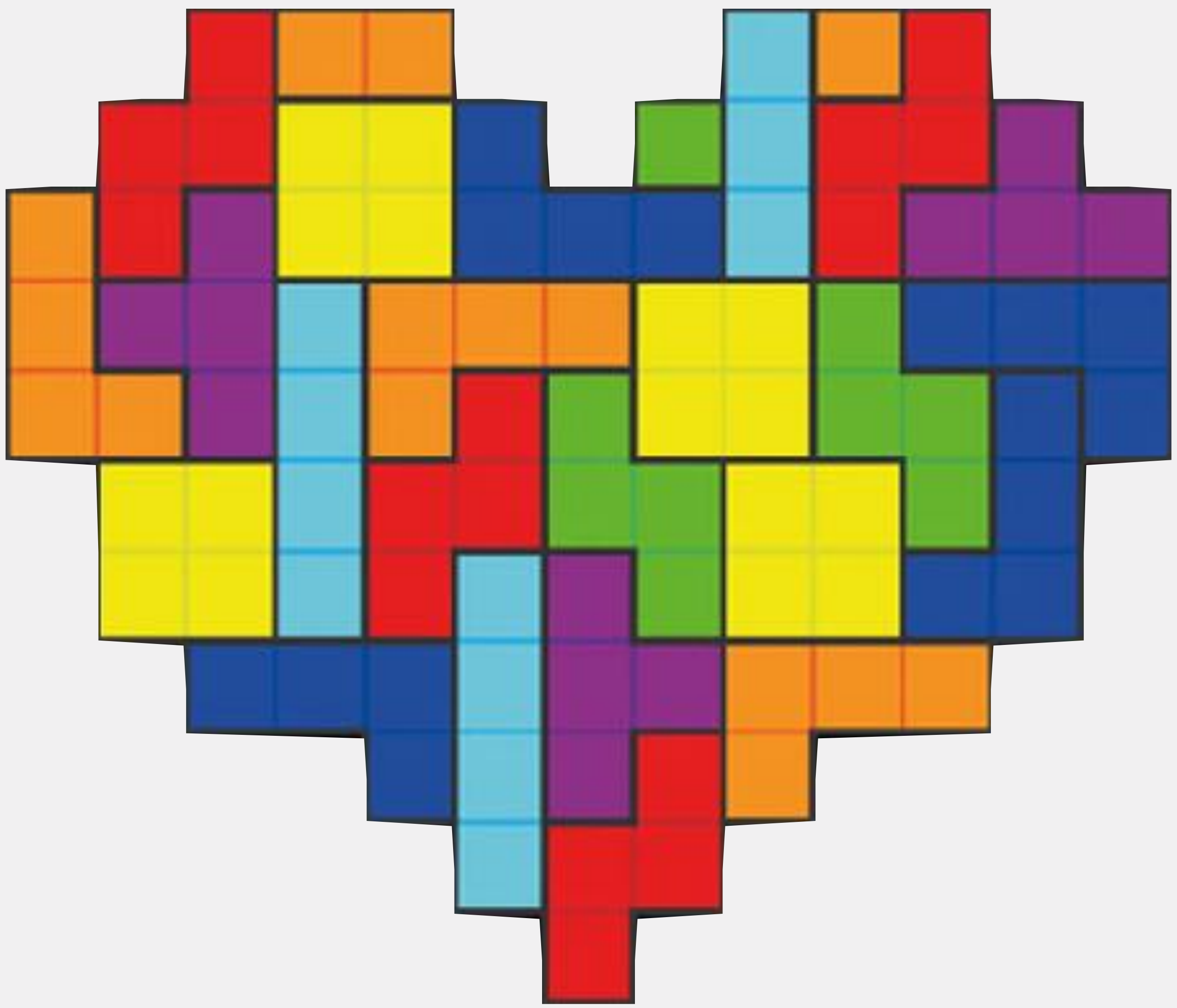
Looking for the perfect VM scheduler



Fabien Hermenier
— placing rectangles since 2006

@fhermeni
fabien.hermenier@nutanix.com
https://fhermeni.github.io

UNIVERSITÉ DE NANTES

2006 - 2010

PhD - Postdoc

Gestion dynamique des tâches dans les grappes,
une approche à base de machines virtuelles

THE UNIVERSITY OF UTAH

2011

Postdoc

How to design a better testbed:
Lessons from a decade of network experiments

Université Nice Sophia Antipolis

2011 - 2016

Associate professor

VM scheduling, green computing

NUTANIX™

Entreprise cloud company

"Going beyond hyperconverged infrastructures"

VM scheduling, resource management
Virtualization

Inside a private cloud

Clusters

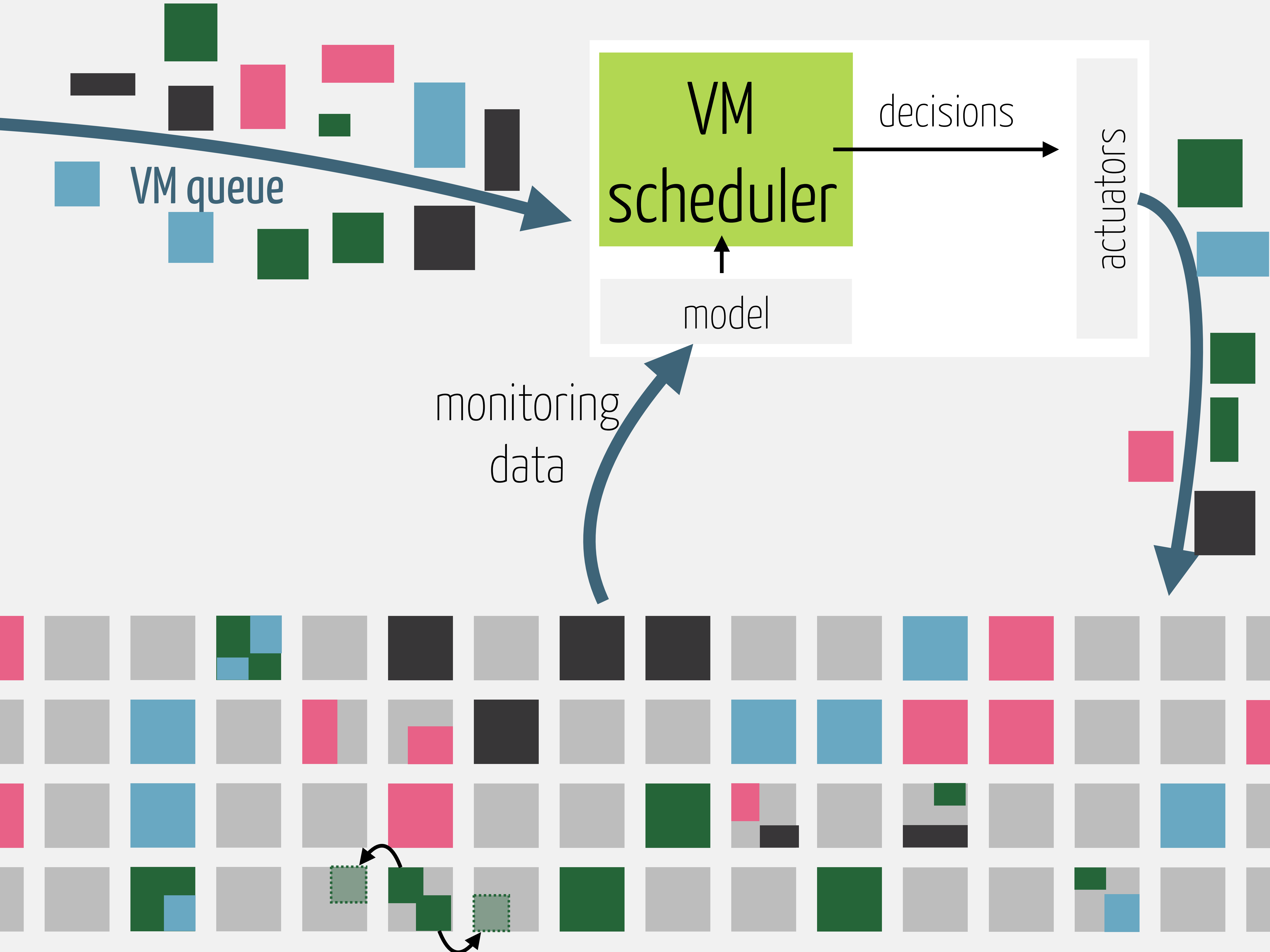
from 2 to *x* physical servers

isolated applications

virtual machines
containers

storage layer

SAN based: converged infrastructure
shared over the nodes: hyper-converged infrastructure



