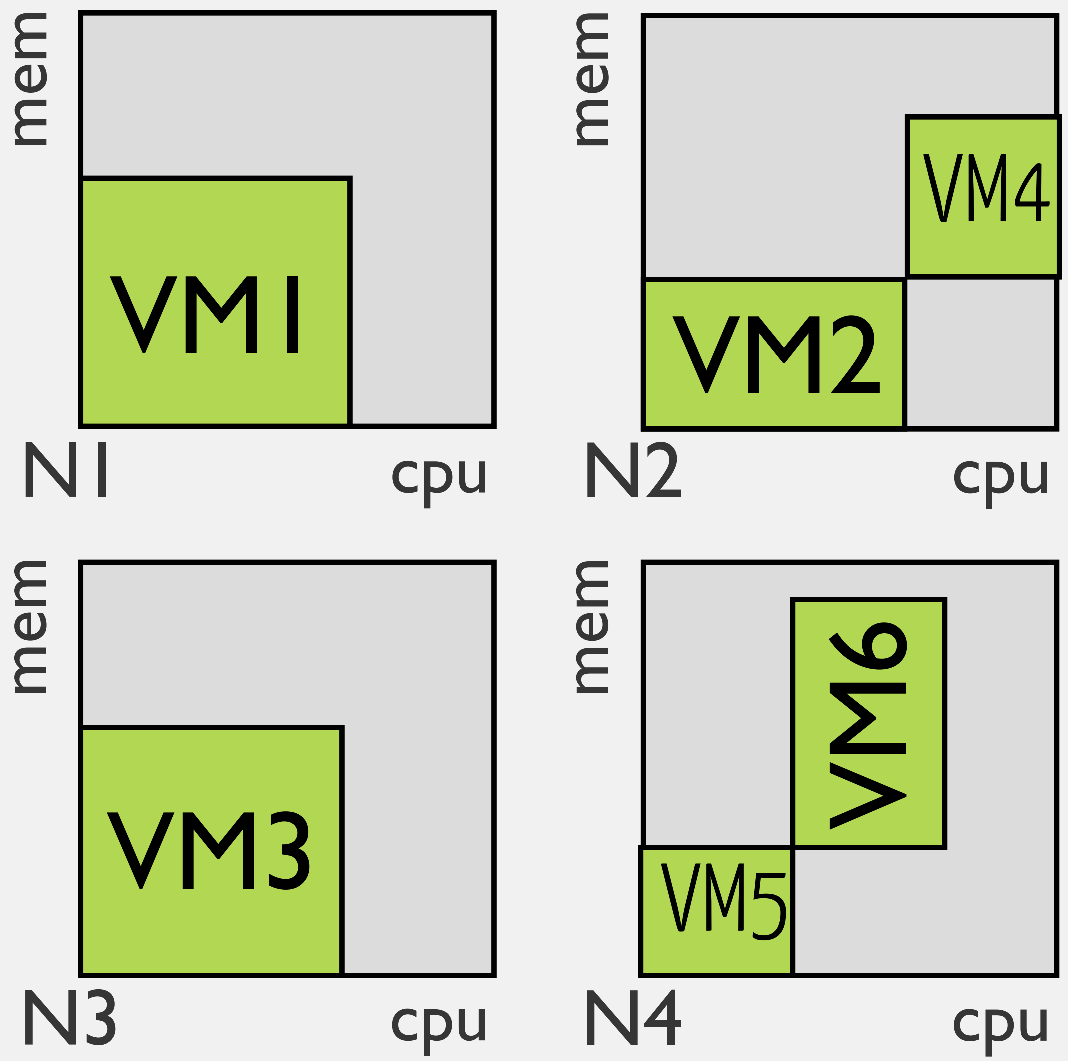
how to support migrations



temporary,
resources are used on the source and the destination nodes

how to support migrations

a simple way



n-phases vector packing

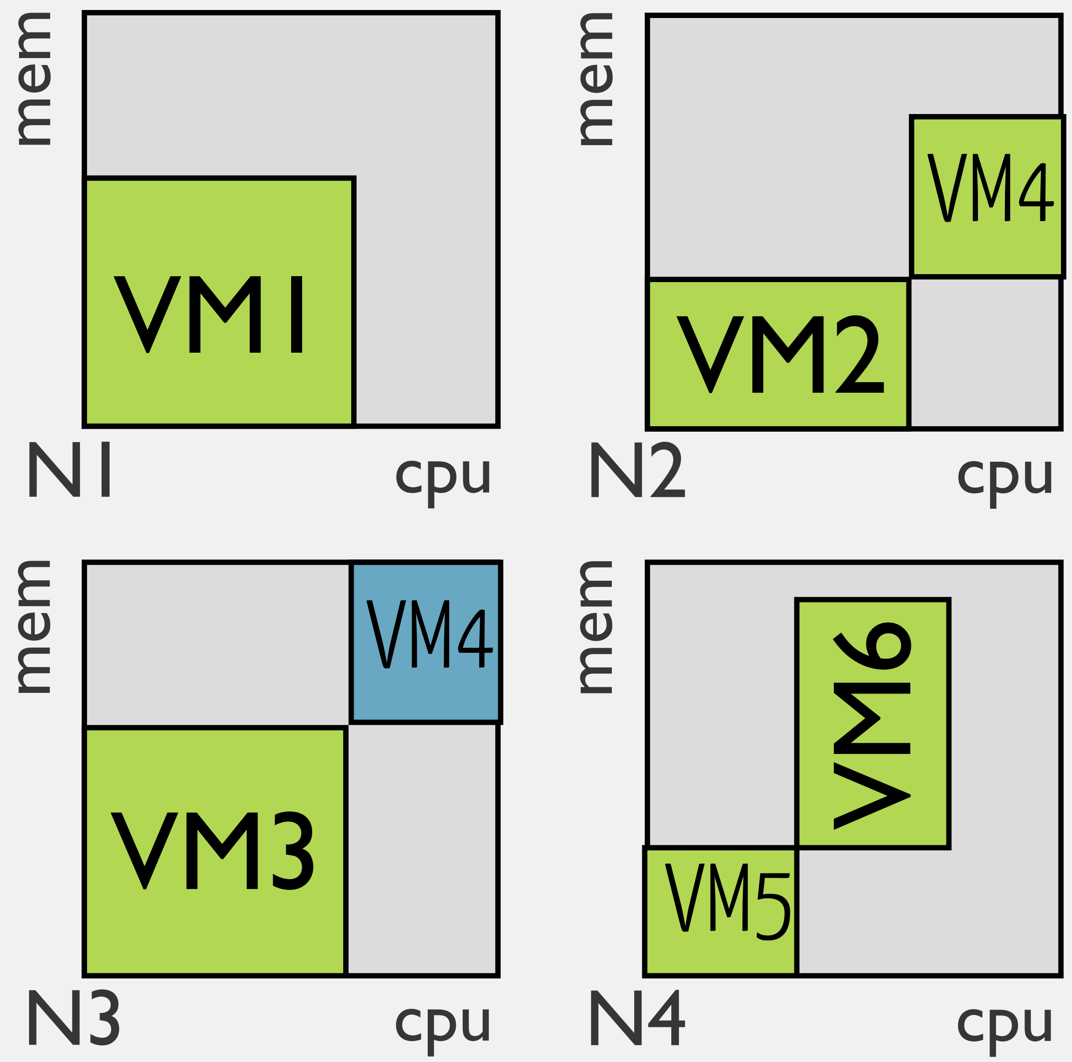
VM duplication between 2 phases
to simulate the migration

easy to implement

hard to manipulate to
compute long term previsions

how to support migrations

a simple way



n-phases vector packing

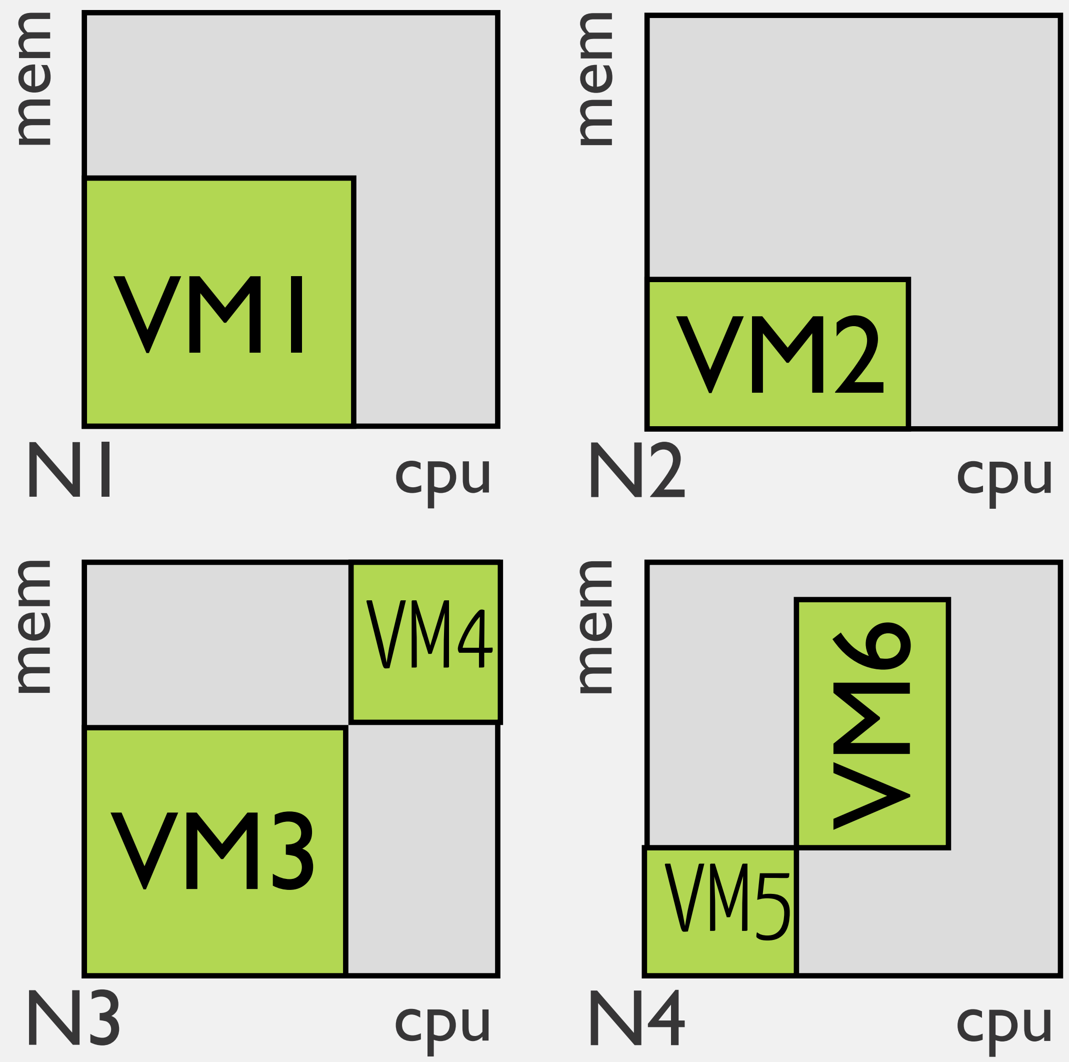
VM duplication between 2 phases
to simulate the migration

easy to implement

hard to manipulate to
compute long term previsions

how to support migrations

a simple way



n-phases vector packing

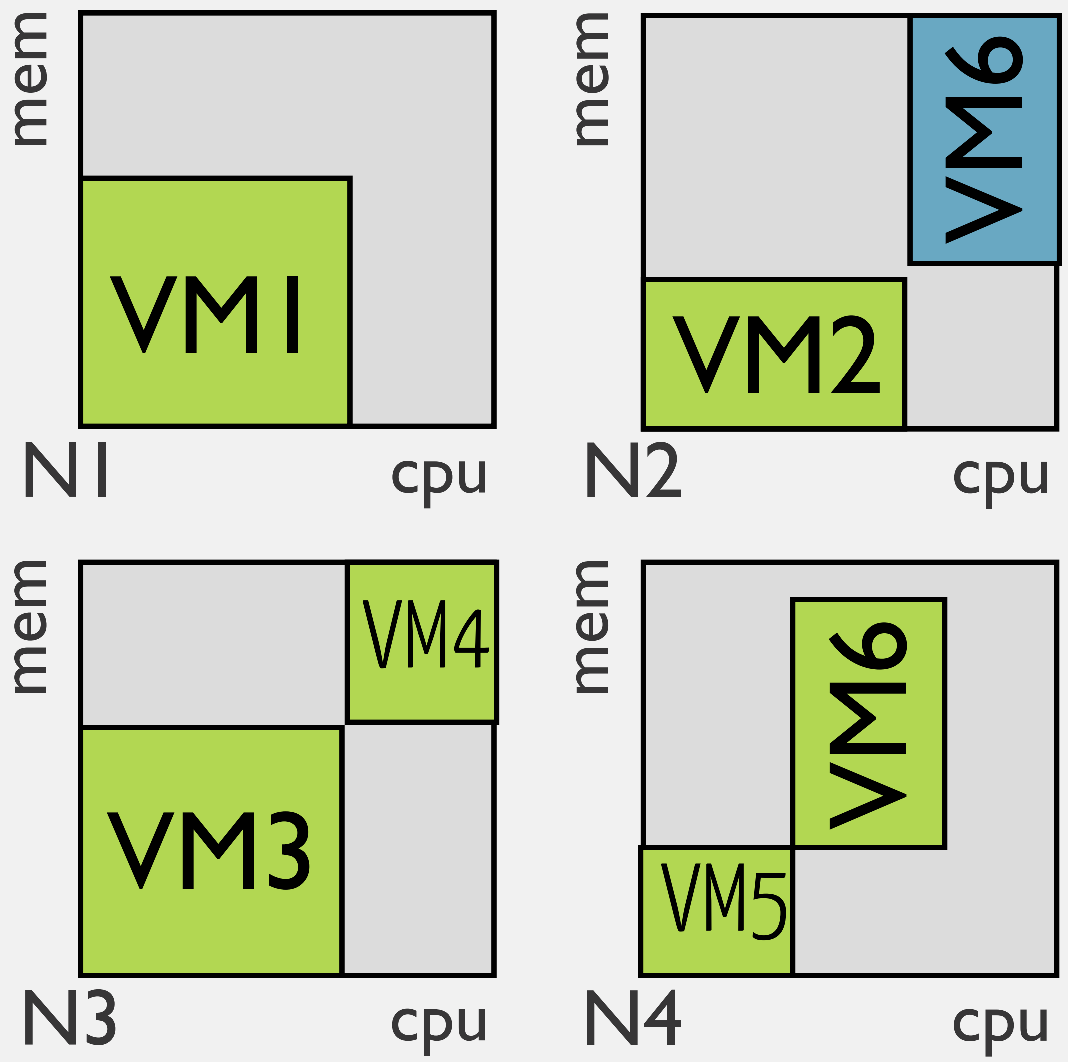
VM duplication between 2 phases
to simulate the migration