VM queue
VM scheduler
decisions
actuators
model
monitoring data

VM scheduling

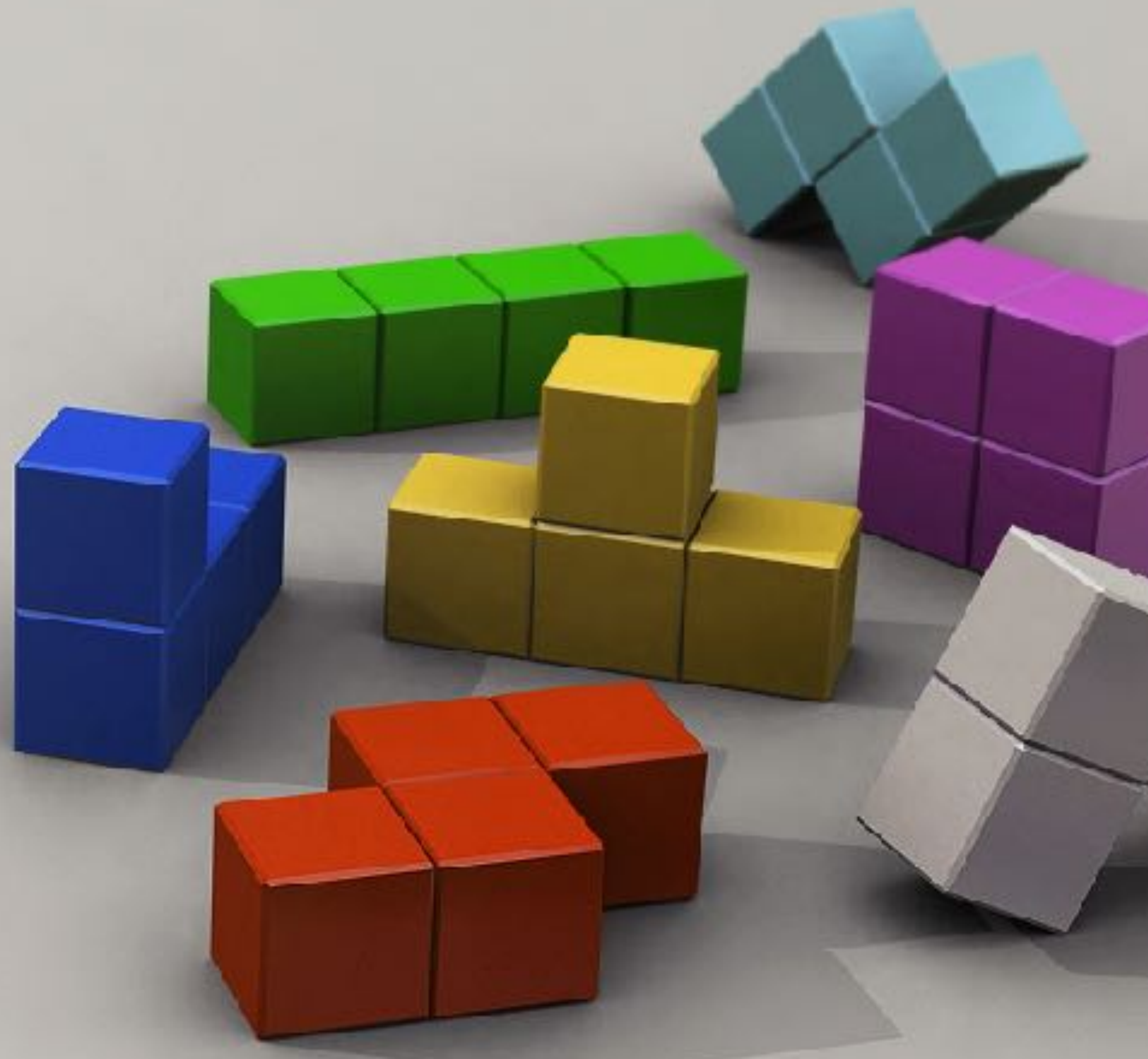
find a server to every VM to run

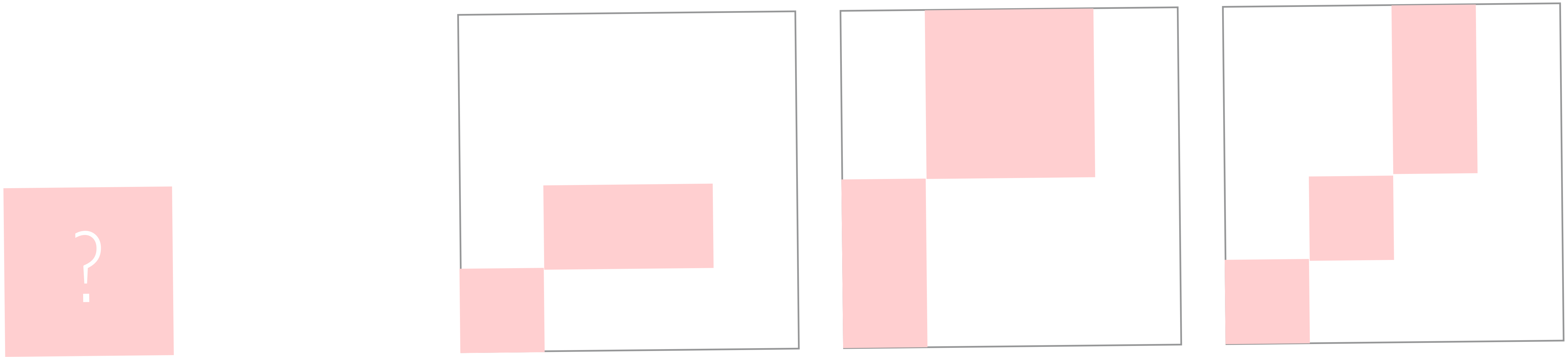
Such that

compatible hw
enough pCPU
enough RAM
enough storage
enough whatever

While

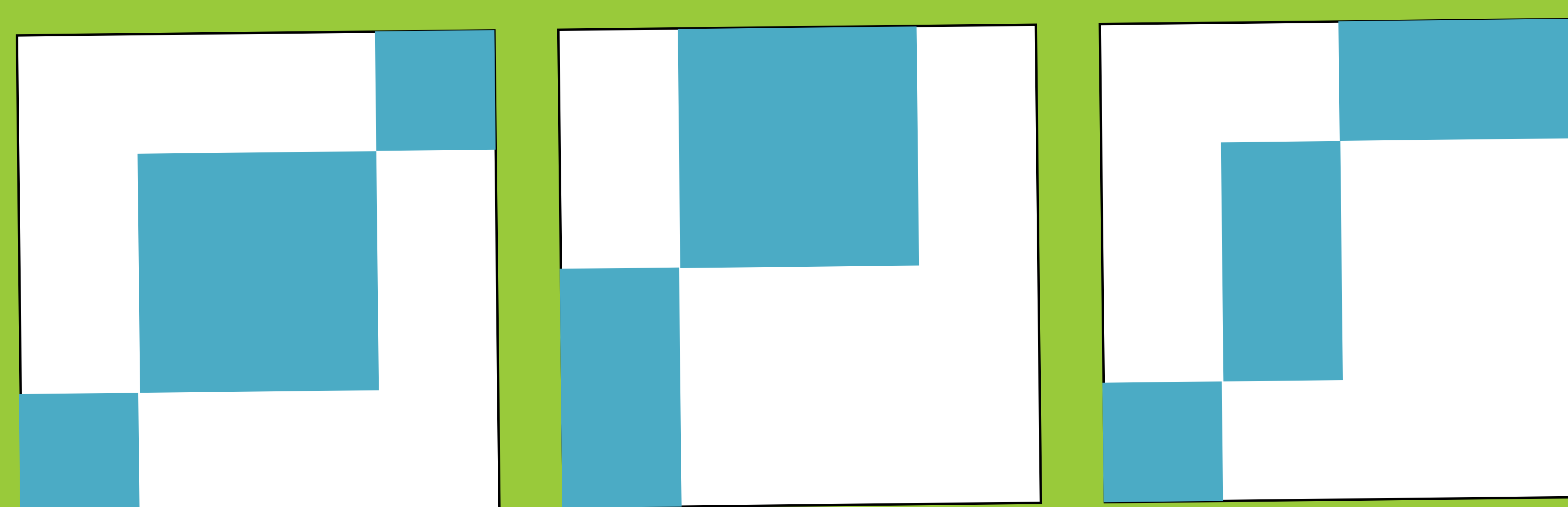
min or max sth

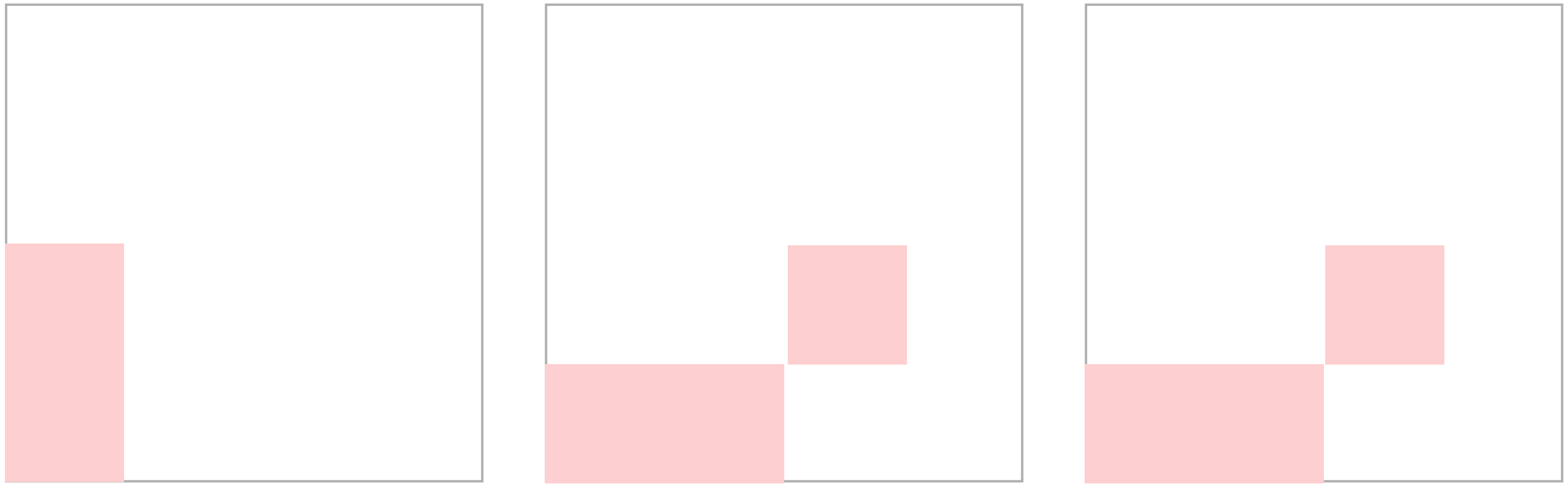




A good VM scheduler provides

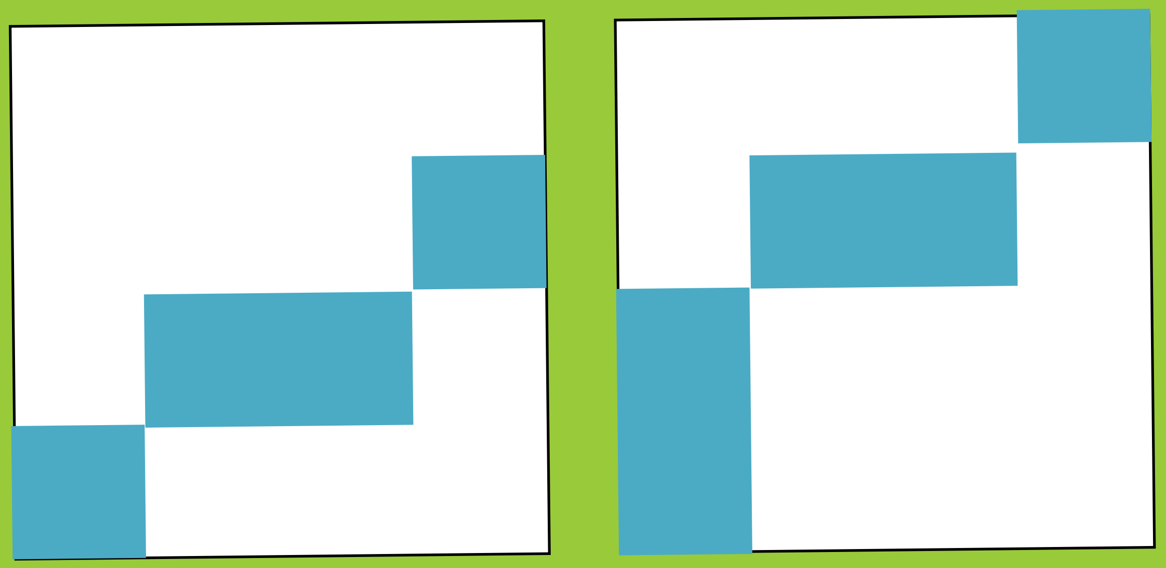
Bigger business value,
same infrastructure





A good VM scheduler provides

Same business value,
smaller infrastructure



KEEP
CALM
AND
CONSOLIDATE
AS HELL

1 node = 

VDI workload:
12+ vCPU/1 pCPU

100+ VMs / server

static schedulers

consider the VM queue

deployed everywhere [1,2,3,4]

fragmentation issues

dynamic schedulers

live-migrations [5] to
address fragmentation

Costly
(storage, migration latency)

thousands of articles [10-13]

over-hyped ? [9]

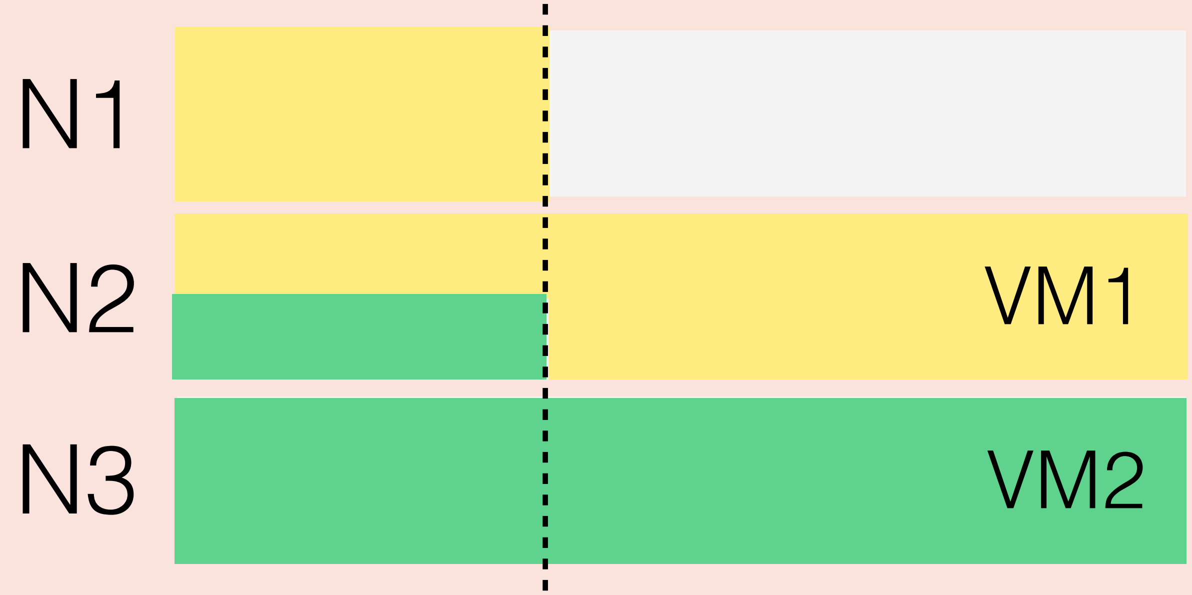
but used in private clouds [6,7,8]
(steady workloads ?)

Placement constraints

various concerns	performance, security, power efficiency, legal agreements, high-availability, fault-tolerance …
dimension	spatial or temporal
enforcement level	hard or soft
manipulated concepts	state, placement, resource allocation, action schedule, counters, etc.

discrete constraints

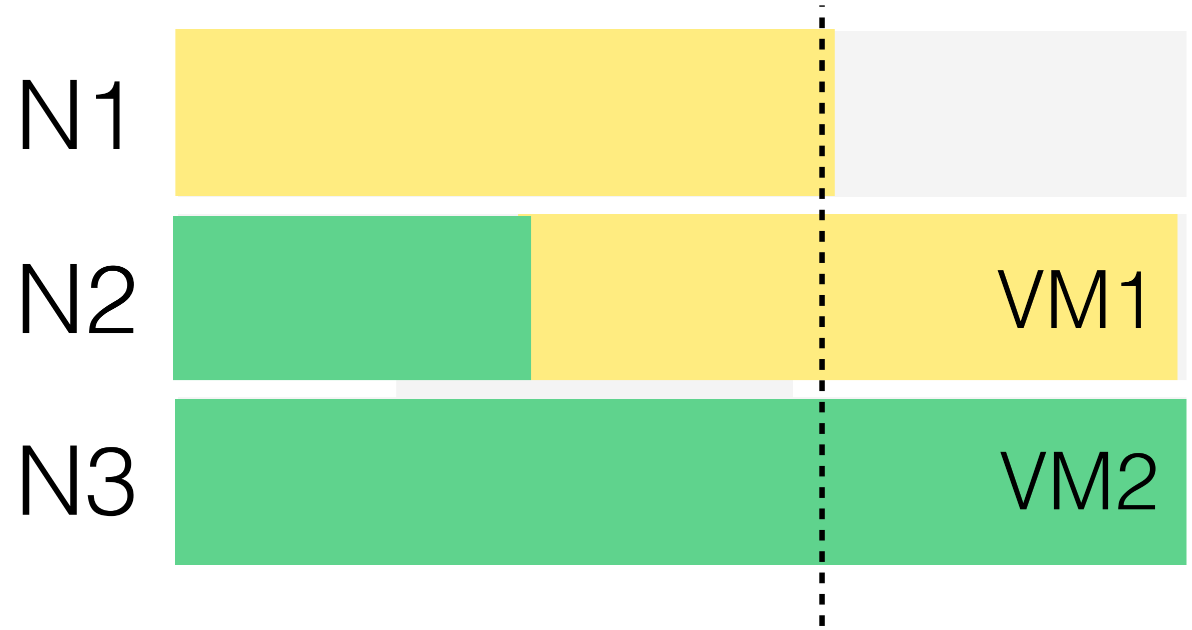
```
>>spread(VM[1,2])
ban(VM1, N1)
ban(VM2, N2)
```



"simple" spatial problem

continuous constraints [15]

```
spread(VM[1,2])
ban(VM1, N1)
ban(VM2, N2)
```



harder scheduling problem
(think about actions interleaving)

hard constraints

`spread(VM[1..50])`

must be satisfied
all or nothing approach
not always meaningful

`mostlySpread(VM[1..50], 4, 6)`

soft constraints [6]