easy to implement

hard to manipulate to
compute long term previsions

how to support migrations

a simple way



n-phases vector packing

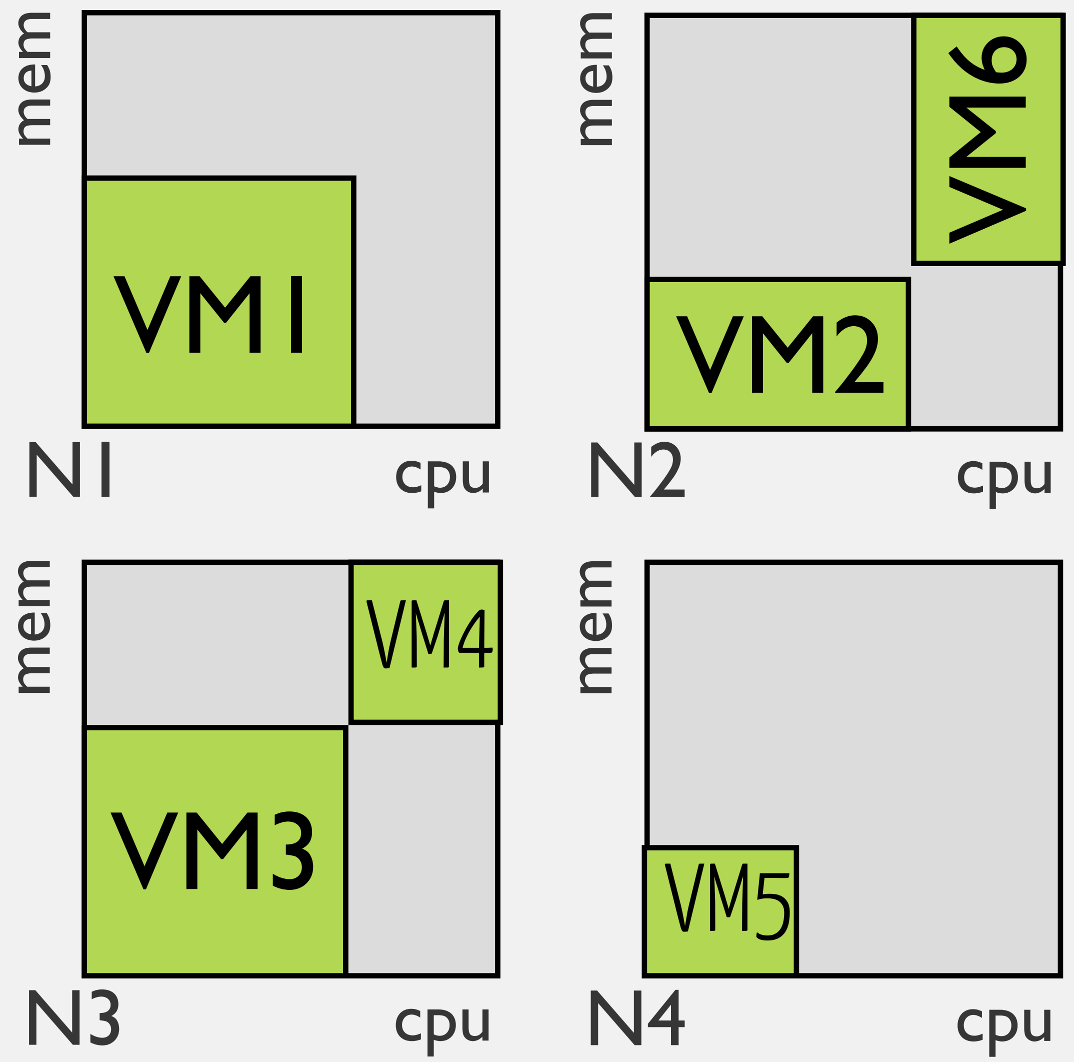
VM duplication between 2 phases
to simulate the migration

easy to implement

hard to manipulate to
compute long term previsions

how to support migrations

a simple way



n-phases vector packing

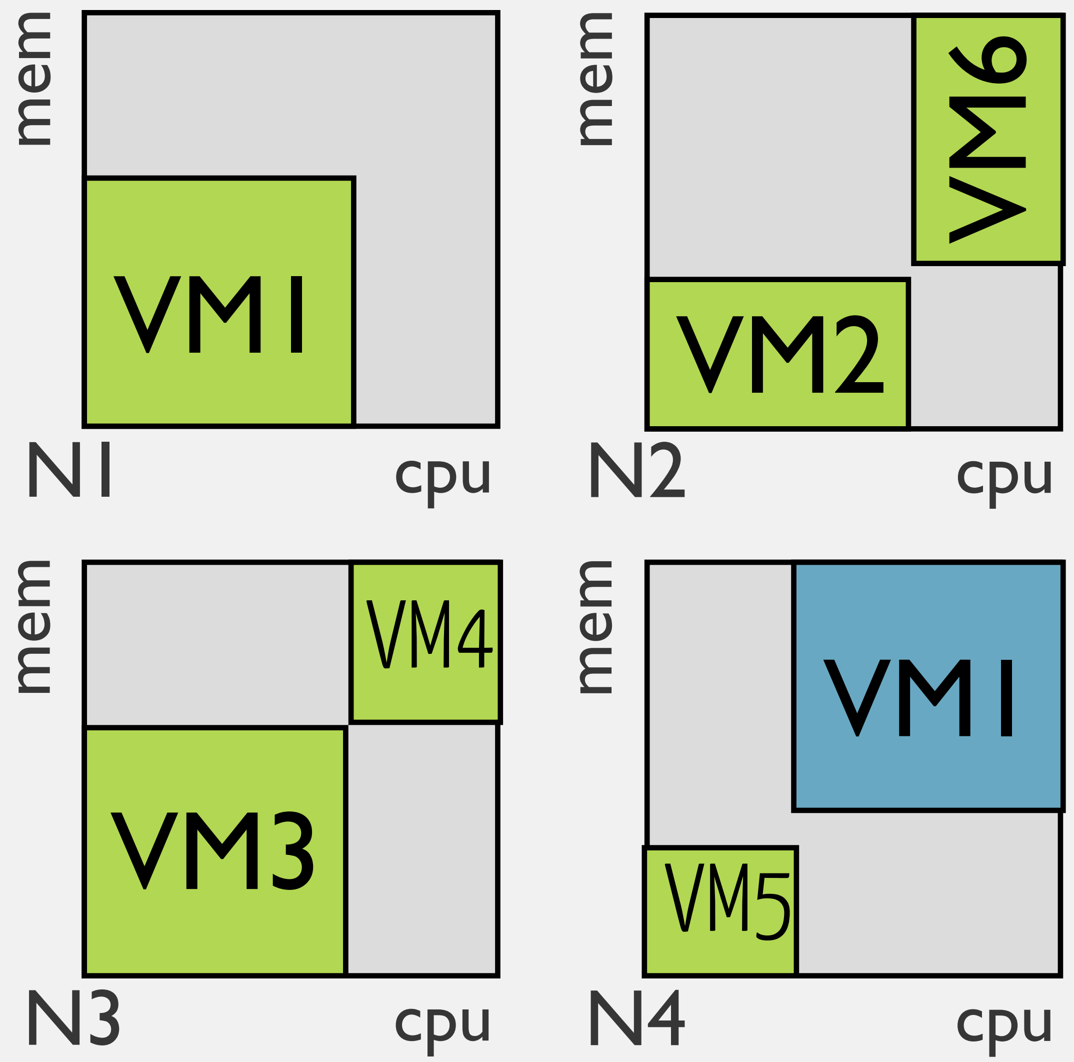
VM duplication between 2 phases
to simulate the migration

easy to implement

hard to manipulate to
compute long term previsions

how to support migrations

a simple way



n-phases vector packing

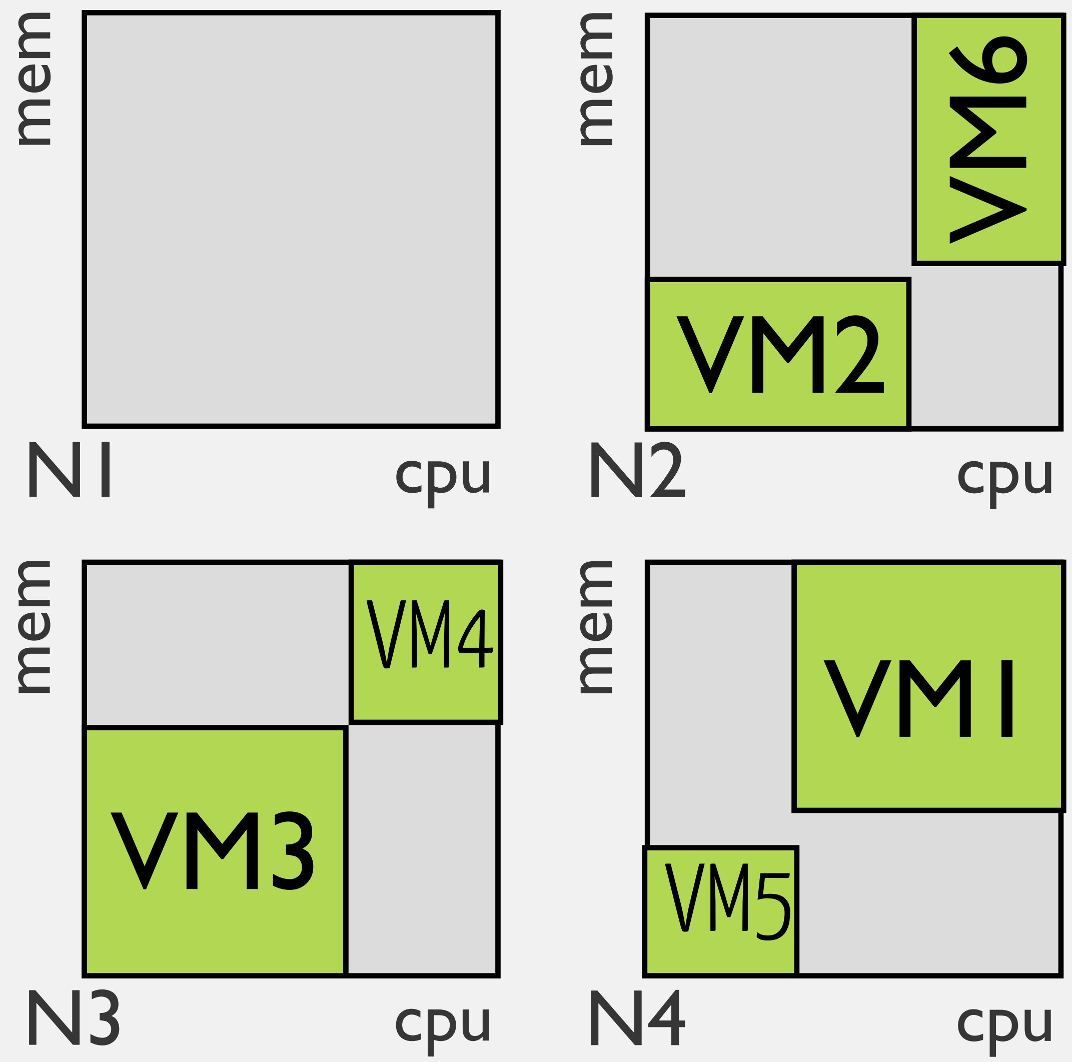
VM duplication between 2 phases
to simulate the migration

easy to implement

hard to manipulate to
compute long term previsions

how to support migrations

a simple way



n-phases vector packing

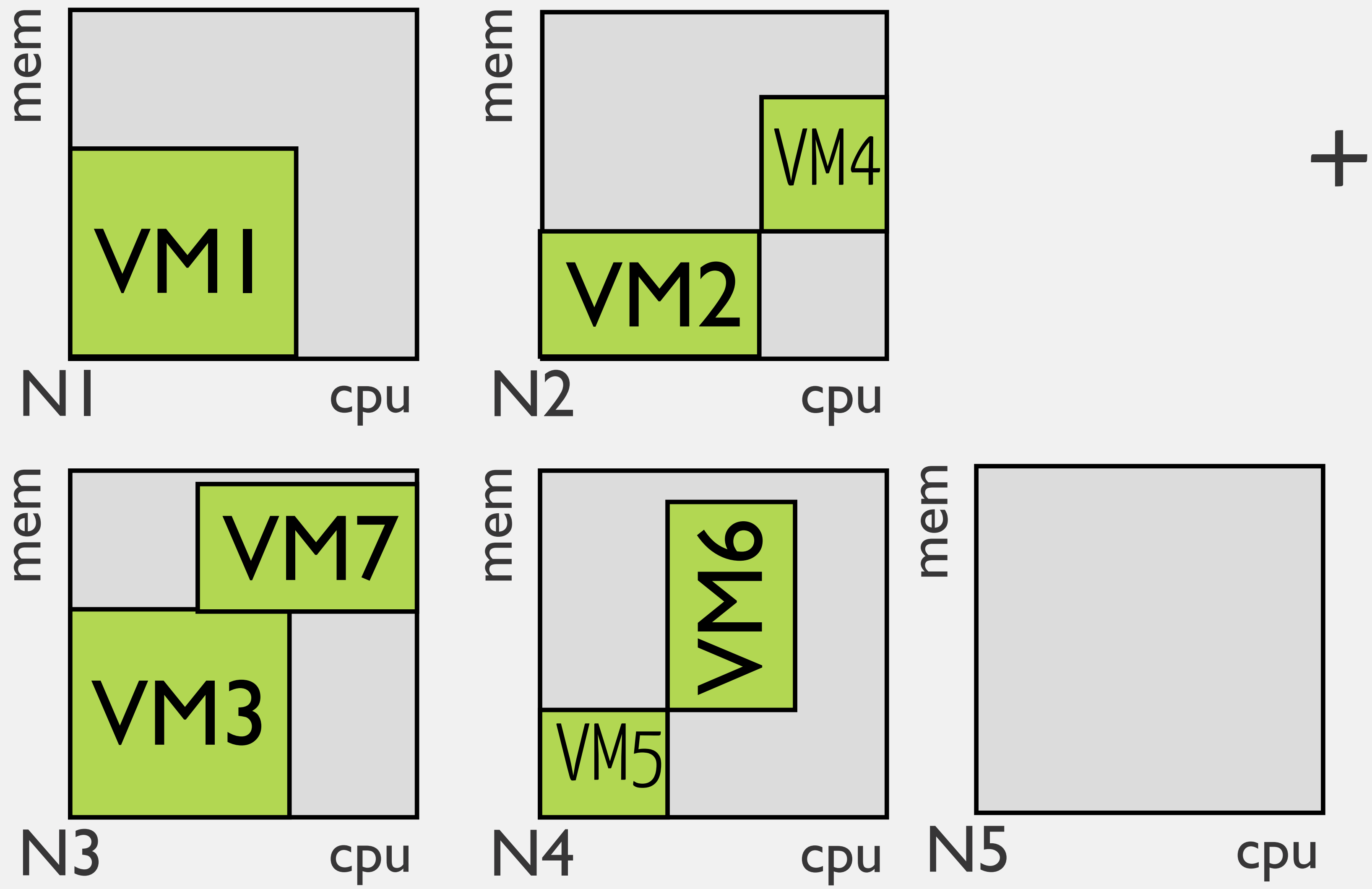
VM duplication between 2 phases
to simulate the migration