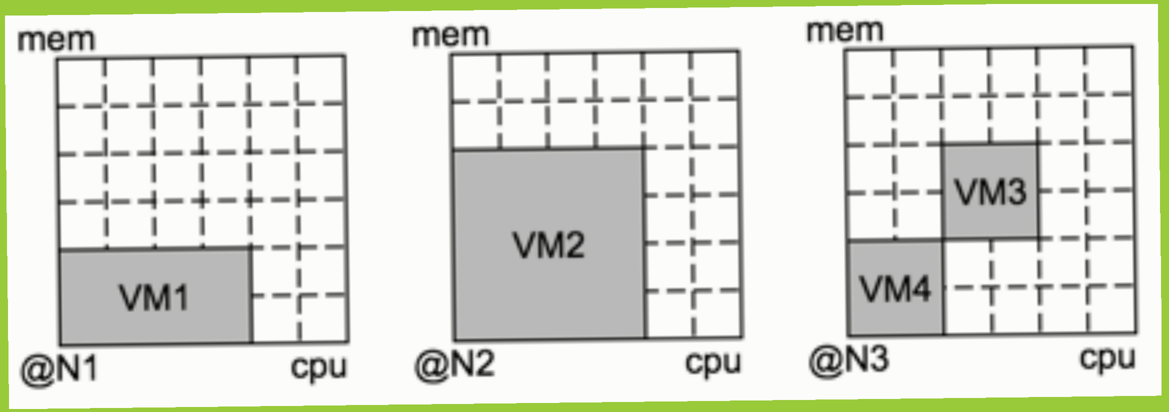
satisfiable or not
internal or external penalty model

harder to implement/scale
hard to standardise ?

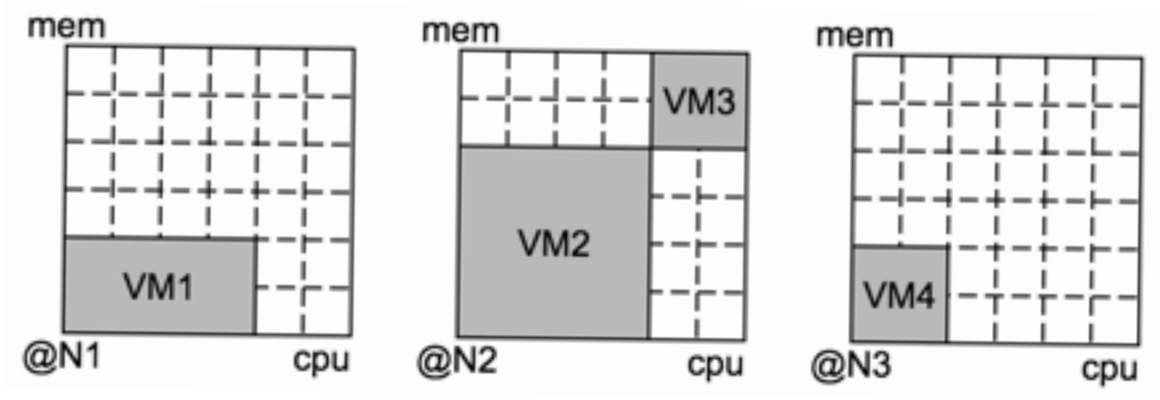
High-availability

x-FT VMs must survive to any crash of x nodes

1 - FT

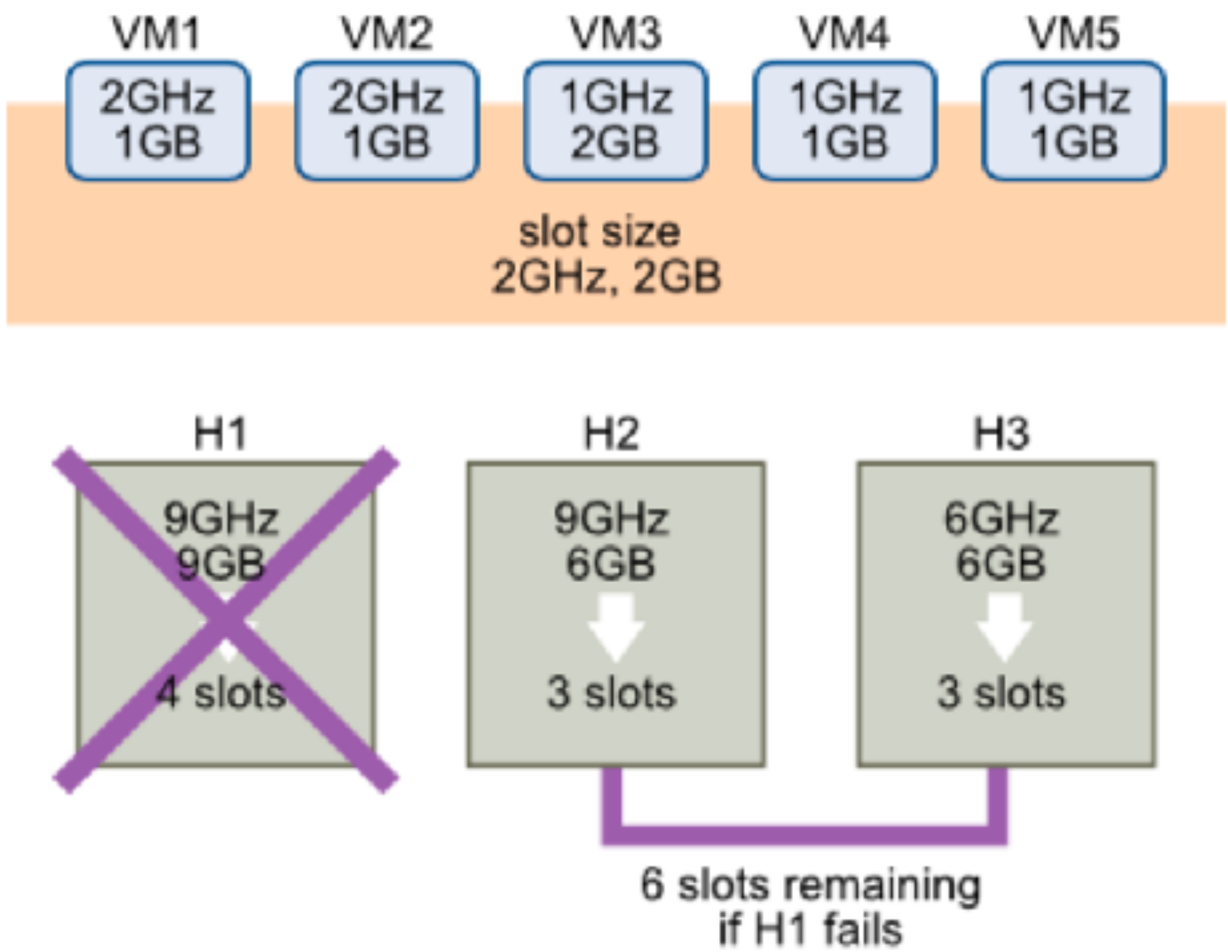


0 - FT



exact approach: solve n^x placement problems [17]

The VMWare DRS way



slot based

catch the x- biggest nodes

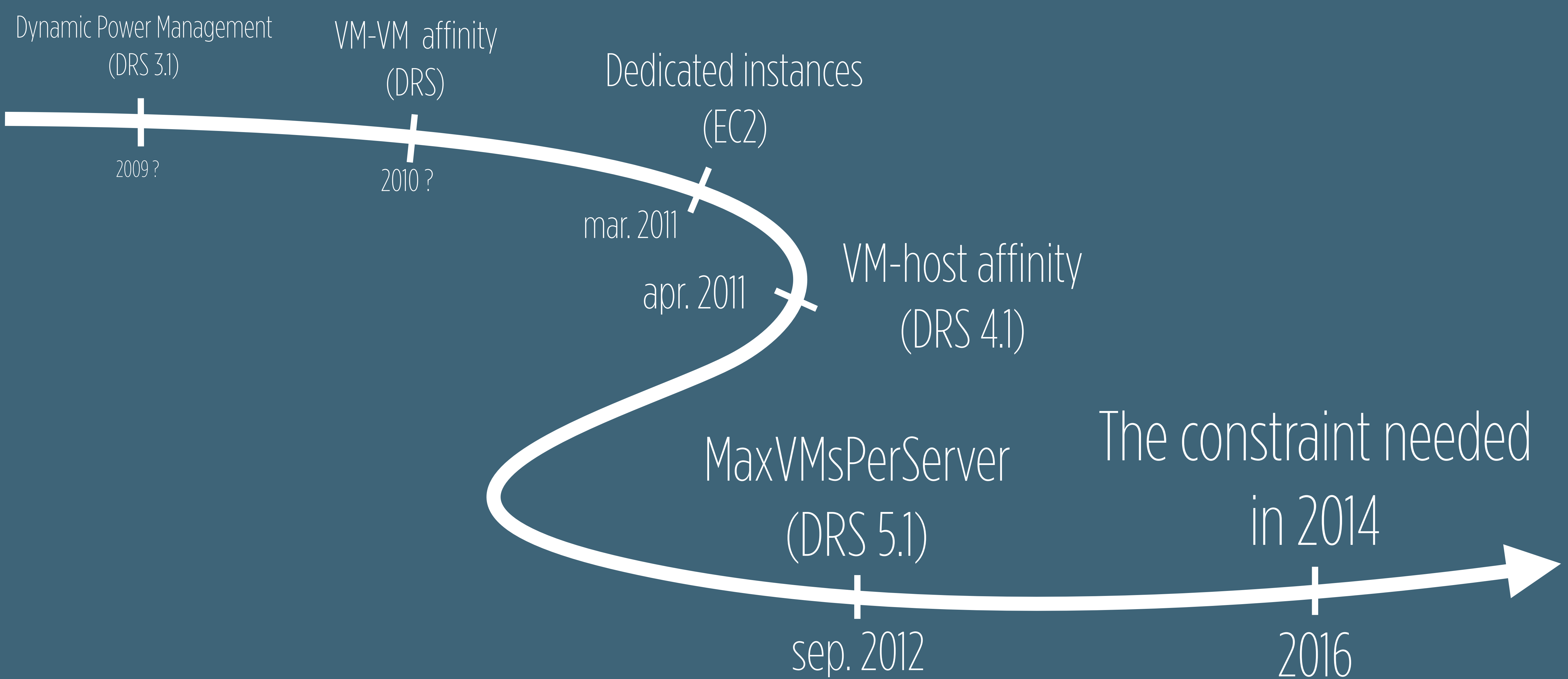
checks the remaining free slots

simple, scalable

waste with heterogeneous VMs

cluster based

The constraint catalog evolves



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 ?