easy to implement

hard to manipulate to
compute long term previsions

how to support migrations

the alternative way



offline(N2) + no CPU sharing

1-phase

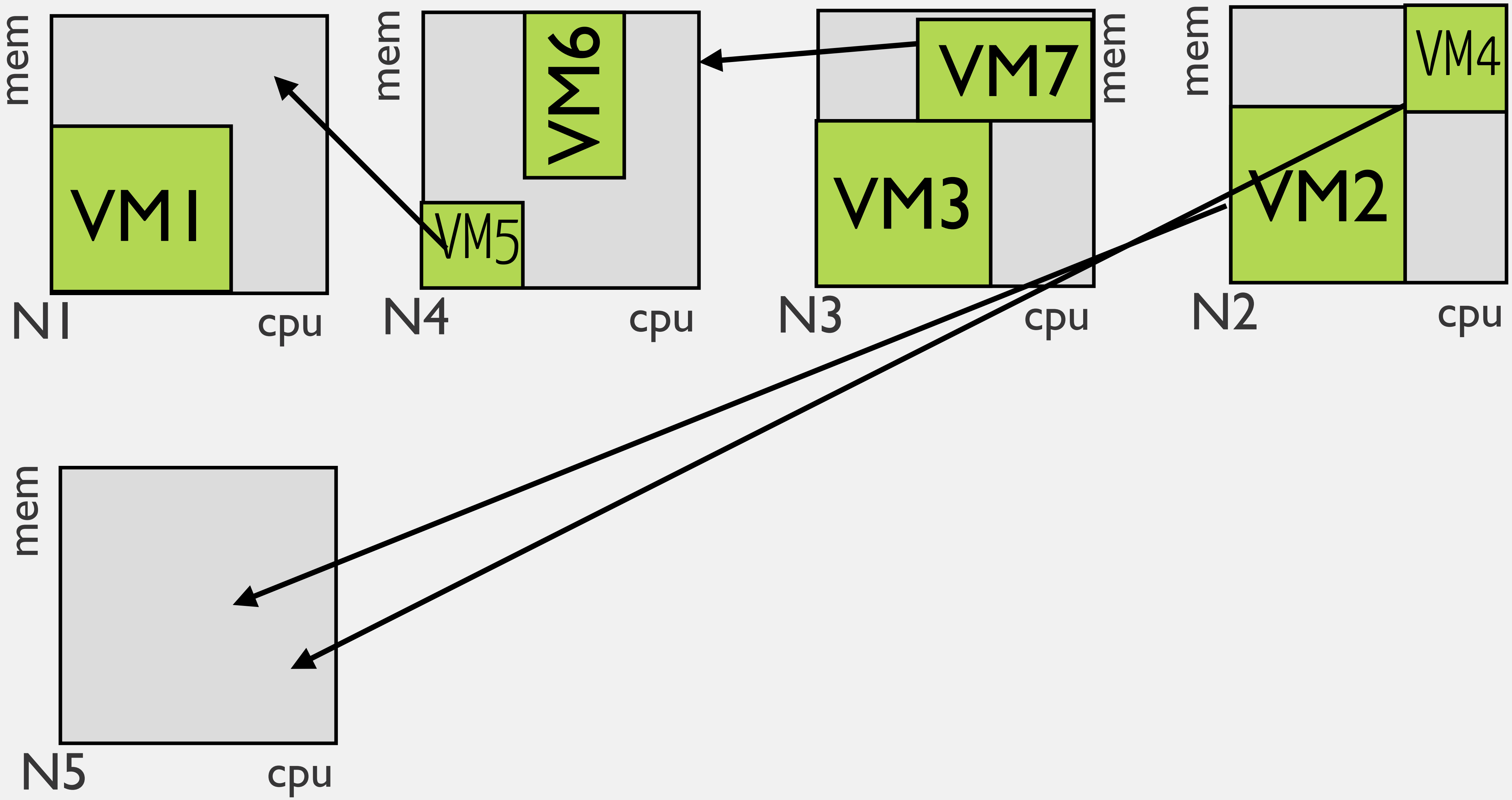
+ compute dependencies
step by steps

ok to implement

/!\ cycles
#migrations

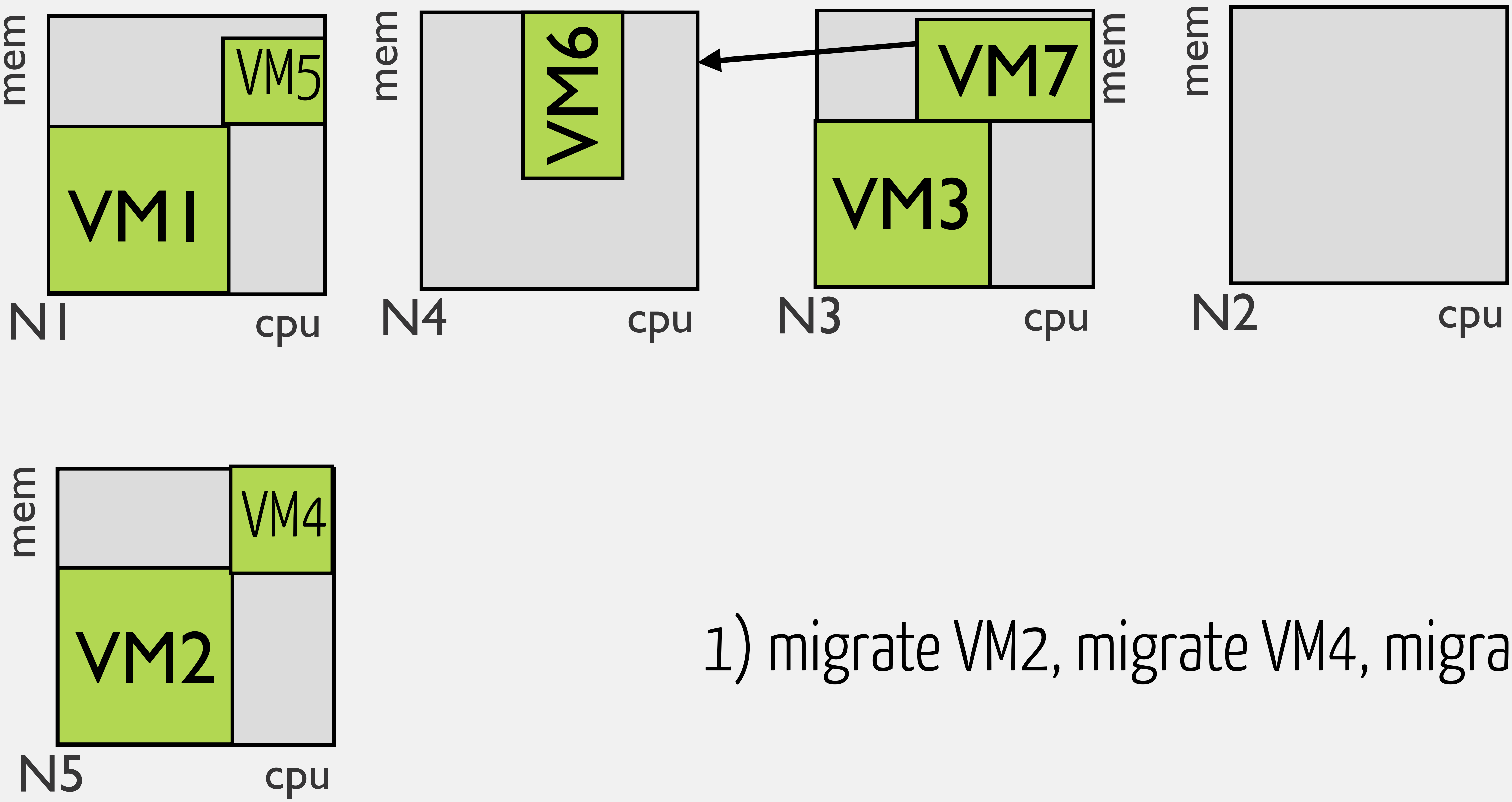
how to support migrations

the alternative way



how to support migrations

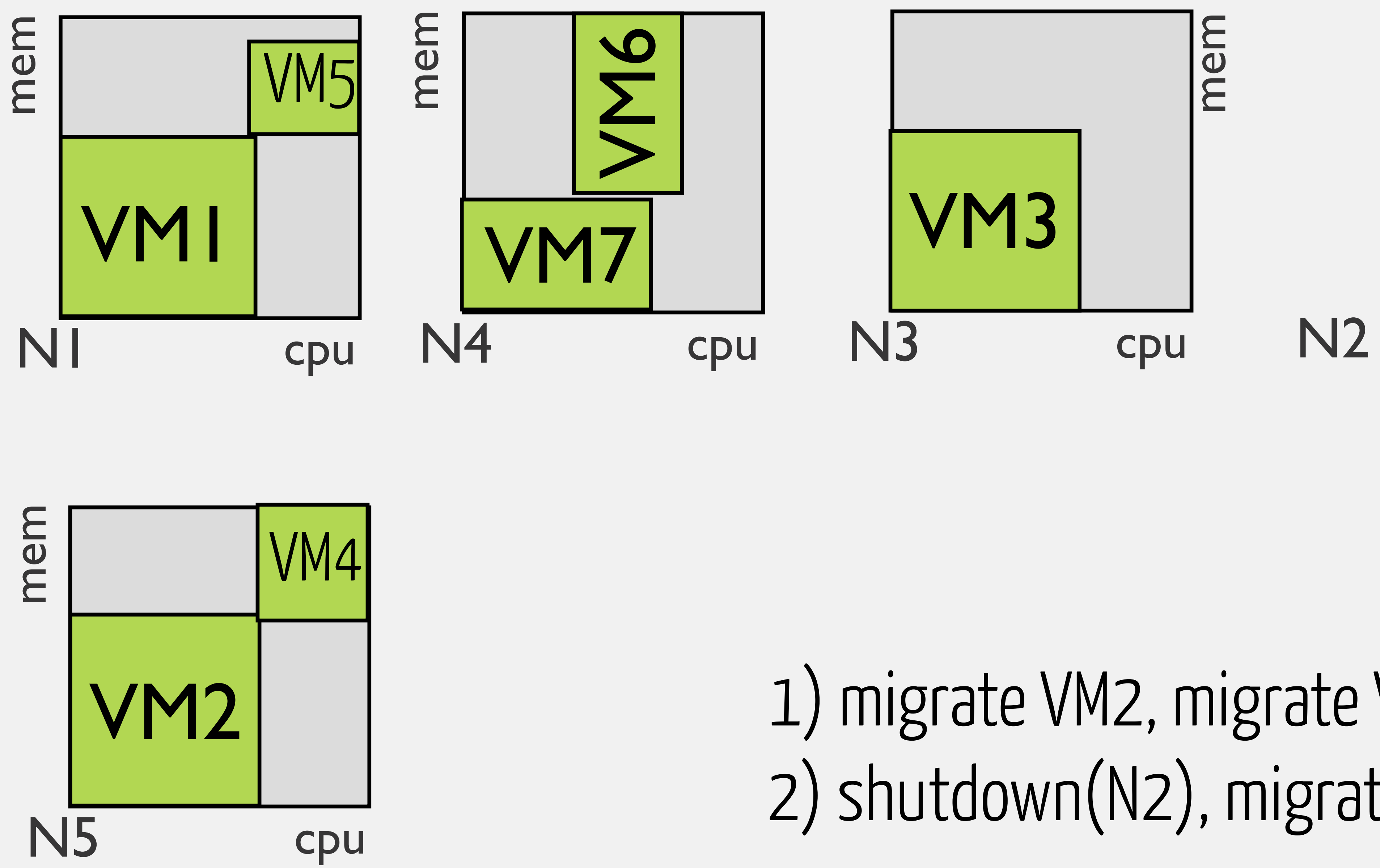
the alternative way



1) migrate VM2, migrate VM4, migrate VM5

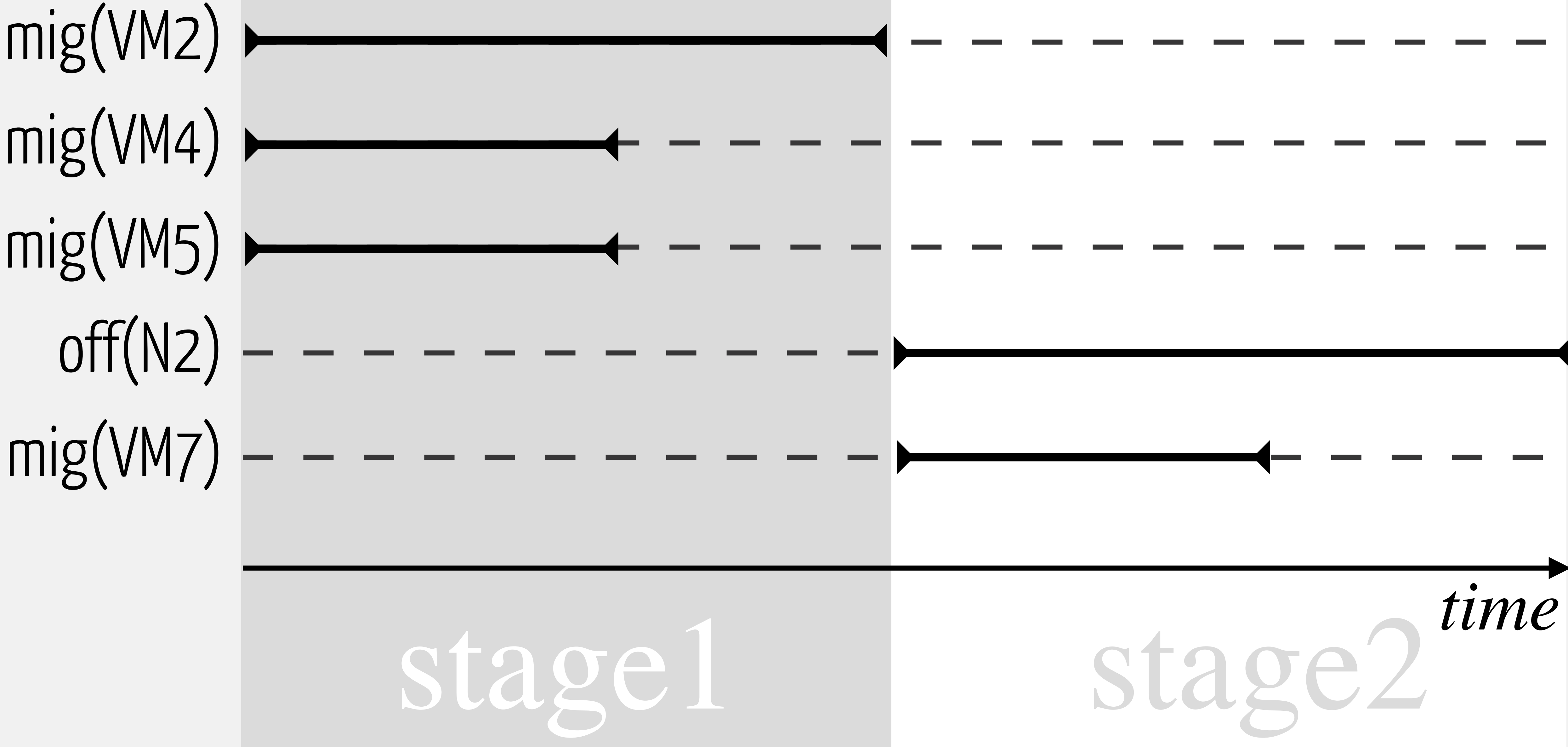
how to support migrations

the alternative way



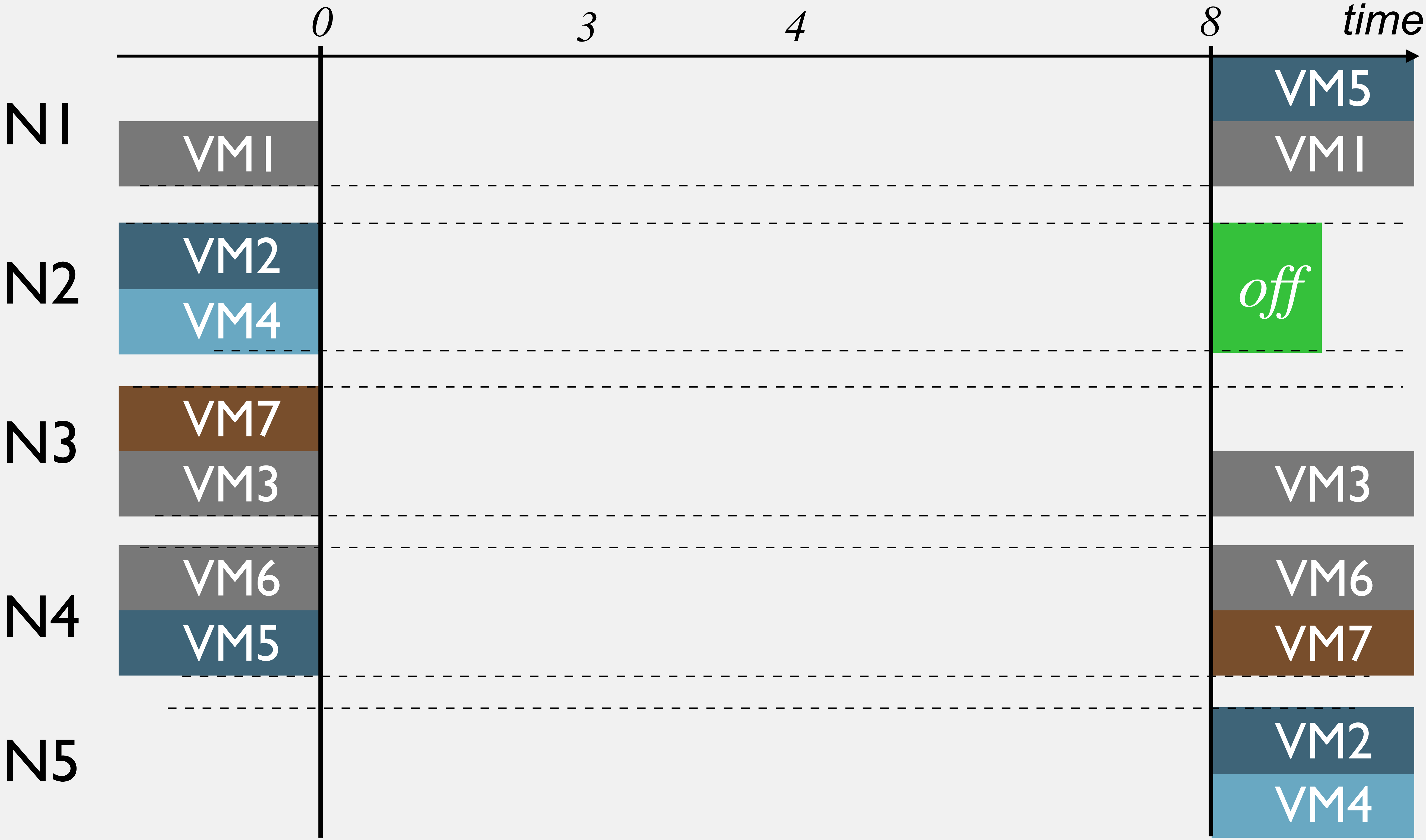
1) migrate VM2, migrate VM4, migrate VM5
2) shutdown(N2), migrate VM7

coarse grain staging
delay actions



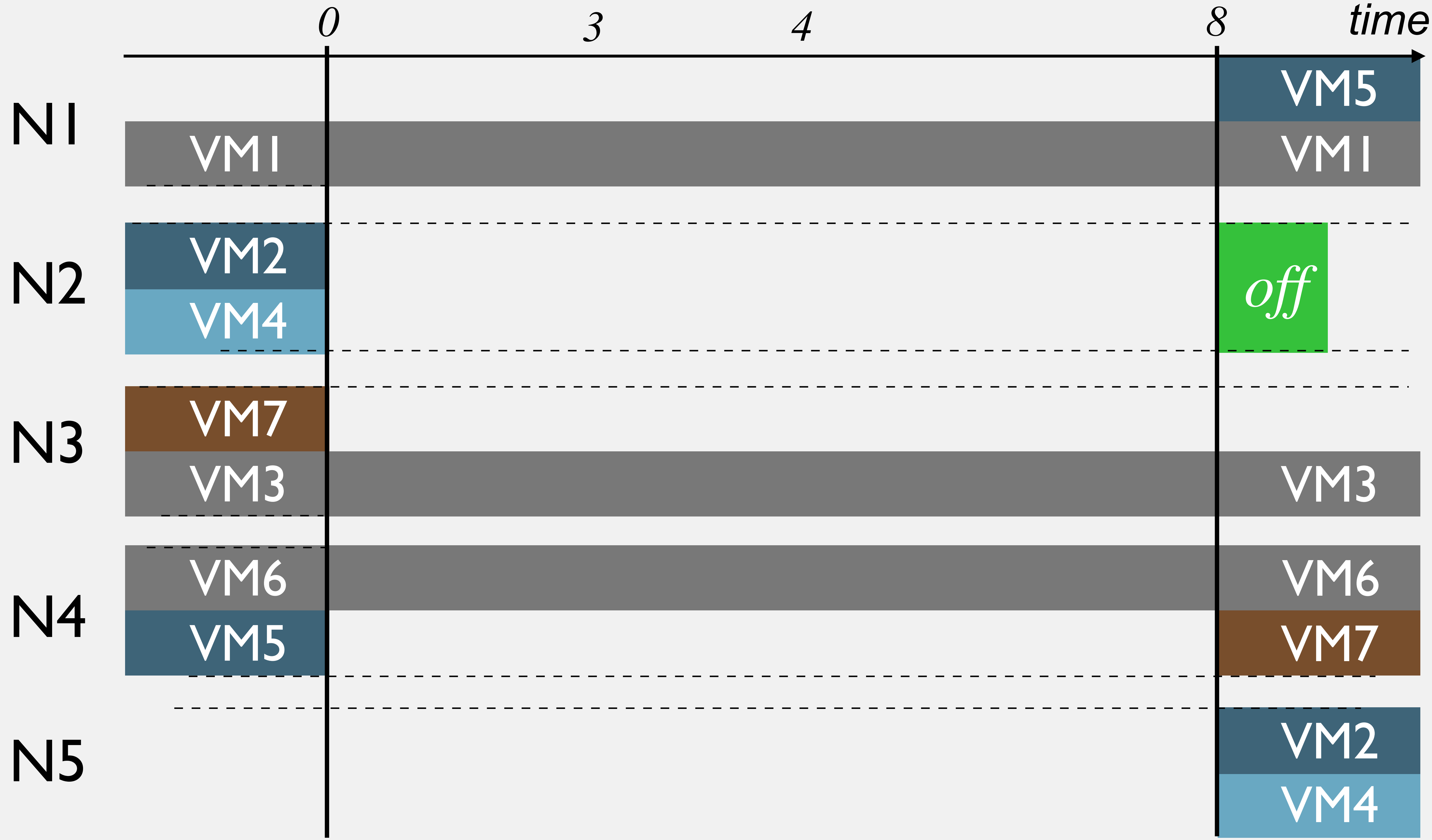
Resource-Constrained Project Scheduling Problem

the clean way to model space and time



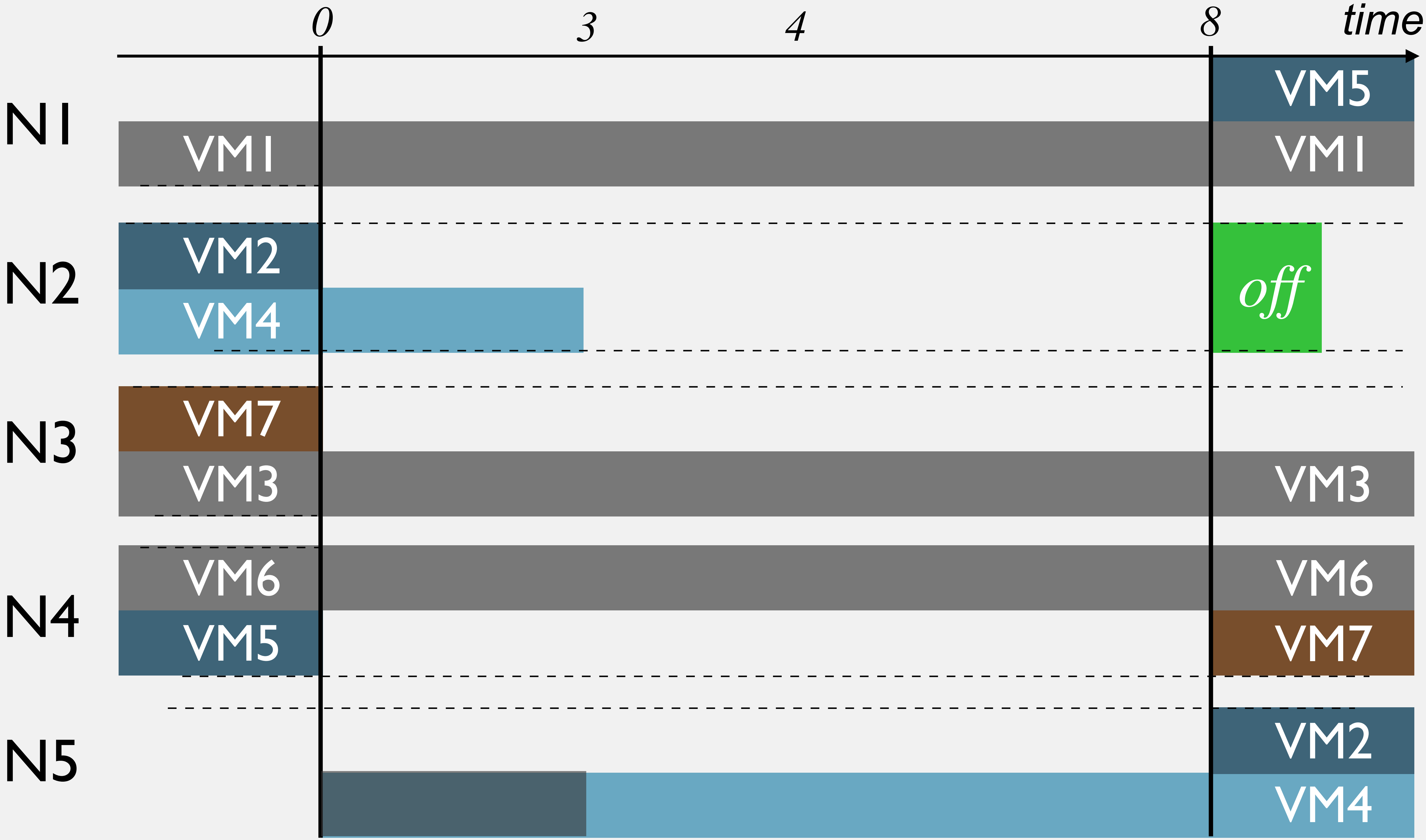
Resource-Constrained Project Scheduling Problem

the clean way to model space and time



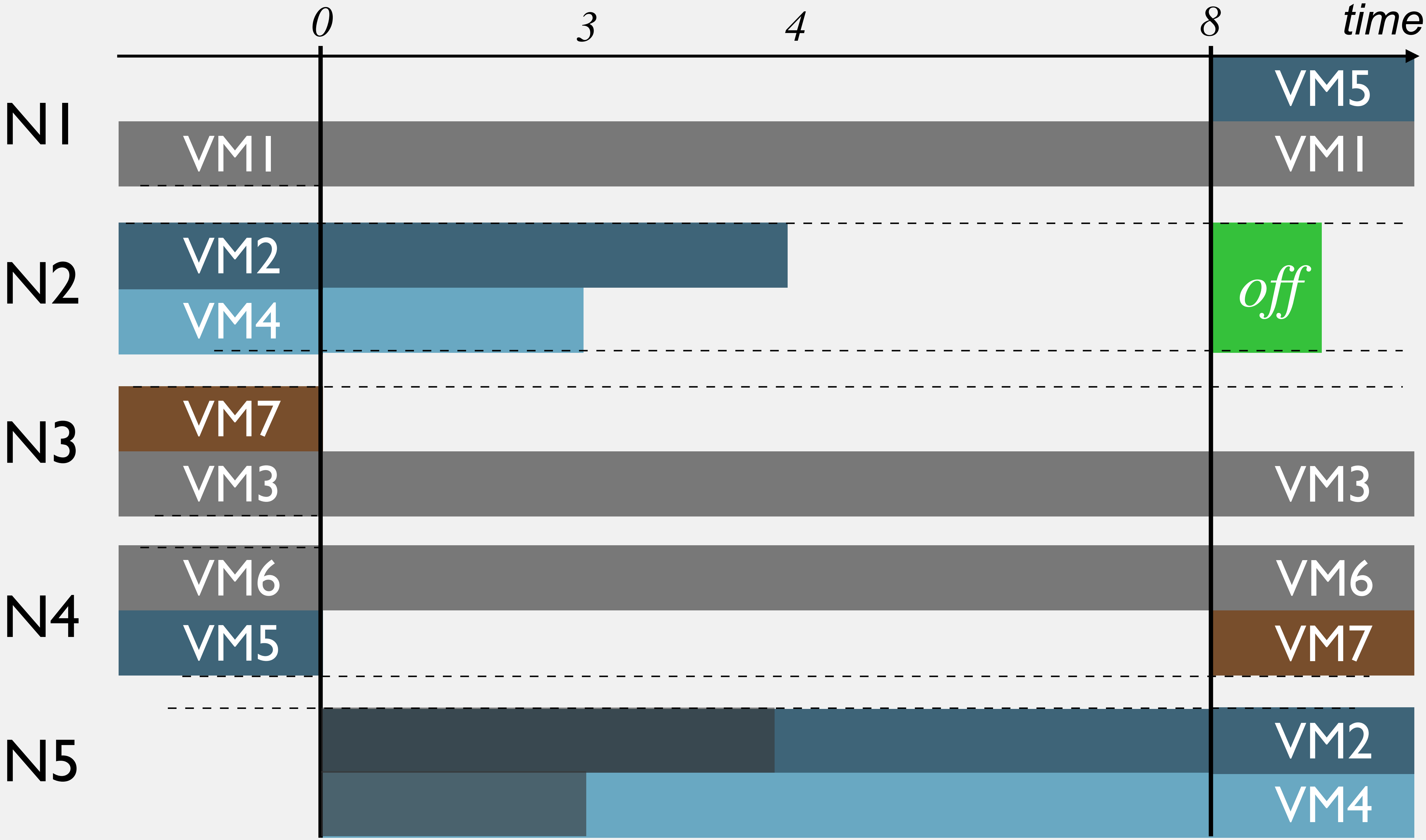
Resource-Constrained Project Scheduling Problem

the clean way to model space and time



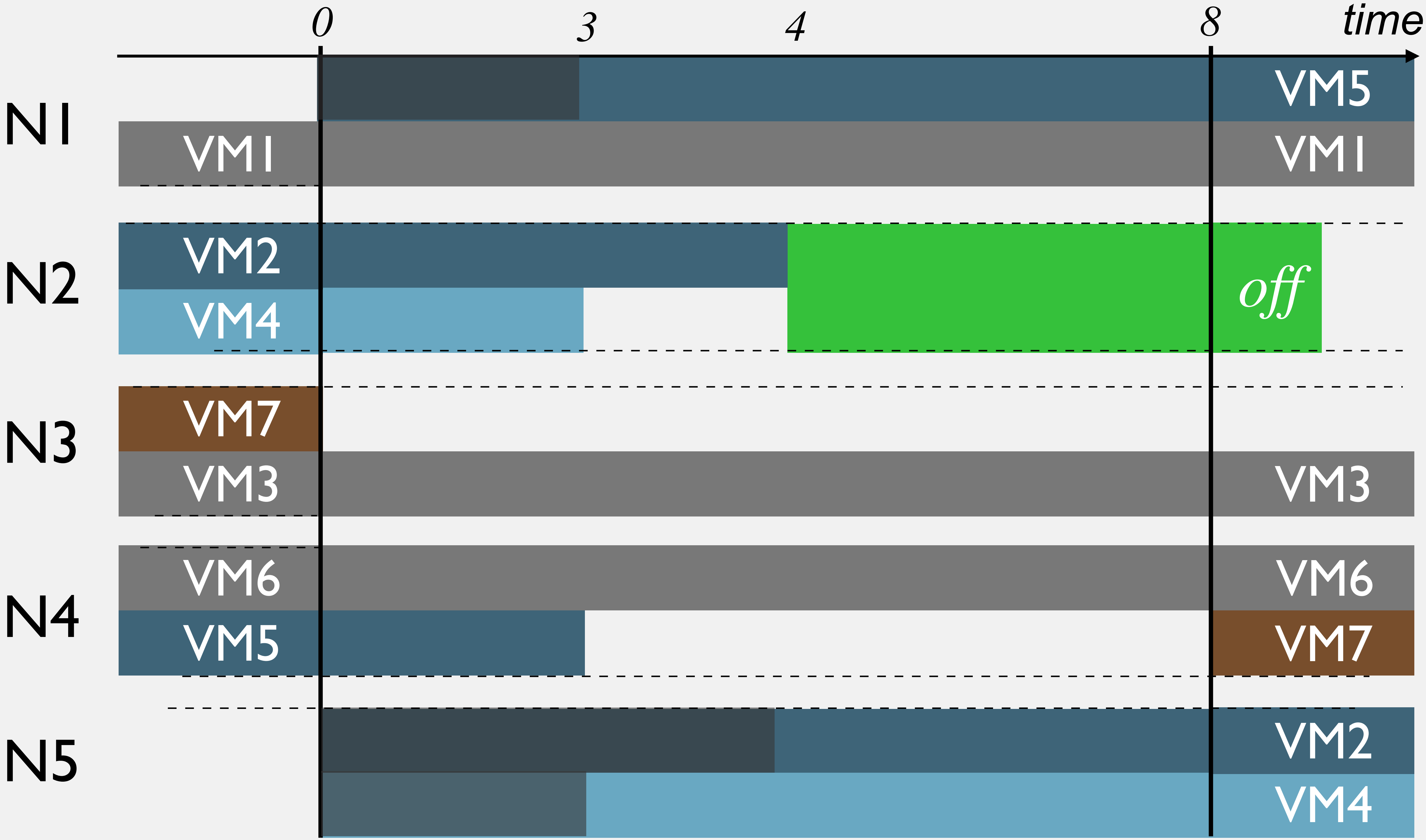
Resource-Constrained Project Scheduling Problem