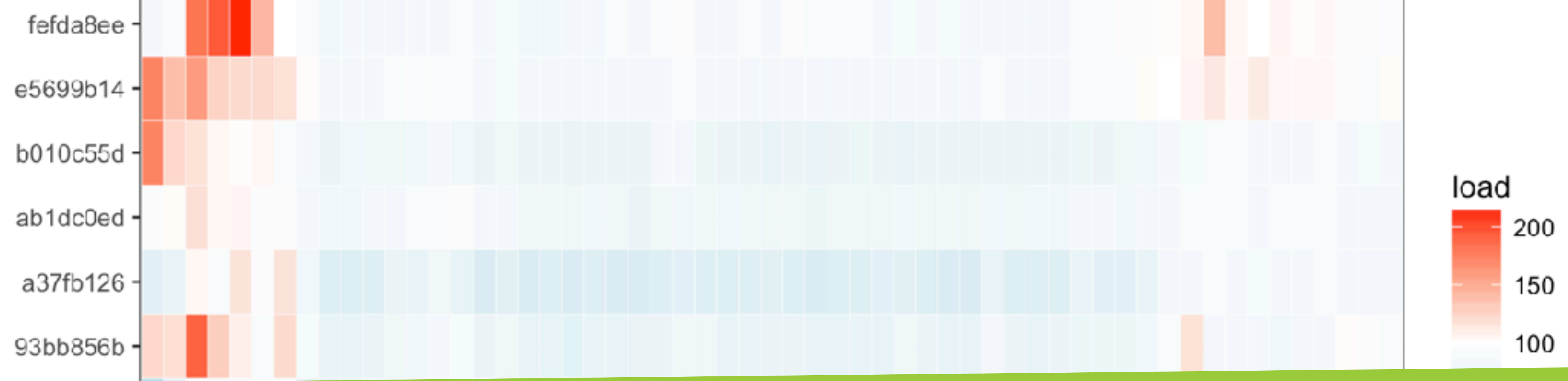
$$min(\alpha\ TCO + \beta\ VIOLATIONS)$$

€ as a common quantifier:

$$max(REVENUES)$$

Optimize or satisfy ?

min(...) or max(...)	threshold based
easy to say	domain specific expertise
hardly provable	verifiable
composable through weighting magic	composable



Acropolis Dynamic Scheduler [18]

Hotspot mitigation

Trigger

Thresholds

85%

CPU

storage-CPU

Maintain

affinity constraints

Resource demand

(from machine learning)

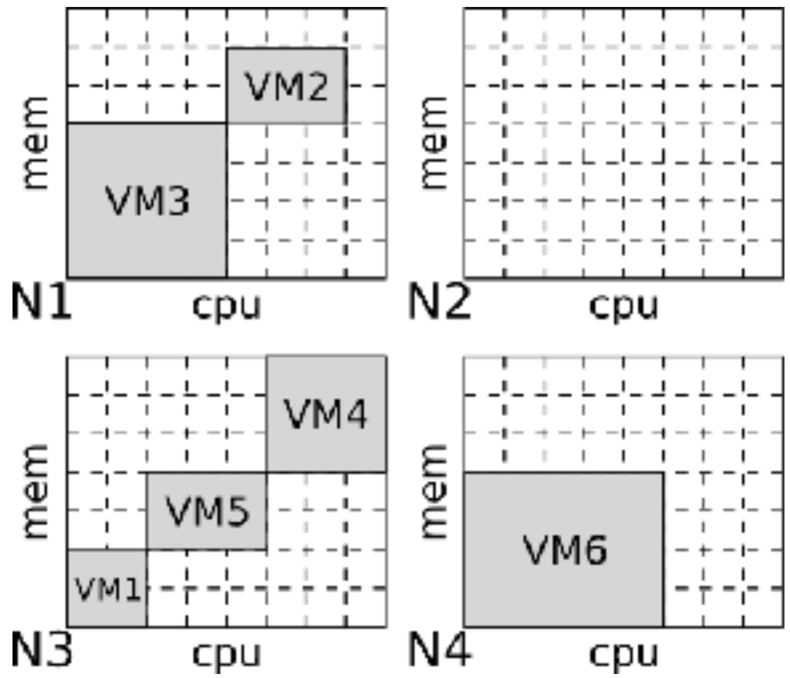
Minimize

$\sum$ mig. cost

BtrPlace

adapt the VM placement depending on pluggable expectations

network and memory-aware migration scheduler, VM-(VM|PM) affinities, resource matchmaking, node state manipulation, counter based restrictions, energy efficiency, discrete or continuous restrictions interaction though a DSL, an API or JSON messages

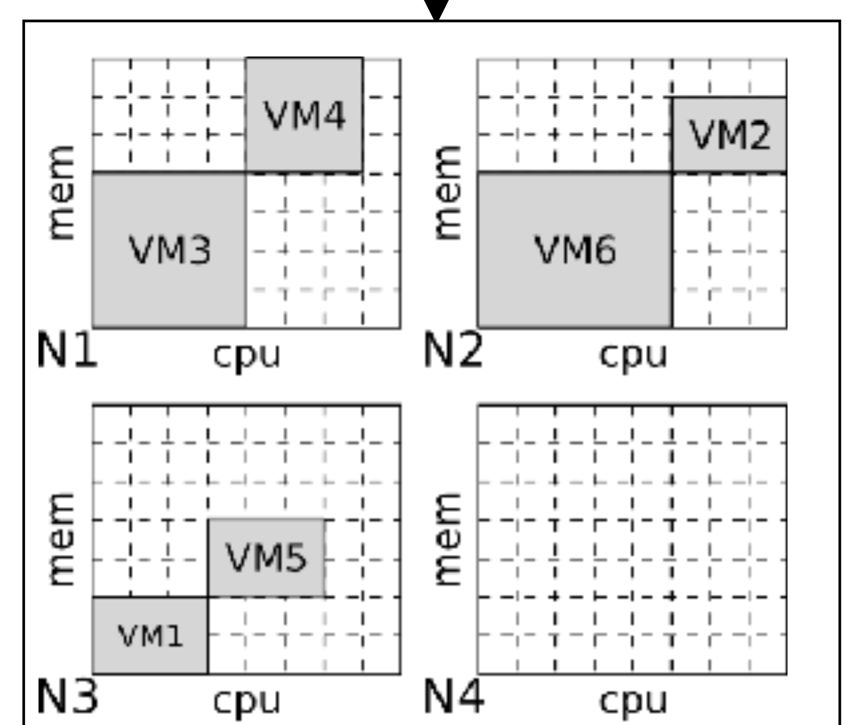



```
spread(VM[2..3]);
preserve(VM1,'cpu', 3);
offline(@N4);
```