the clean way to model space and time



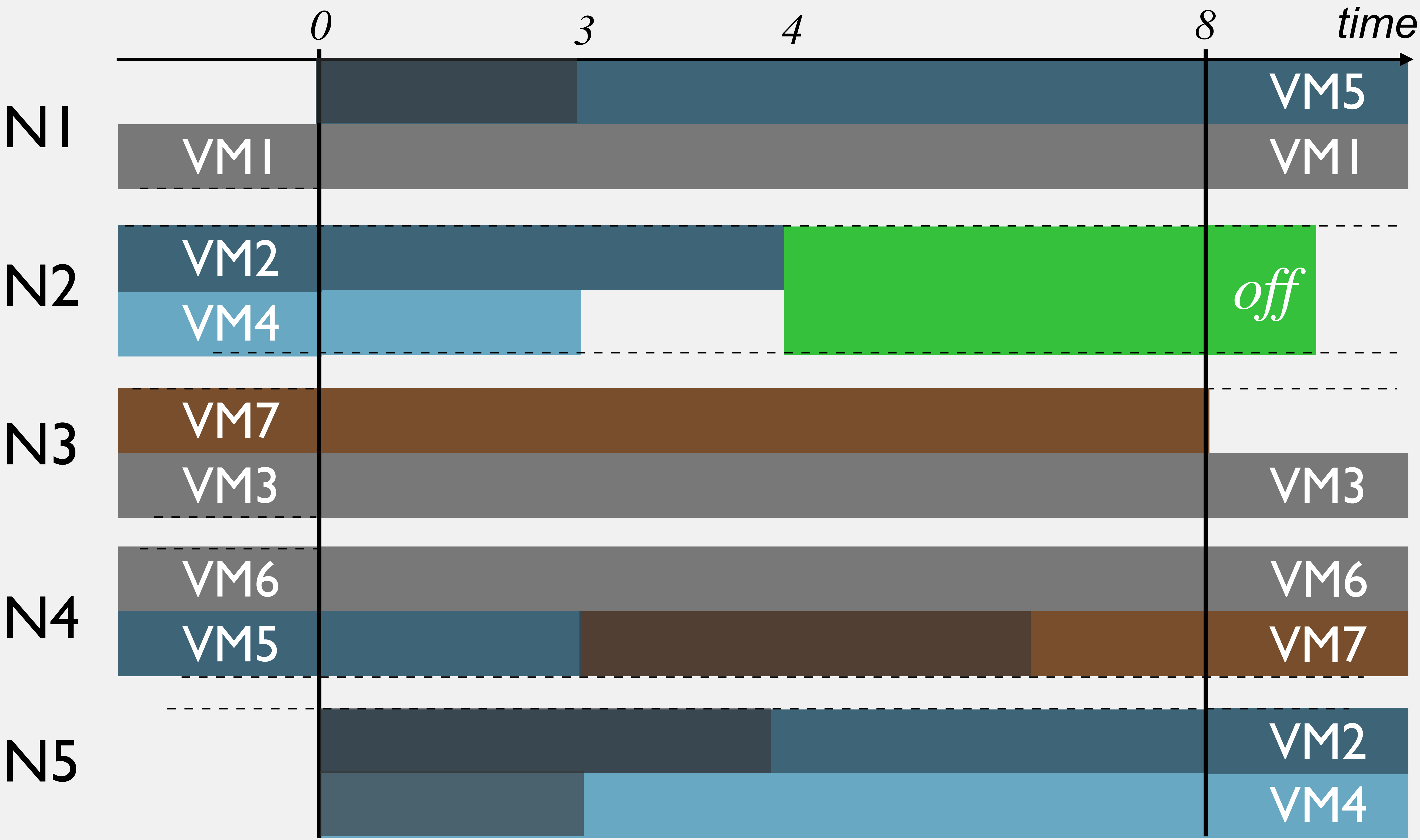
Resource-Constrained Project Scheduling Problem

the clean way to model space and time



Resource-Constrained Project Scheduling Problem

the clean way to model space and time



Resource-Constrained Project Scheduling Problem

the pure way to support migrations

1 resource per (node x dimension), bounded capacity

tasks to model the VM lifecycle.
height to model a consumption
width to model a duration

at any moment, the cumulative task consumption on a resource cannot exceed its capacity

comfortable to express continuous optimisation

very hard to implement (properly)

From a theoretical schedule to a practical one

duration may be longer

convert to an event based schedule

```
0:3 - migrate VM4
0:3 - migrate VM5
0:4 - migrate VM2
3:8 - migrate VM7
4:8 - shutdown(N2)
```

→

```
- : migrate VM4
- : migrate VM5
- : migrate VM2
!migrate(VM2) & !migrate(VM4):   shutdown(N2)
!migrate(VM5):  migrate VM7
```

Back to vector packing based approaches

Fit Fit Decrease (FFD)

basic VM scheduling

```
sort VMs in desc order
for each VM
   pick the first suitable node
```

Fit Fit Decrease (FFD)

basic VM scheduling

```
sort VMs in desc order
for each VM
   pick the first suitable node
```

difficult VMs first !

Fit Fit Decrease (FFD)

basic VM scheduling

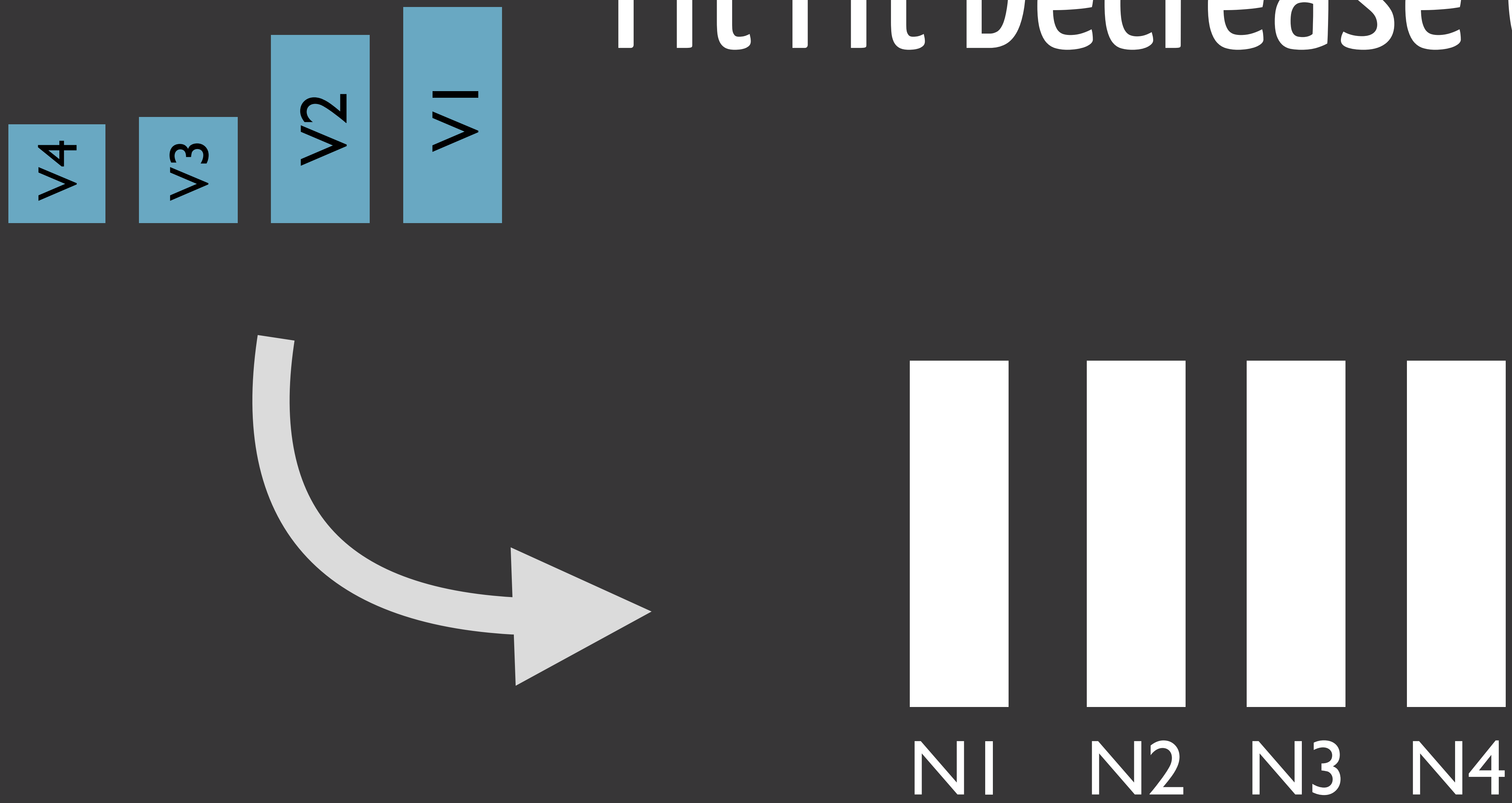
```
sort VMs in desc order
for each VM
   pick the first suitable node
```

difficult VMs first !

enough free resources

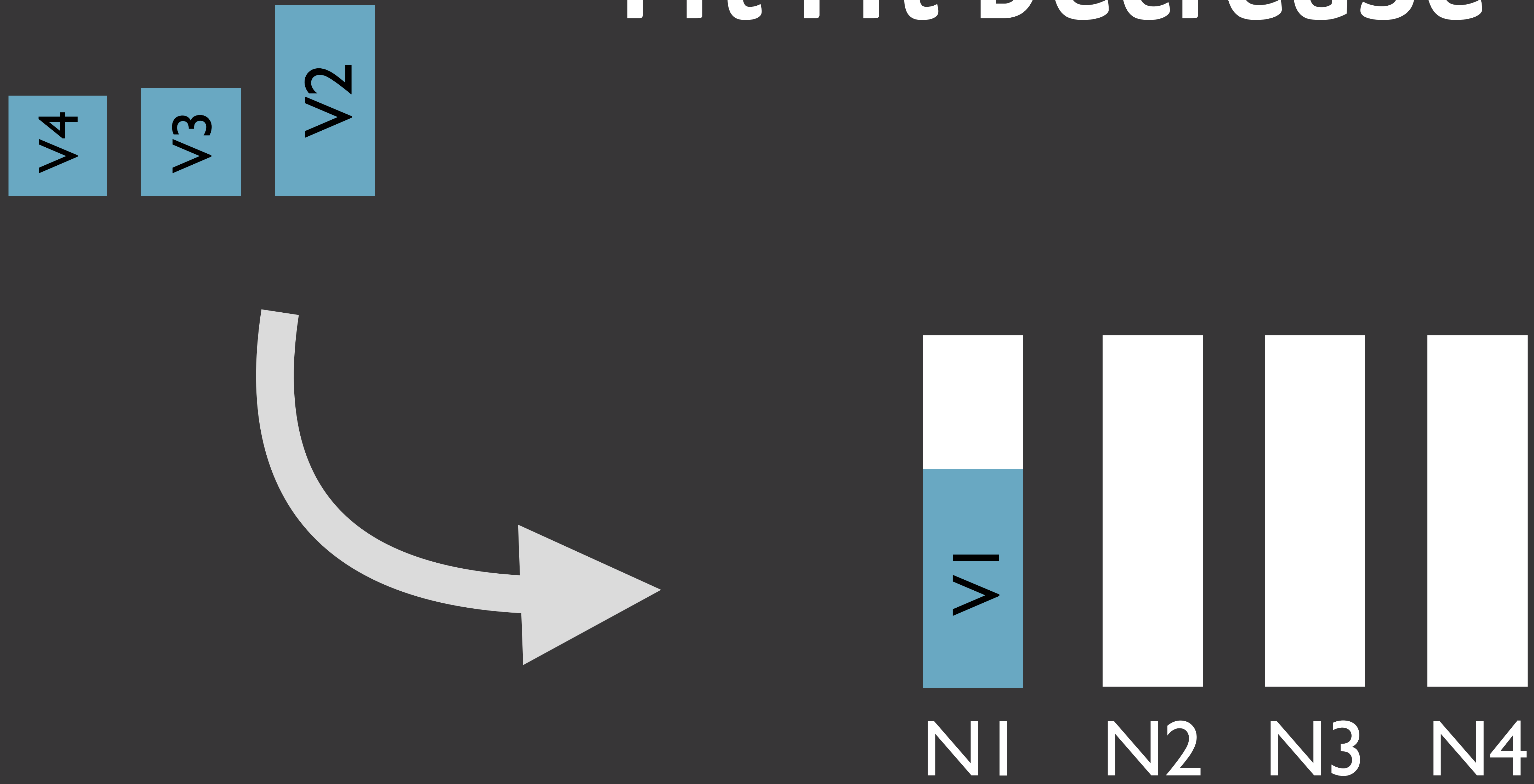
Fit Fit Decrease (FFD)

example



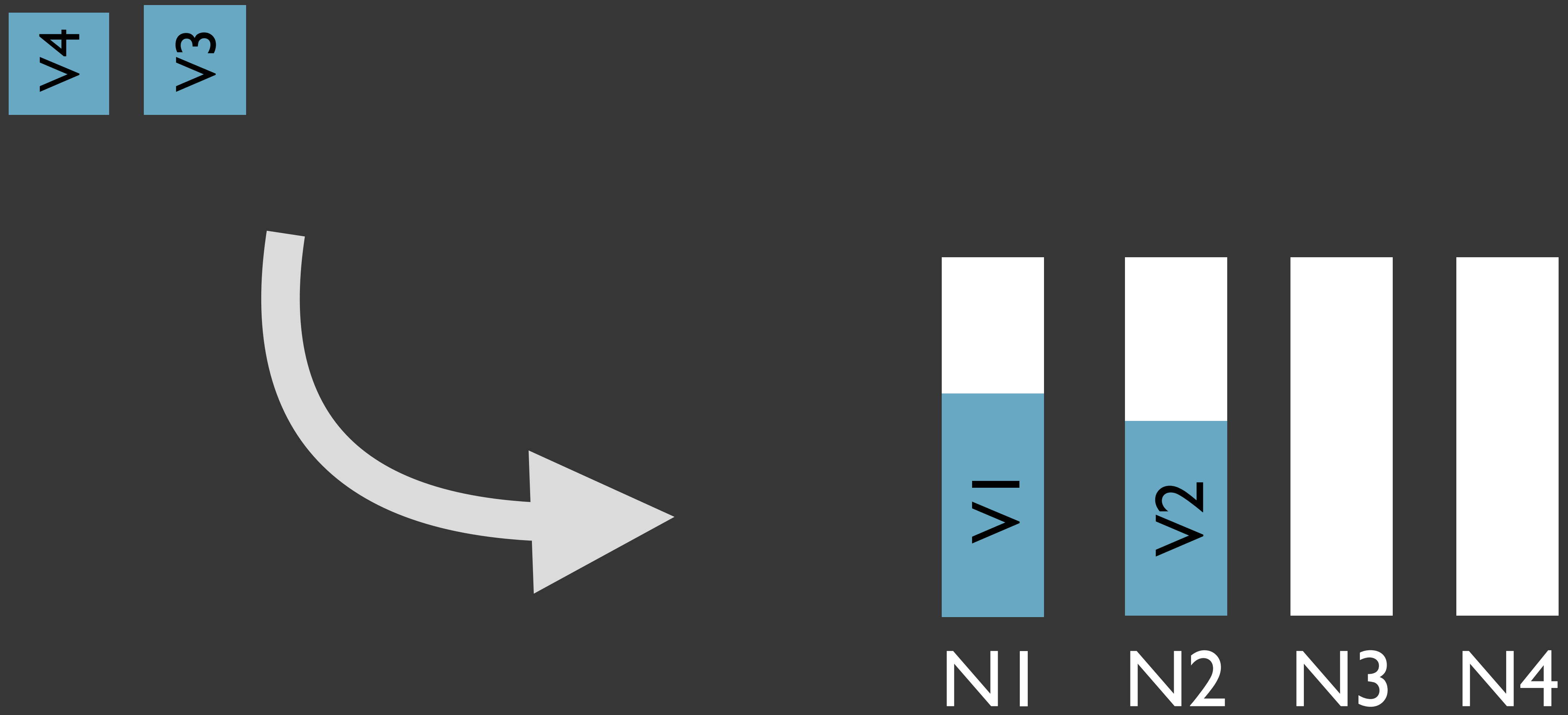
Fit Fit Decrease (FFD)

example



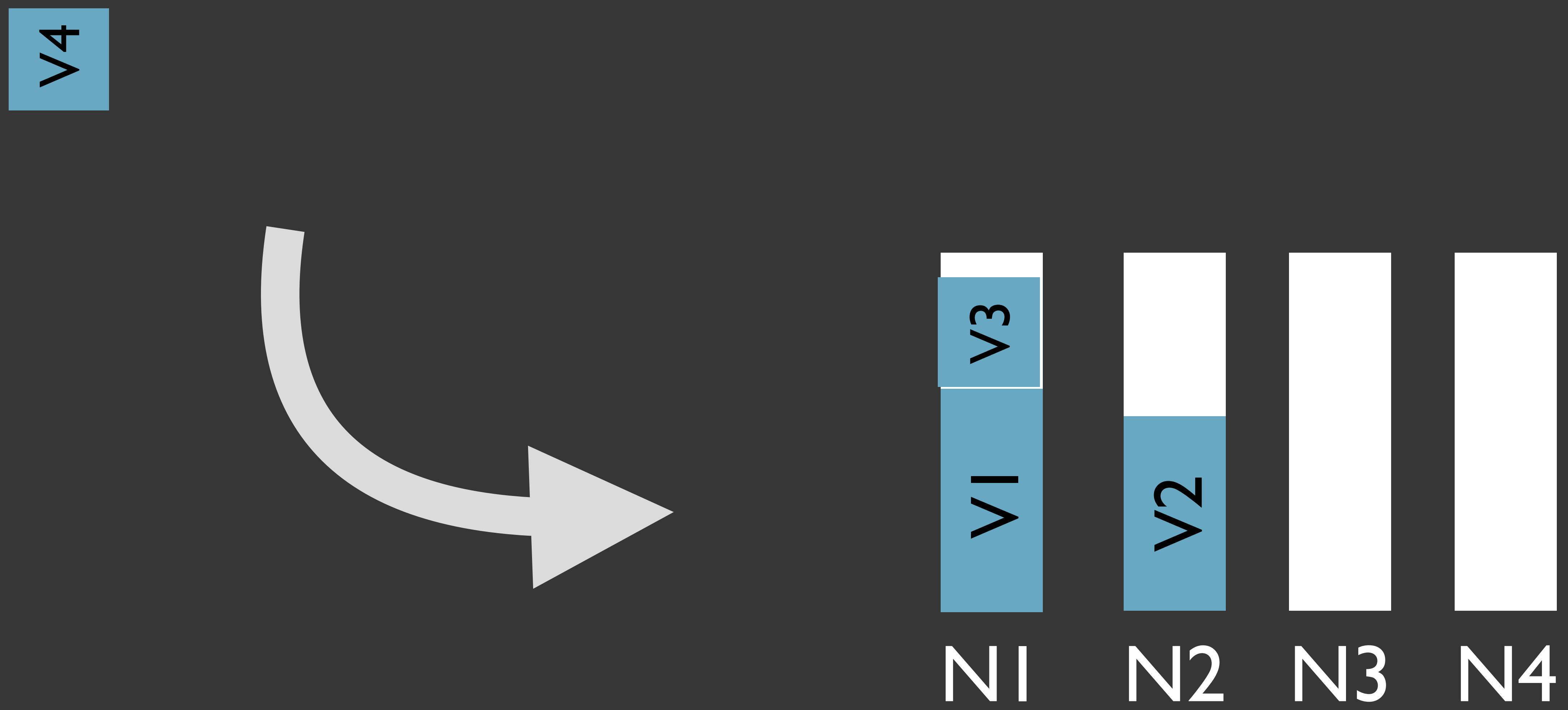
Fit Fit Decrease (FFD)

example



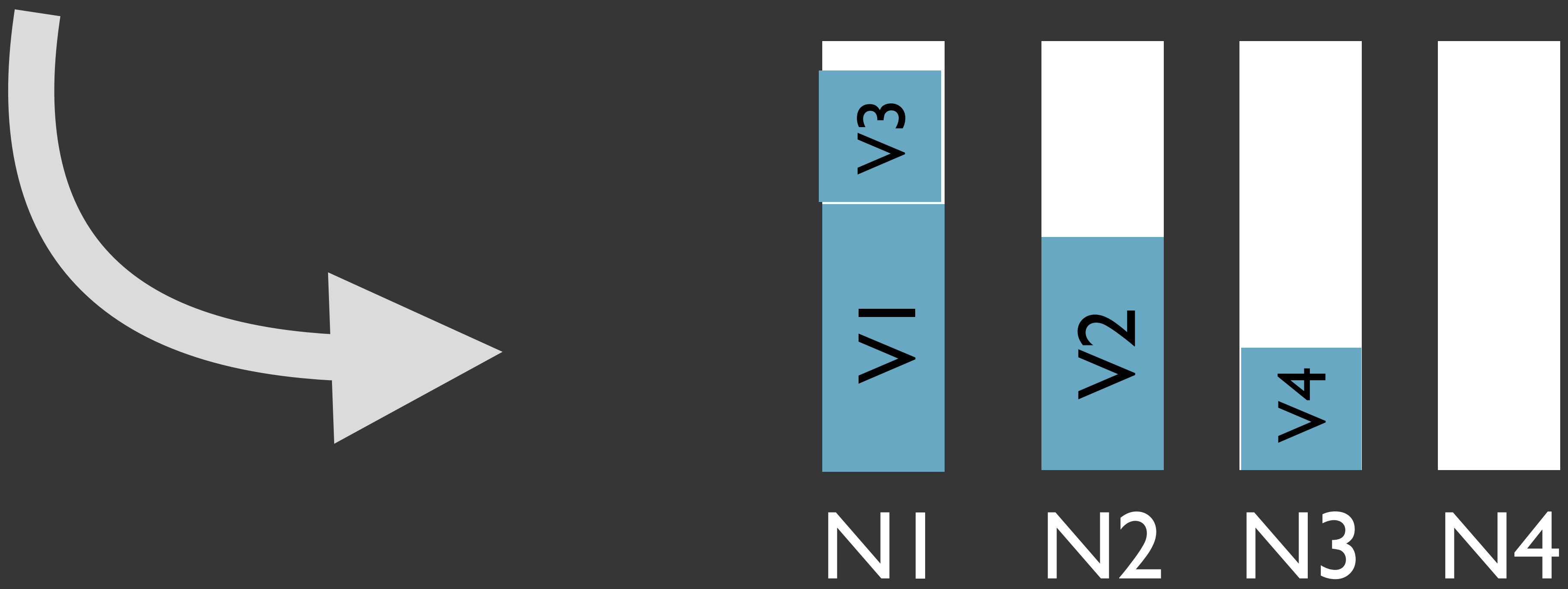
Fit Fit Decrease (FFD)

example

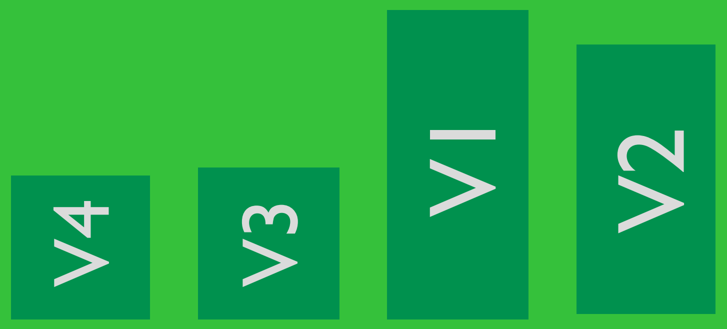


Fit Fit Decrease (FFD)

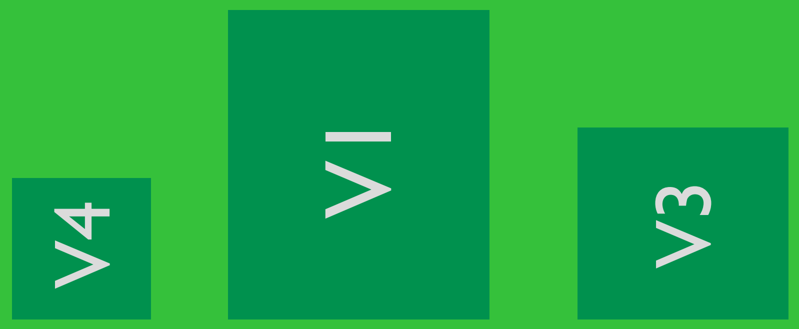
example



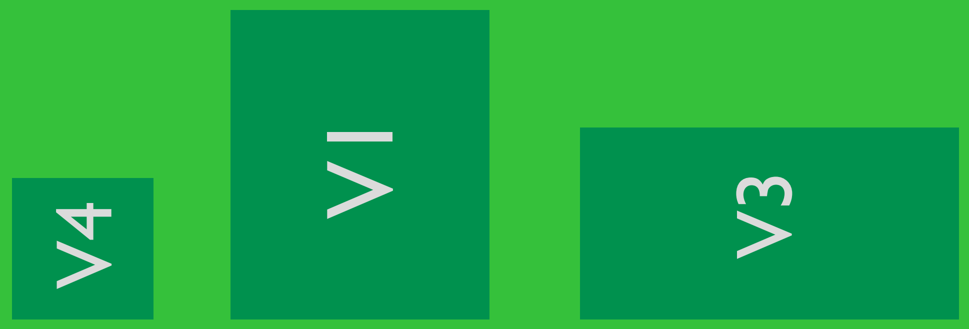
multi-dimension sorting ?



easy, 1 dimension is varying

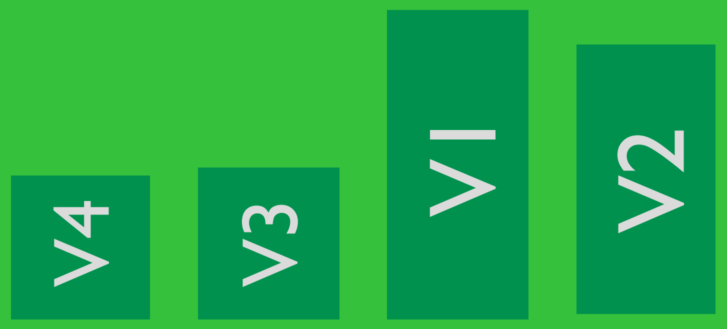


easy, uniform variation

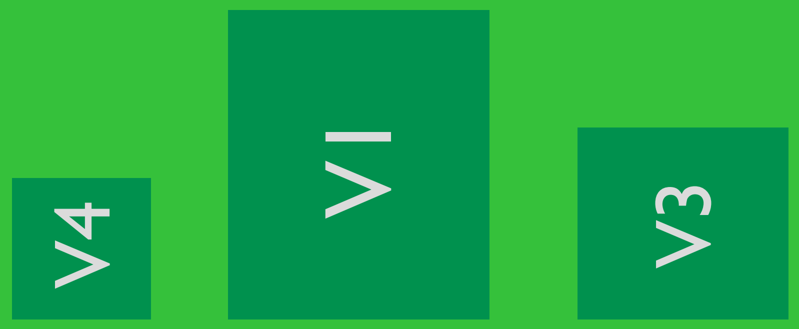


?

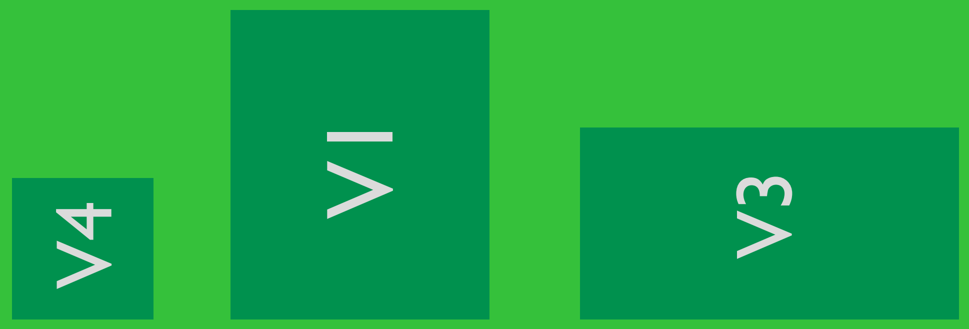
multi-dimension sorting ?



easy, 1 dimension is varying



easy, uniform variation



sort by dimension
aggregate dimensions
find the most critical one
...

Balancing VMs

Why?

Why?

to reduce loss in terms of failures
to reduce hotspots
to absorb load spikes

balancing

in theory

$$min(stddev([load(n_1), \ldots, load(n_i)]))$$

Worst Fit Decrease (WFD)

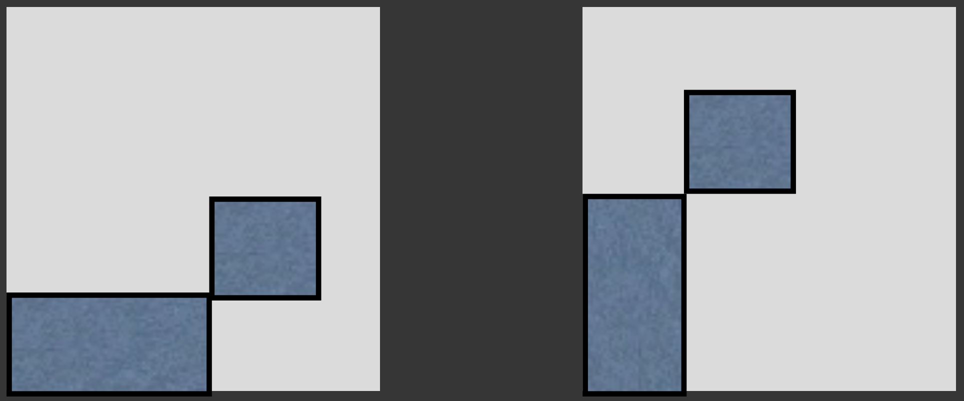
balancing, a practice

```
sort VMs in desc order
for each VM
 pick the suitable node
      with the highest remaining
      space
```

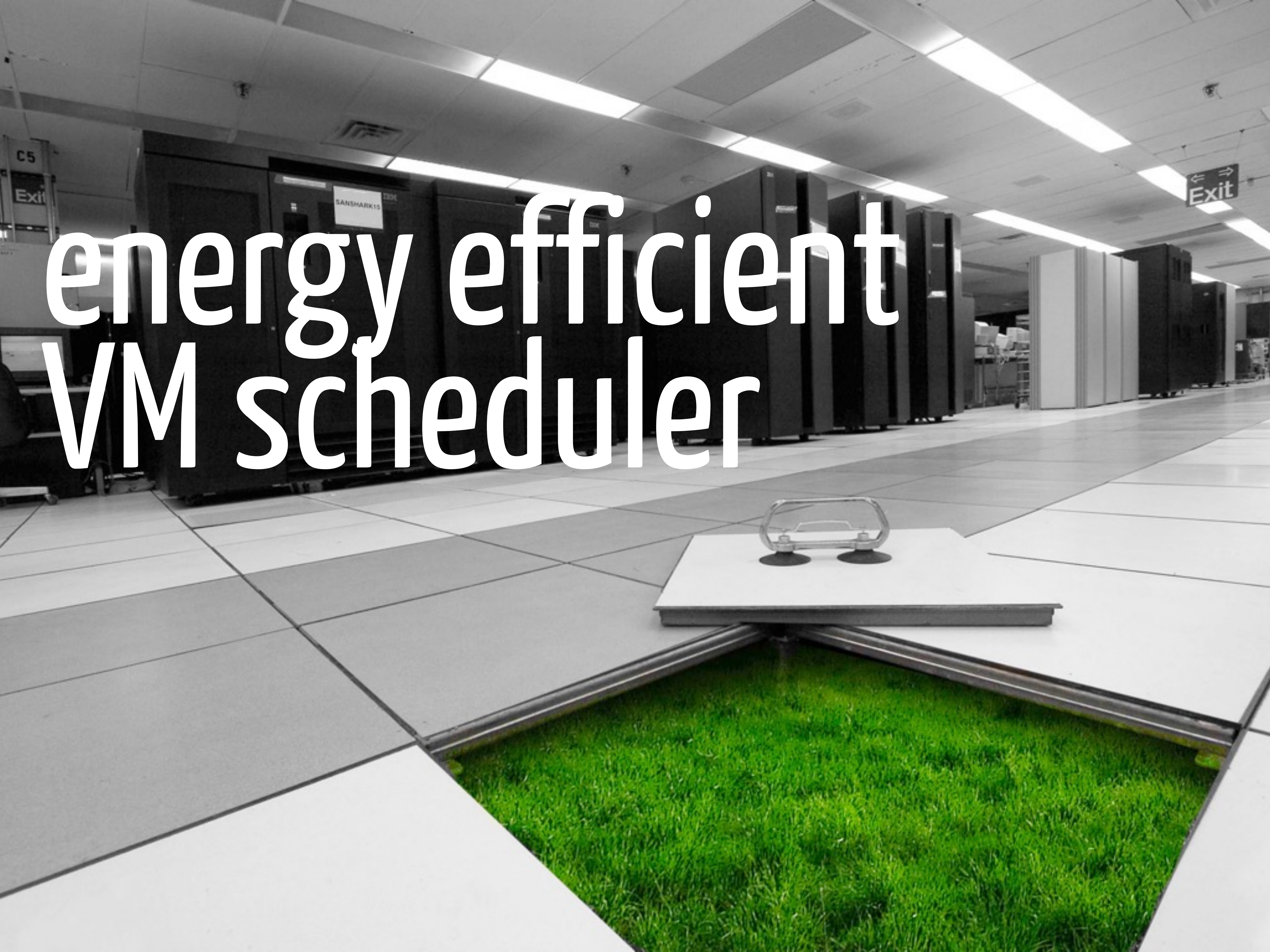
balancing over
multiple dimensions ?