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)
```



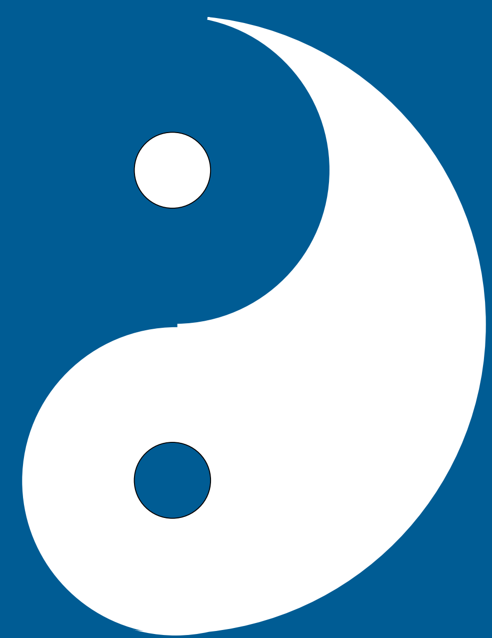
CHOCO

An Open-Source java library
for constraint programming

$$
\begin{aligned}
\mathcal{X} &= \{x_1, x_2, x_3\} \\
\mathcal{D}(x_i) &= [0,2]\,, \forall x_i \in \mathcal{X} \\
\mathcal{C} &= \begin{cases} c_1 : x_1 < x_2 \\ c_2 : x_1 + x_2 \geq 2 \\ c_3 : x_1 < x_3 \end{cases}
\end{aligned}
$$

deterministic composition

high-level constraints



the right model

for the right problem

BtrPlace core CSP

models a reconfiguration plan

1 model of transition per element

action durations as constants *

$$
\begin{aligned}
boot(v \in V) \triangleq \quad & D(v) \in \mathbb{N} \\
& st(v) = [0, H - D(v)] \\
& ed(v) = st(v) + D(v) \\
& d(v) = ed(v) - st(v) \\
& d(v) = D(v) \\
& ed(v) < H \\
& d(v) < H \\
& h(v) \in \{0, .., |N| - 1\}
\end{aligned}