balancing over multiple dimensions ?

dimension normalisation
worst dimension
dimension aggregation



energy efficient VM scheduler



USA

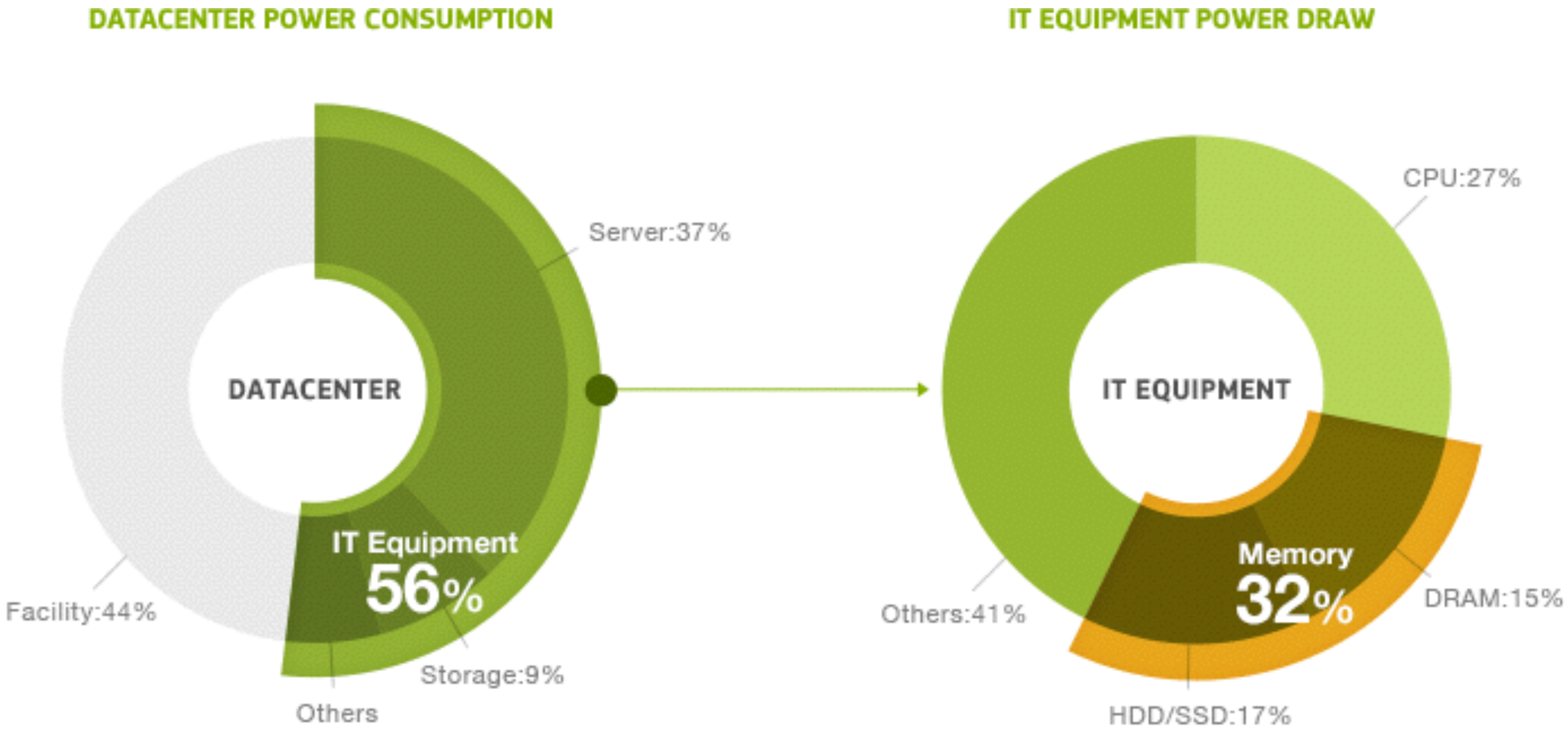
1.5%

of the 2005 budget

3 %

in 2010 ?

(Environmental Protection Agency, 2007)



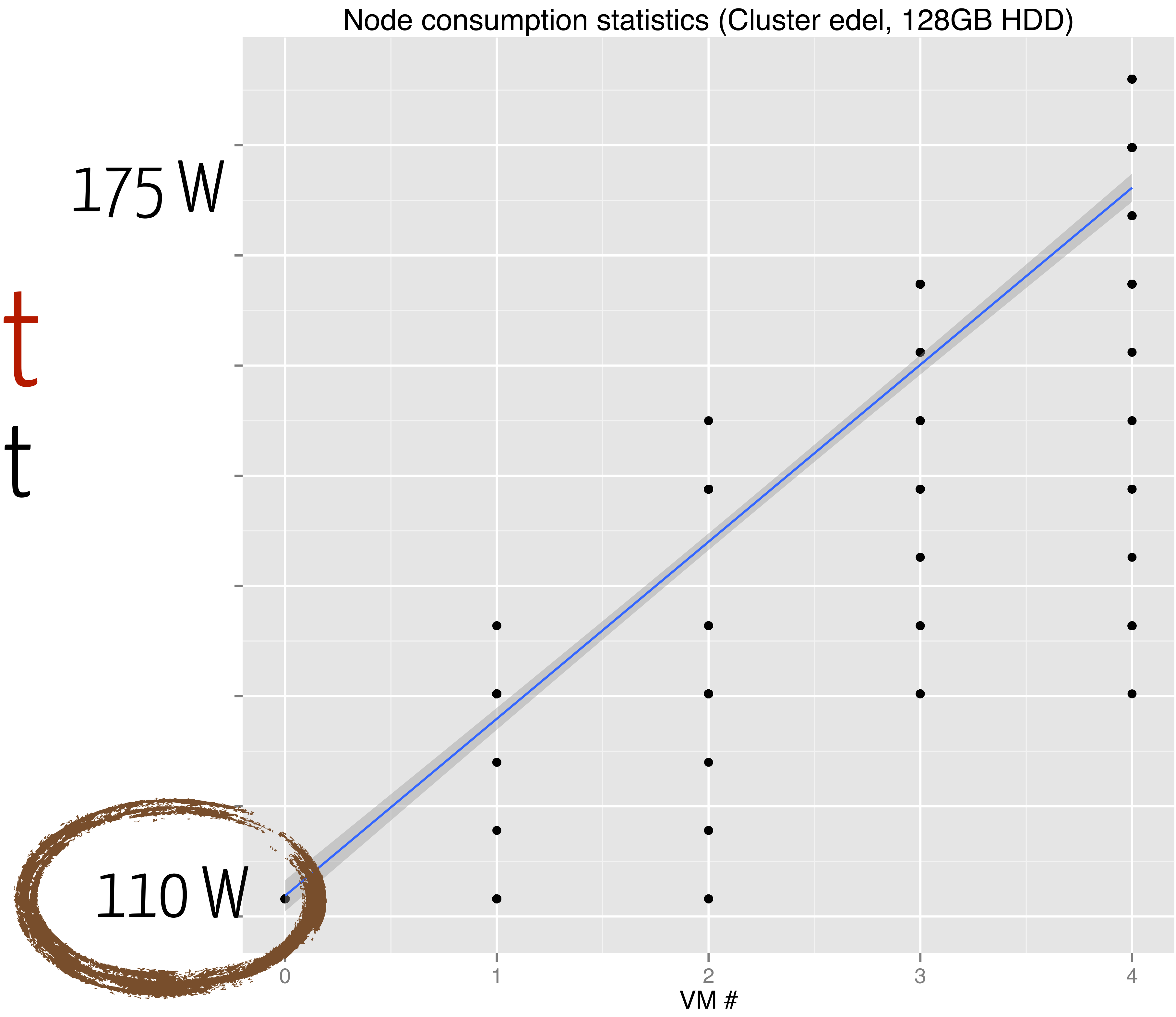
DATACENTER POWER CONSUMPTION

IT EQUIPMENT POWER DRAW

Source: Uptime Institute's 2012 Data Center Survey
J. Koomey's Report, Aug 2011

Source: Samsung, IDC, EMC

servers are not energy efficient



how to reduce consumption with identical nodes

?

The principles

place VMs on the minimum
number of nodes

turn off idle nodes

rince/repeat for
the dynamic version

Best Fit Decrease (BFD)

a simple heuristic to pack VMs

```
sort VMs in desc order
for each VM
 pick the suitable node
     with the least remaining
     space
```