$$

$$relocatable(v \in V) \triangleq \ldots$$

$$shutdown(v \in V) \triangleq \ldots$$

$$suspend(v \in V) \triangleq \ldots$$

$$resume(v \in V) \triangleq \ldots$$

$$kill(v \in V) \triangleq \ldots$$

$$bootable(n \in N) \triangleq \ldots$$

$$haltable(n \in N) \triangleq \ldots$$

Views bring additional concerns

new variables and relations

ShareableResource(r) ::=

$$\forall n \in \mathcal{N}, \sum_{v \in \mathcal{V}, host(v)=n} cons(v,r) \leq capa(n,r)$$

Network() ::= …

Power() ::= …

High-Availability() ::= …

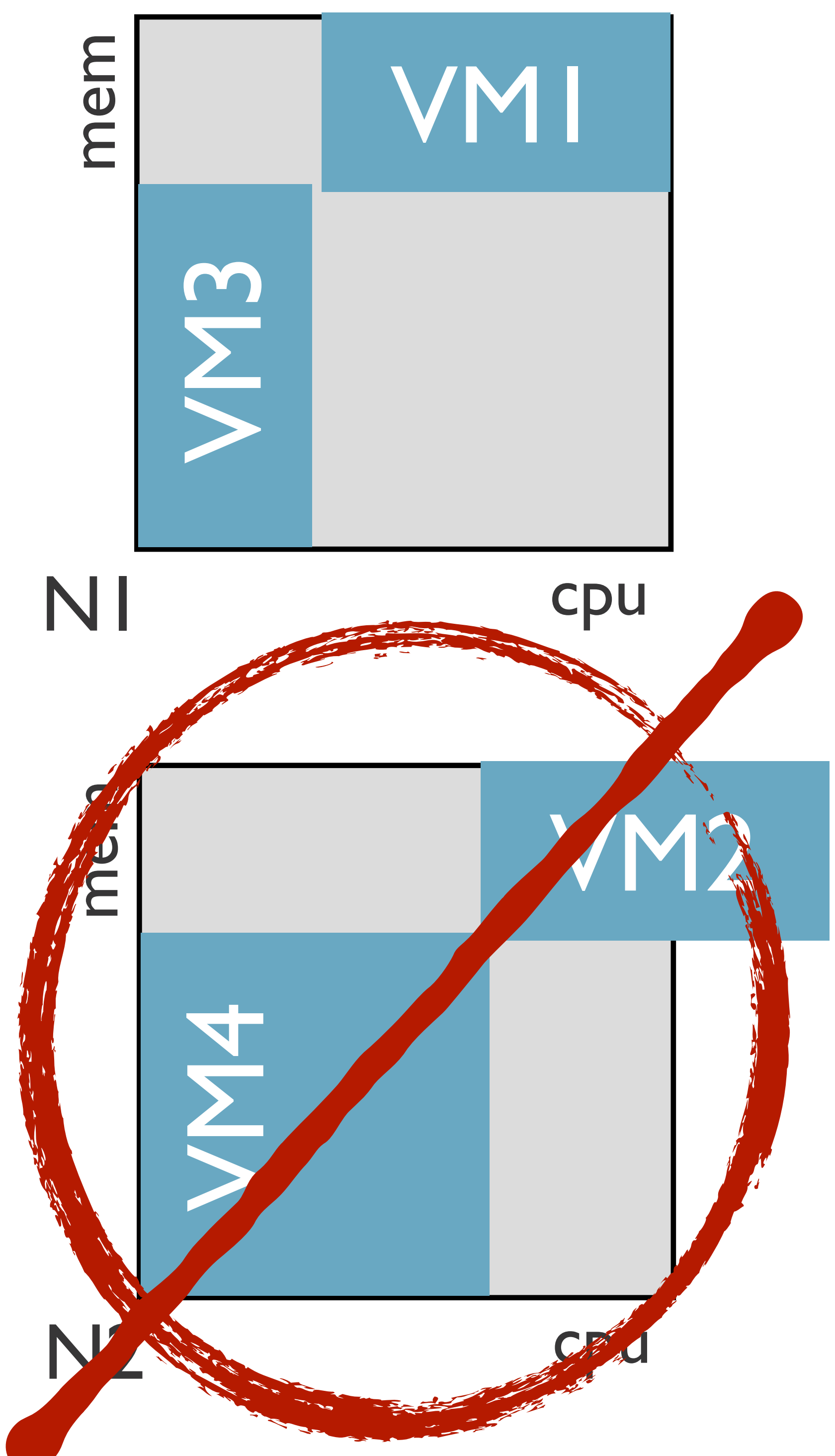
Constraints state new relations

$$spread(X \subseteq \mathcal{V}) \triangleq \forall(a,b) \in X, host(a) \neq host(b)$$

$$lonely(X \subseteq \mathcal{V}) \triangleq \bigcup_{v \in X} host(v) \not\subseteq \bigcup_{v \in \mathcal{V} \setminus X} host(v)$$

...

vector packing problem