efficiency decreases when nb. of dimensions increase

2-pass dynamic VM packing

to manage the running VMs

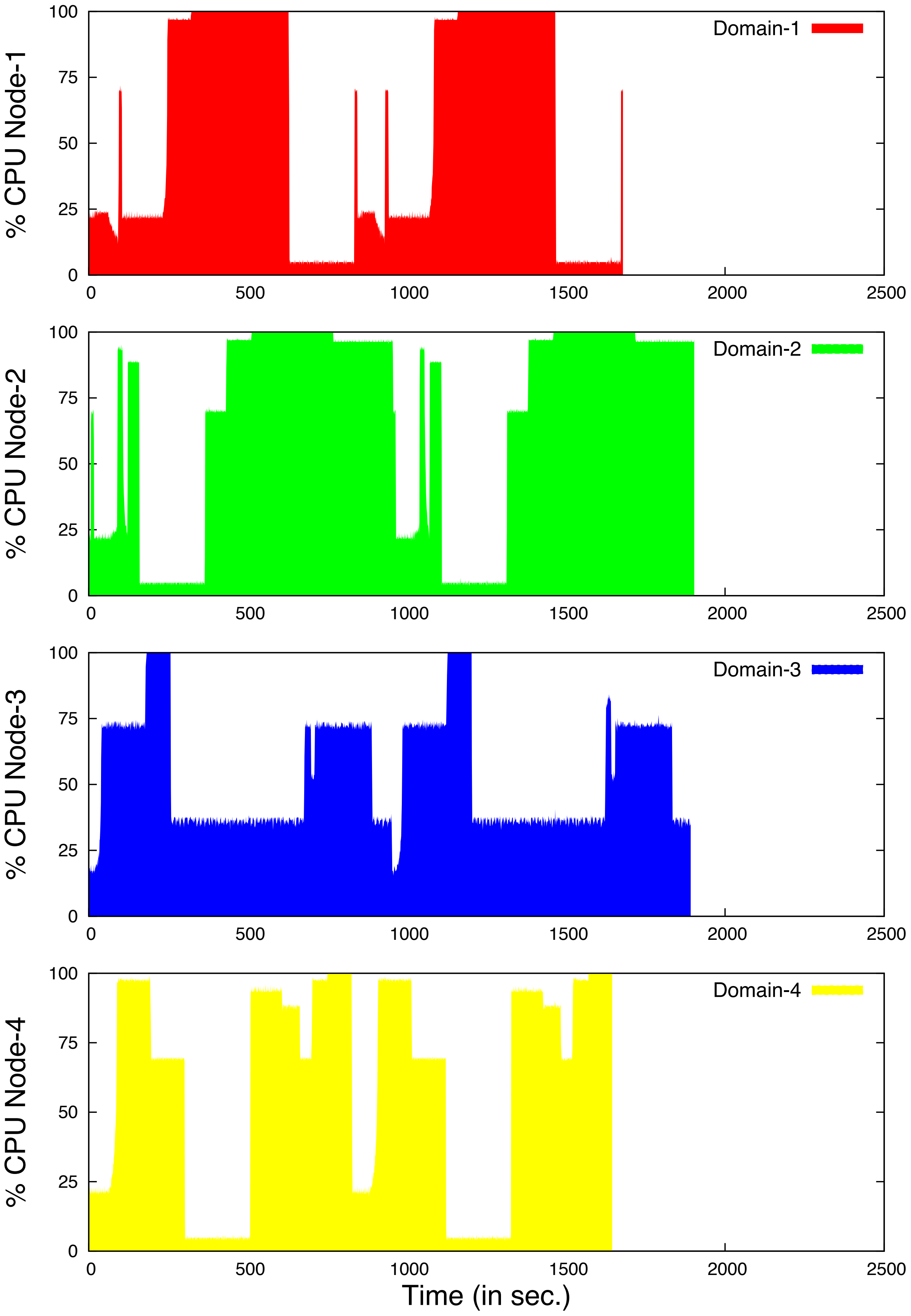
1) address performance issues

on saturated nodes,
BFD to move away VMs on non-saturated nodes

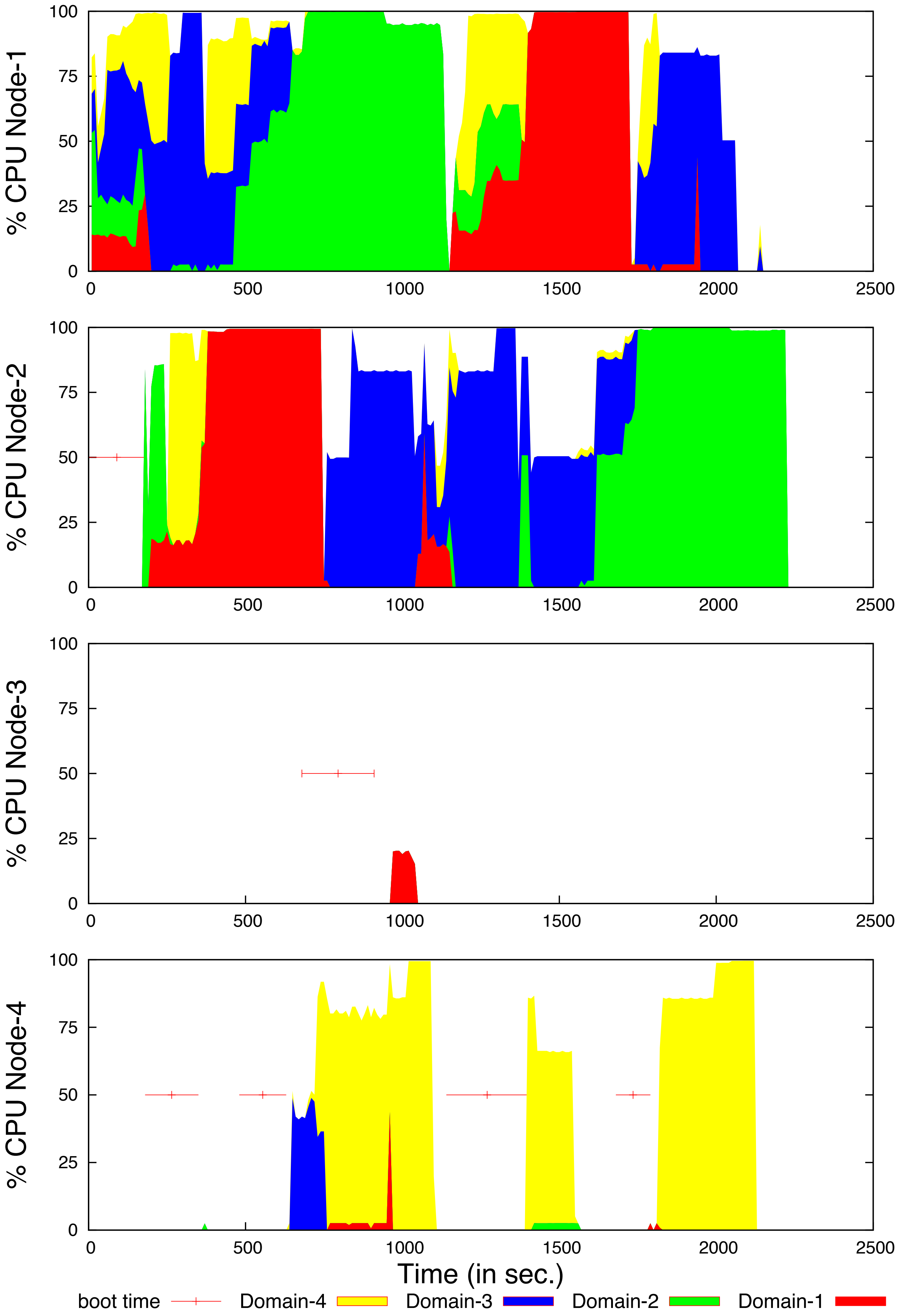
2) address energy efficiency issues

on low-loaded nodes,
BFD to move away VMs on heavily loaded nodes

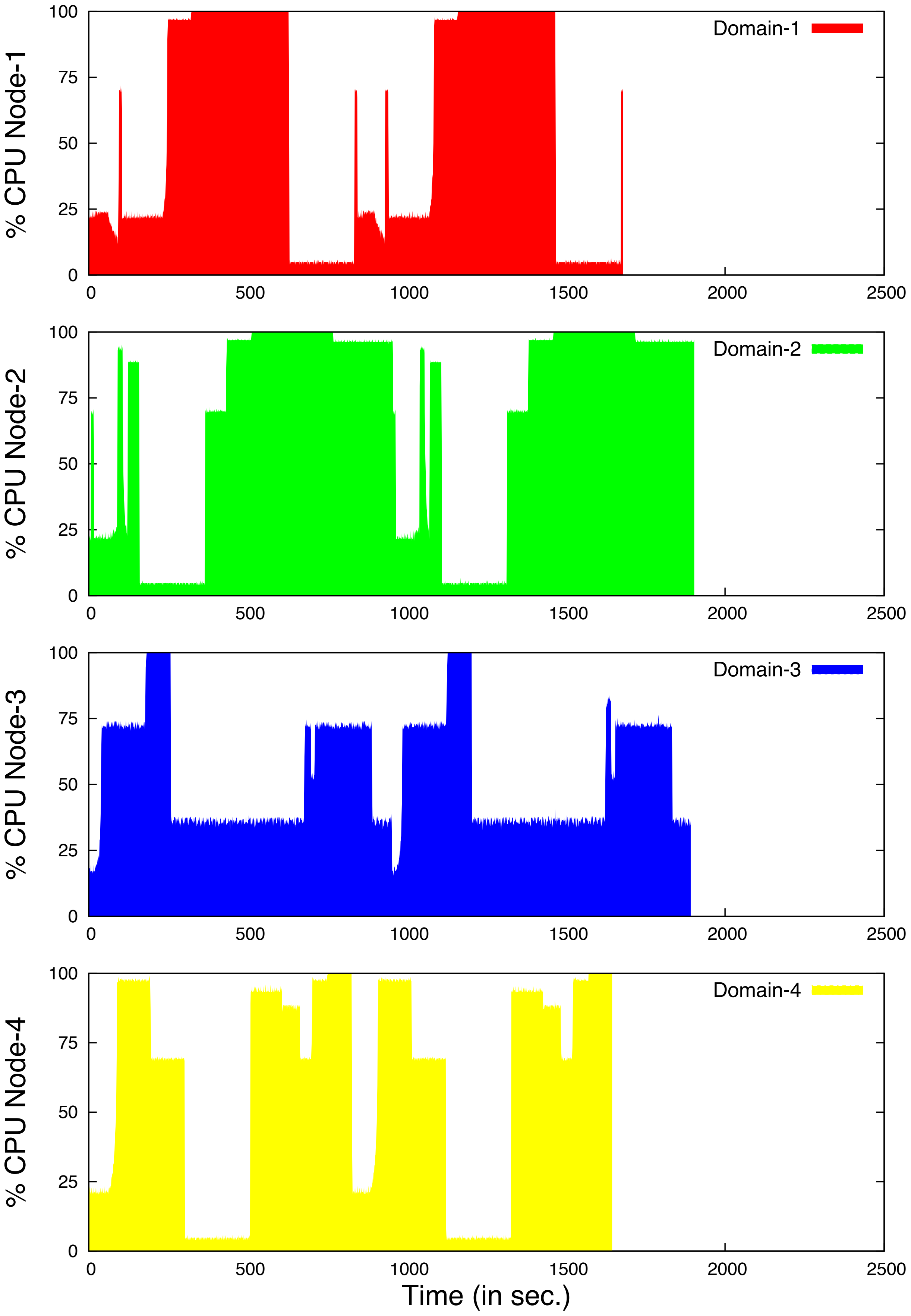
many variations using threshold based systems, load predictions ...



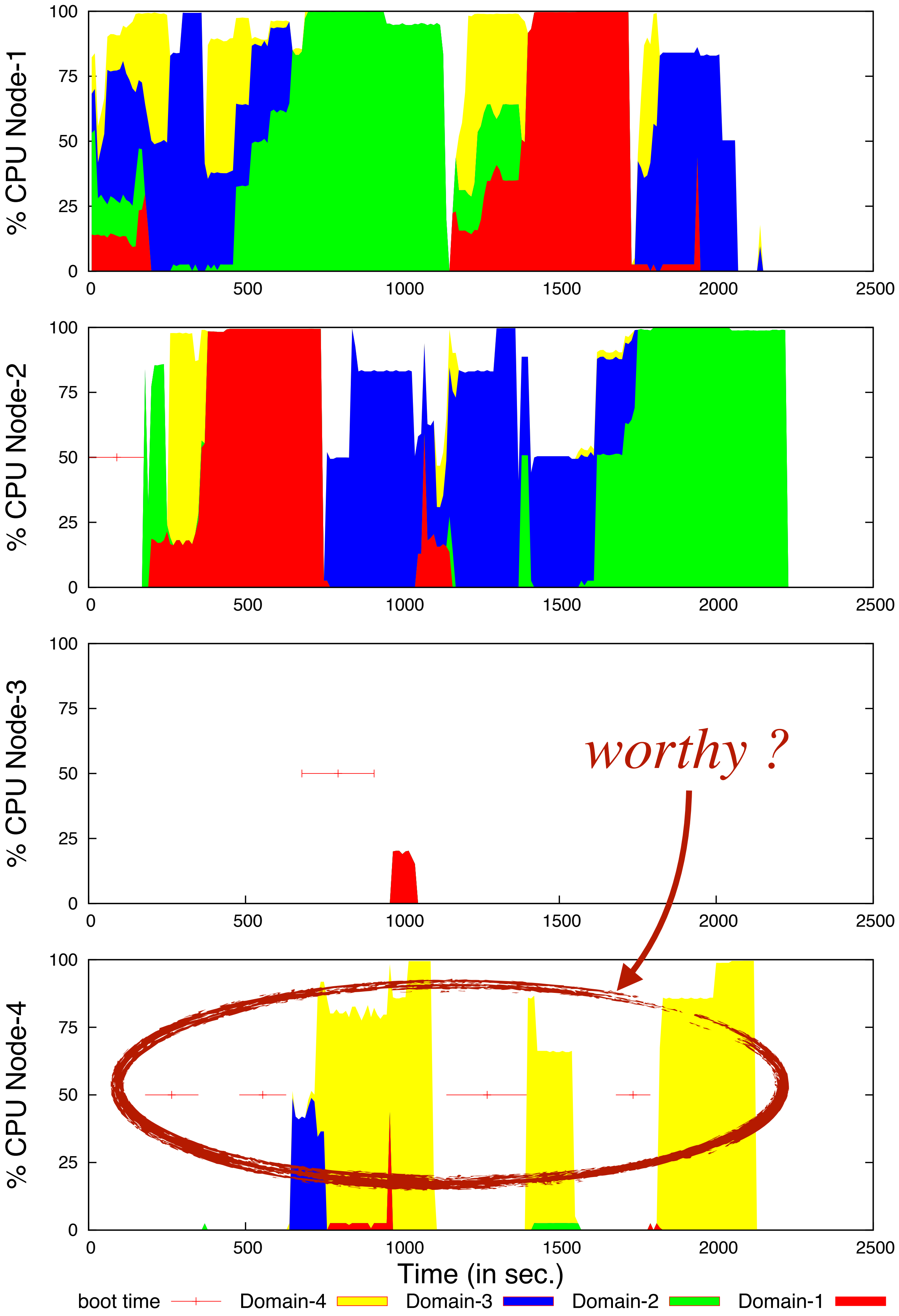
No packing



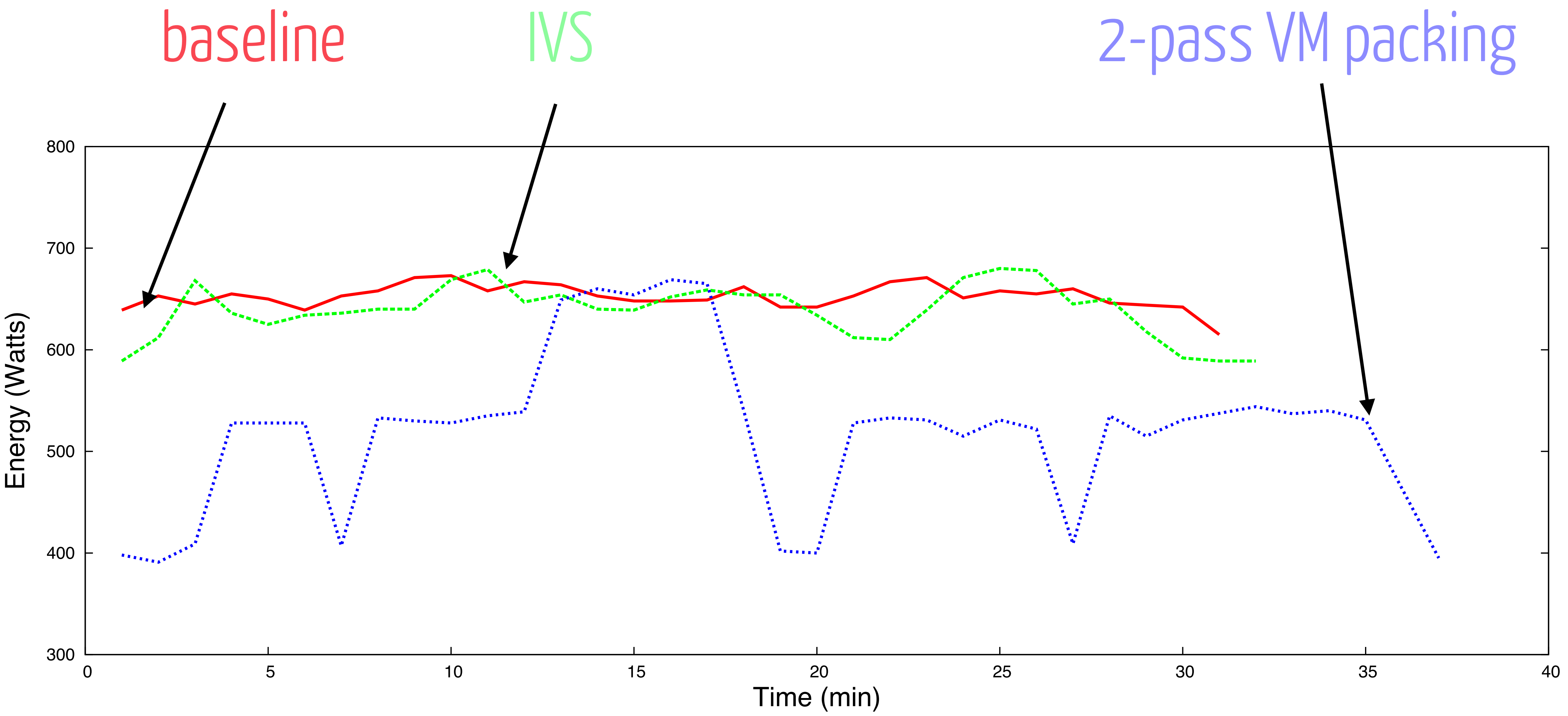
2-passes packing



No packing



2-passes packing



interesting gains,
the node boot time was the bottleneck
simplistic model (identical nodes)

Consequences of having heterogeneous nodes ?

Consequences of having heterogeneous nodes ?

different performance
different Power Usage Effectiveness
...

how to estimate the benefits of a migration ?

vector packing problem

+performance model
+power model
+cost model
+...

---

a specialised model

power models

estimate the energy consumption

model the static and the dynamic energy profile of the components

usually linear equations

$$Wh(node) = \alpha\ Hw + \beta$$

VM performance models

a neutral performance unit
mips, ECU, GCU, ...

a model to map
VM performance with their host

$$\mathrm{perf}(VM, N) = \alpha_n \%\mathrm{cpu} + \beta$$

FIT 4 GREEN

hw. particularities

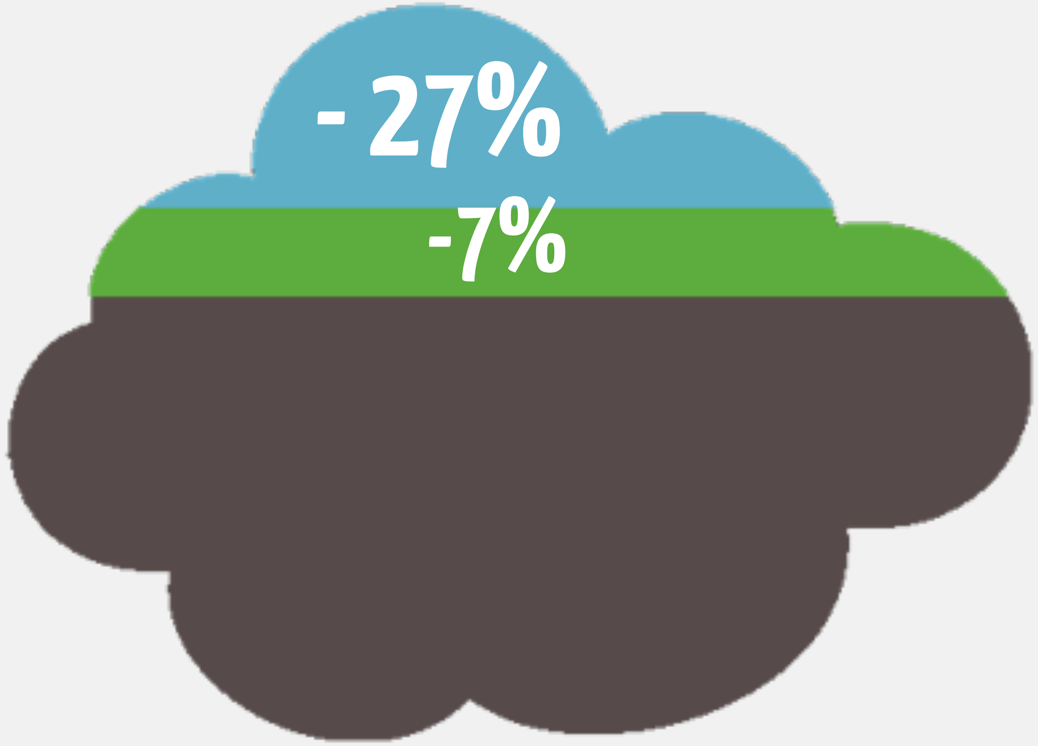
multi-core CPUs,
DDR3 memory,
spinning HD,
PUE / CUE,
boot/shutdown time

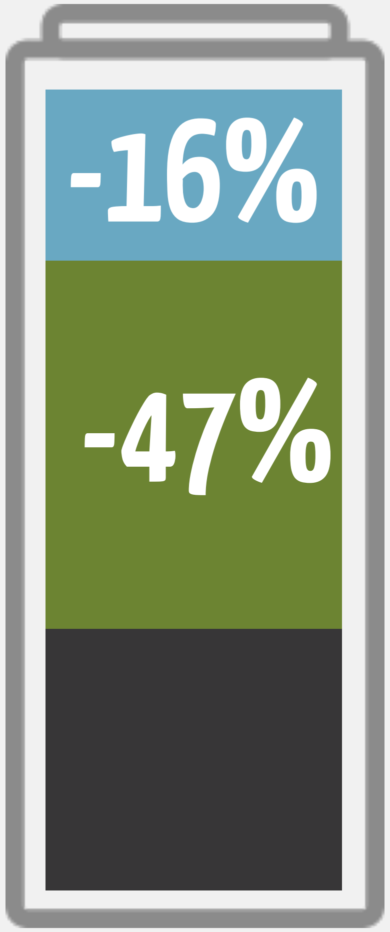
workload particularities

VM template
migration duration
migration payback time

a fine-grain power model

FIT 4 GREEN





coarse to fine
grain optimisation

RECAP

no holy grail

master the problem