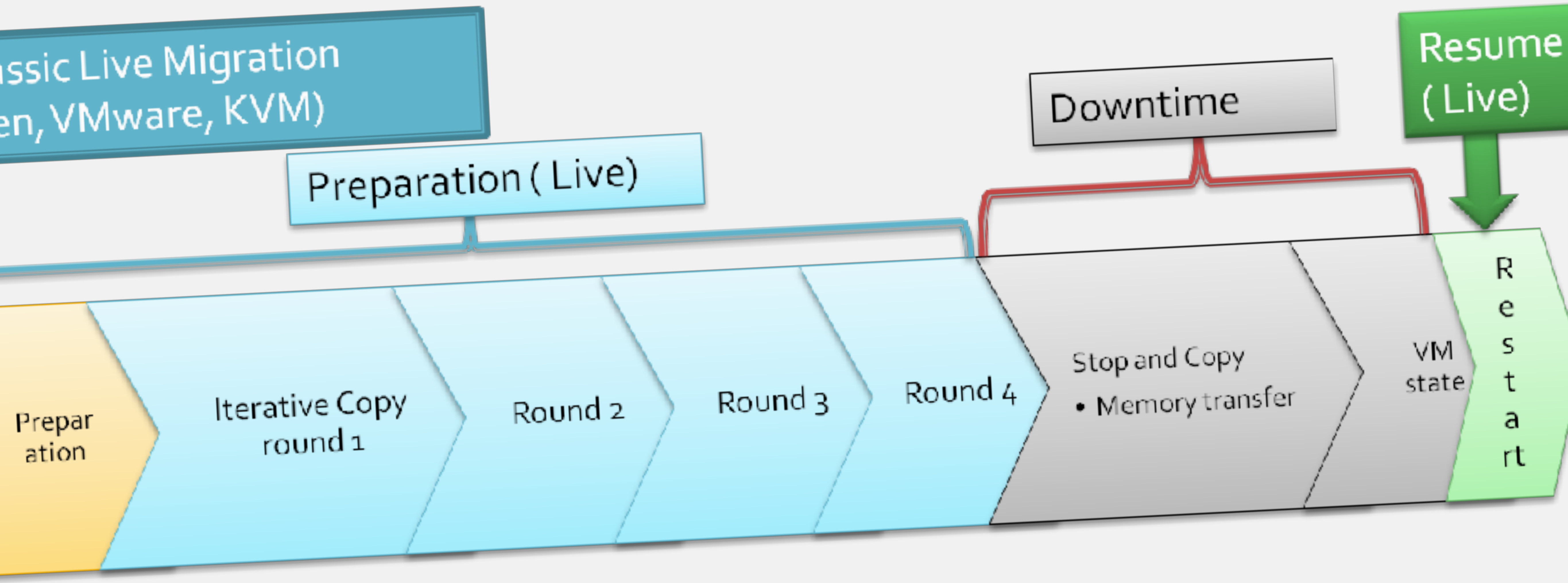
items with a finite volume
to place inside finite bins

generalisation of
the bin packing problem

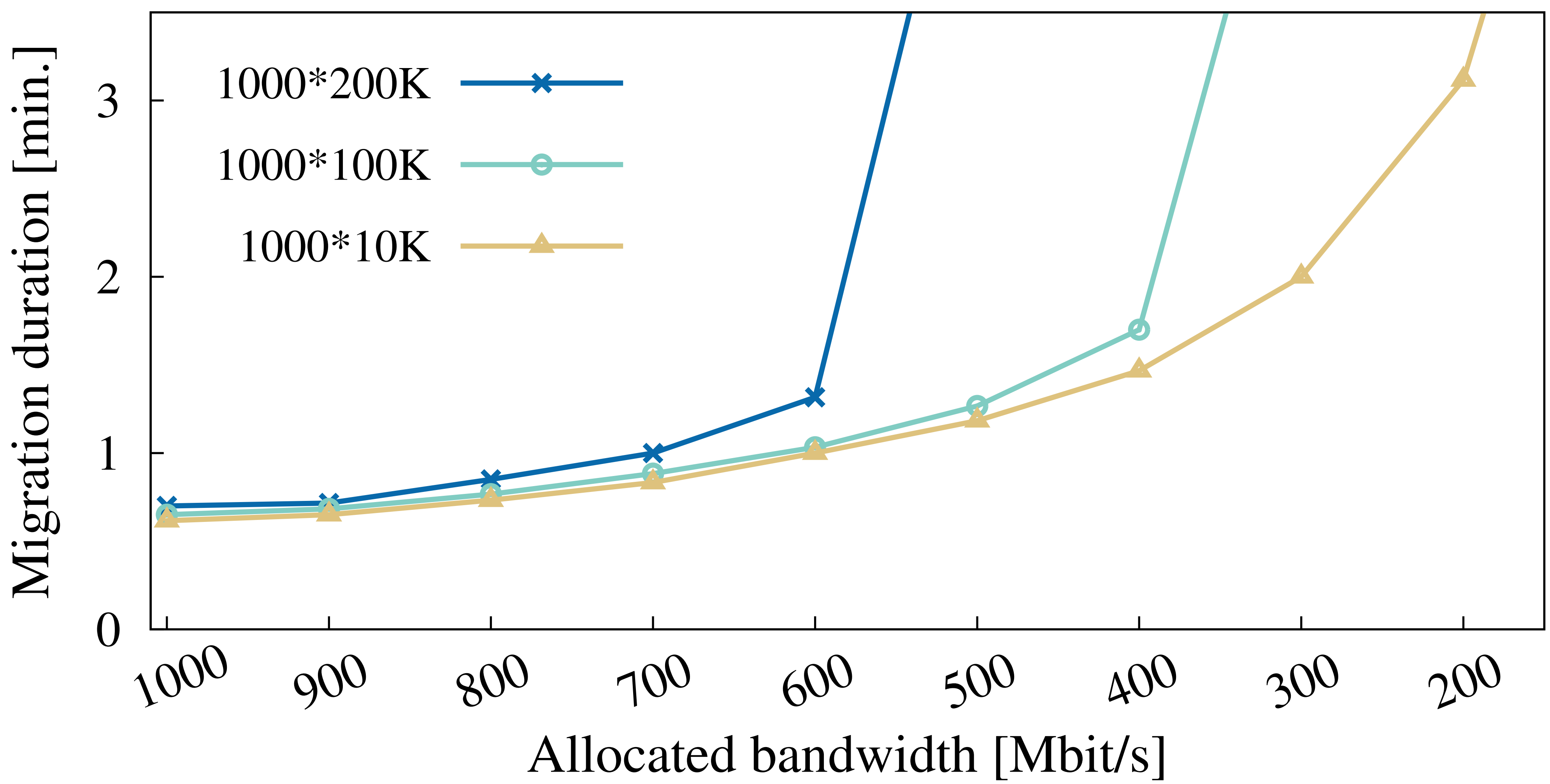
the basic to model the infra.
1 dimension = 1 resource

NP-hard problem

how to support migrations



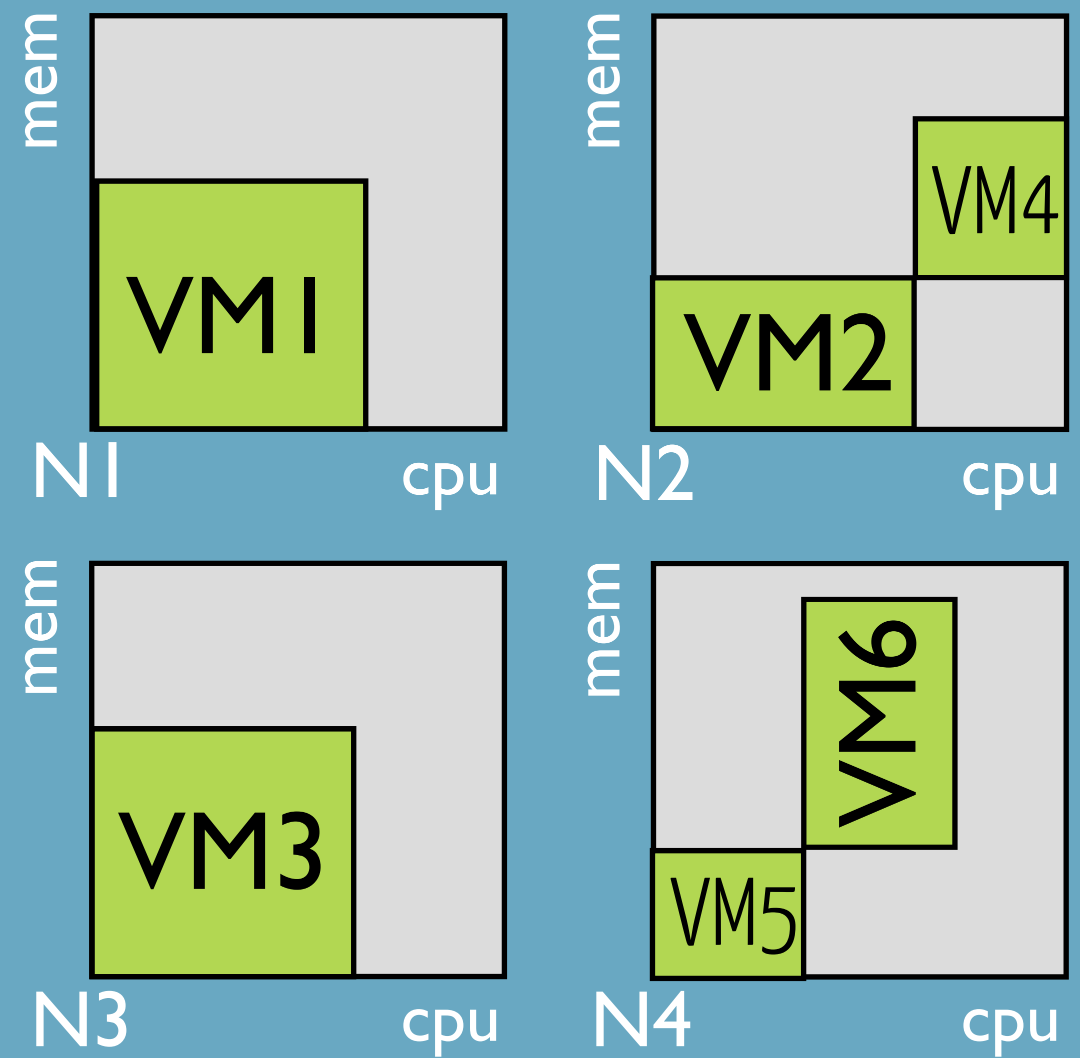
temporary,
resources are used on the source and
the destination nodes



Migrations are costly

dynamic schedulers

Using Vector packing [10,12]

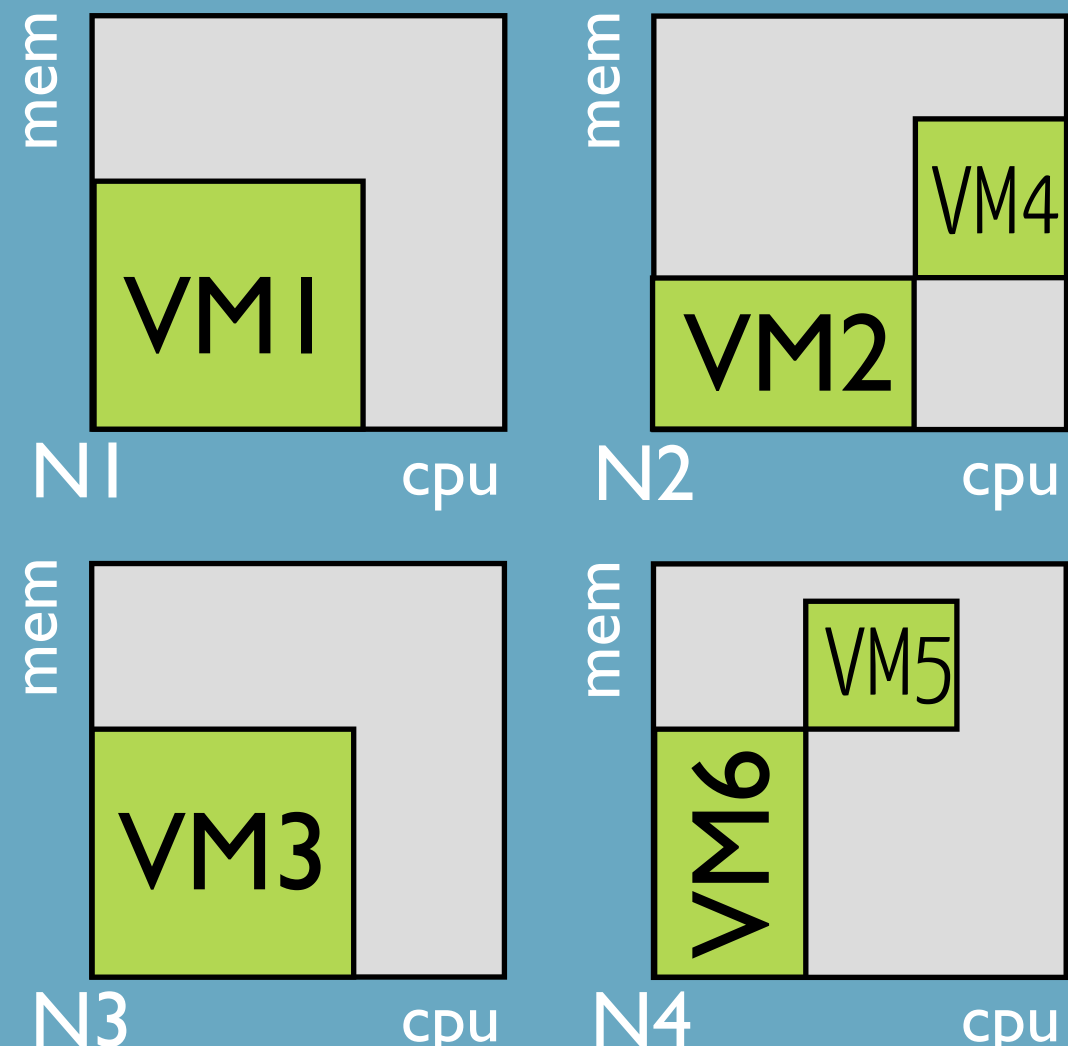


min(#onlineNodes) = 3

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

dynamic schedulers

Using Vector packing [10,12]



min(#onlineNodes) = 3

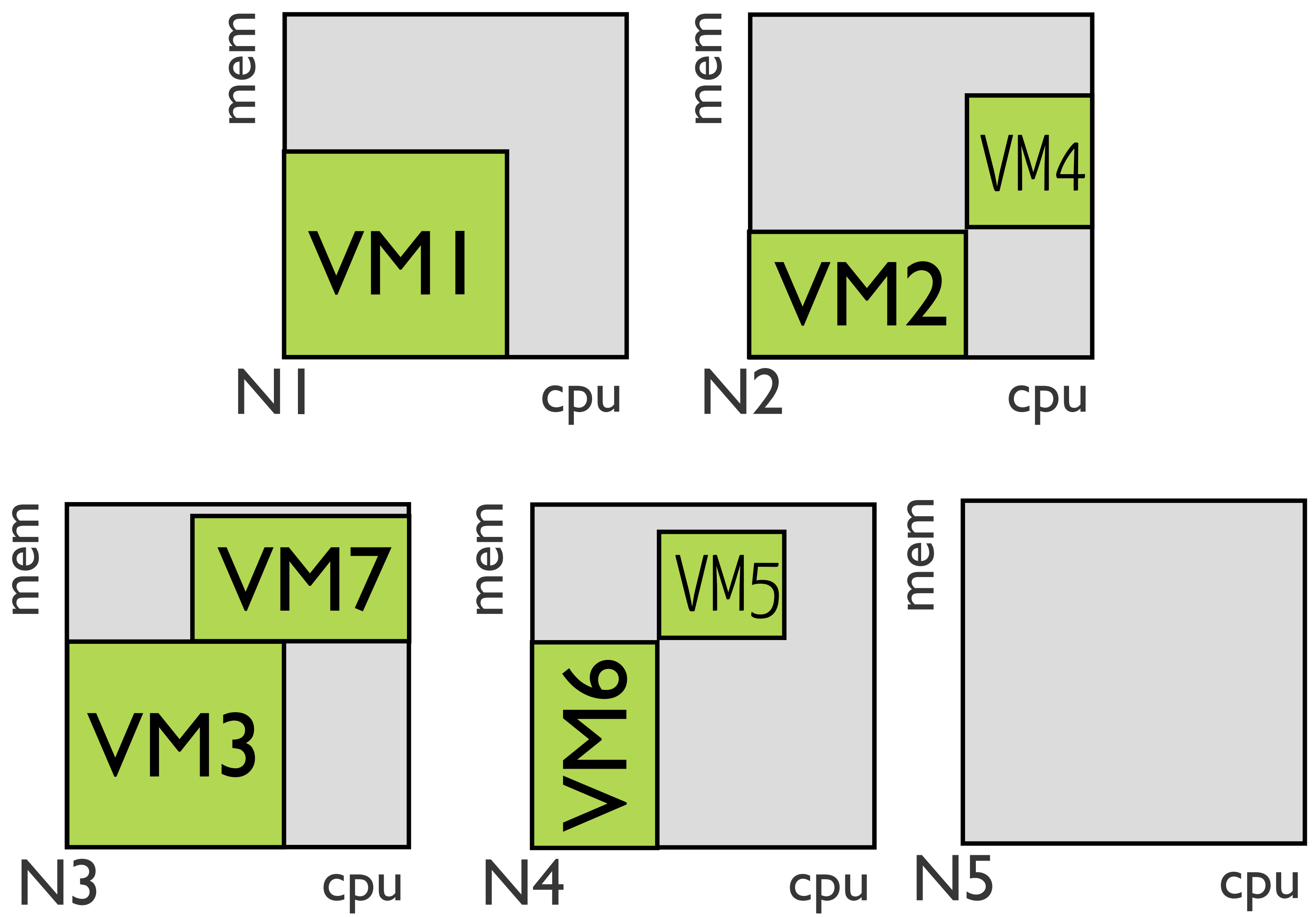
sol #1: 1m,1m,2m

sol #2: 1m,2m
1m

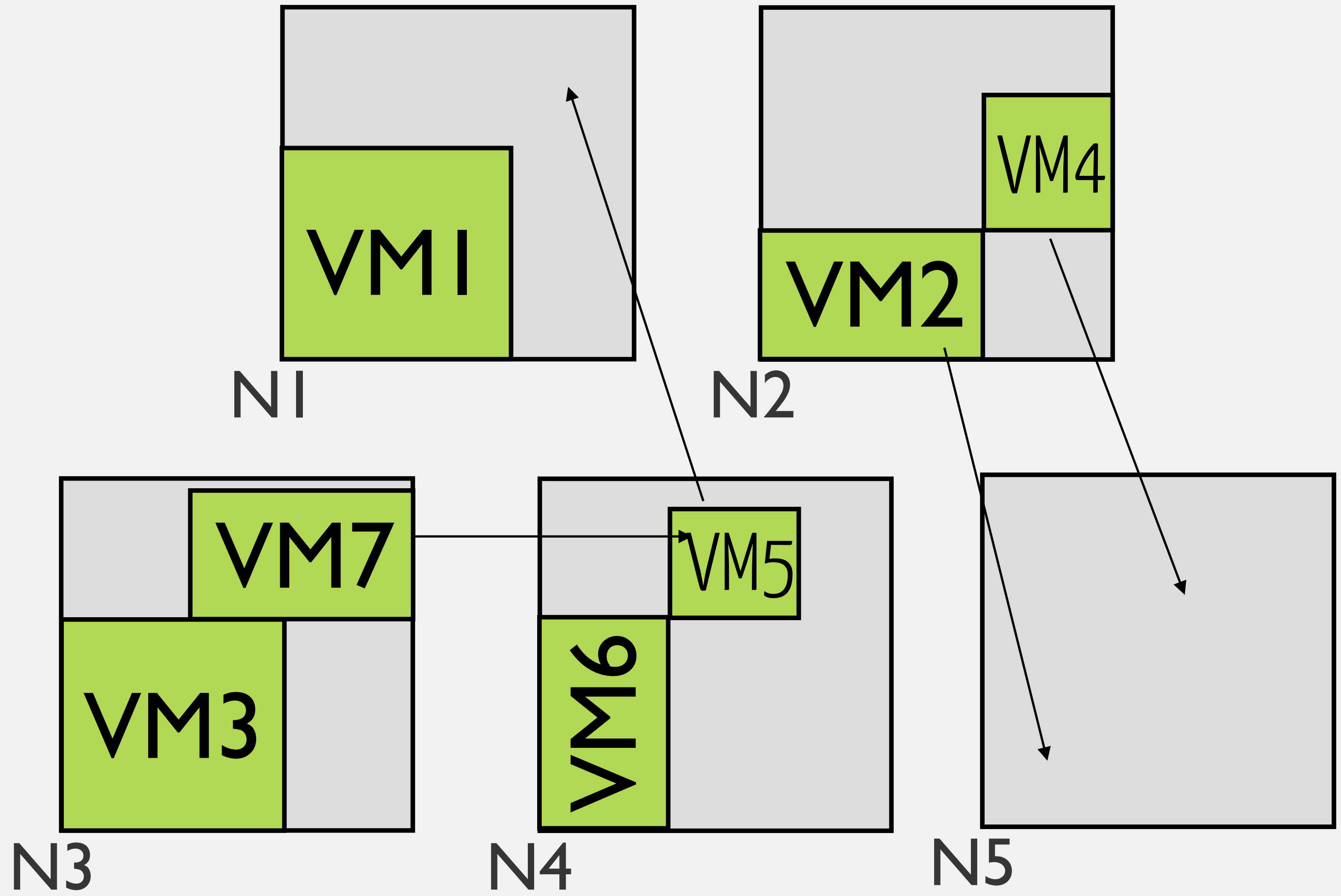
lower MTTR
(faster)

dynamic scheduling using vector packing

[10, 12]



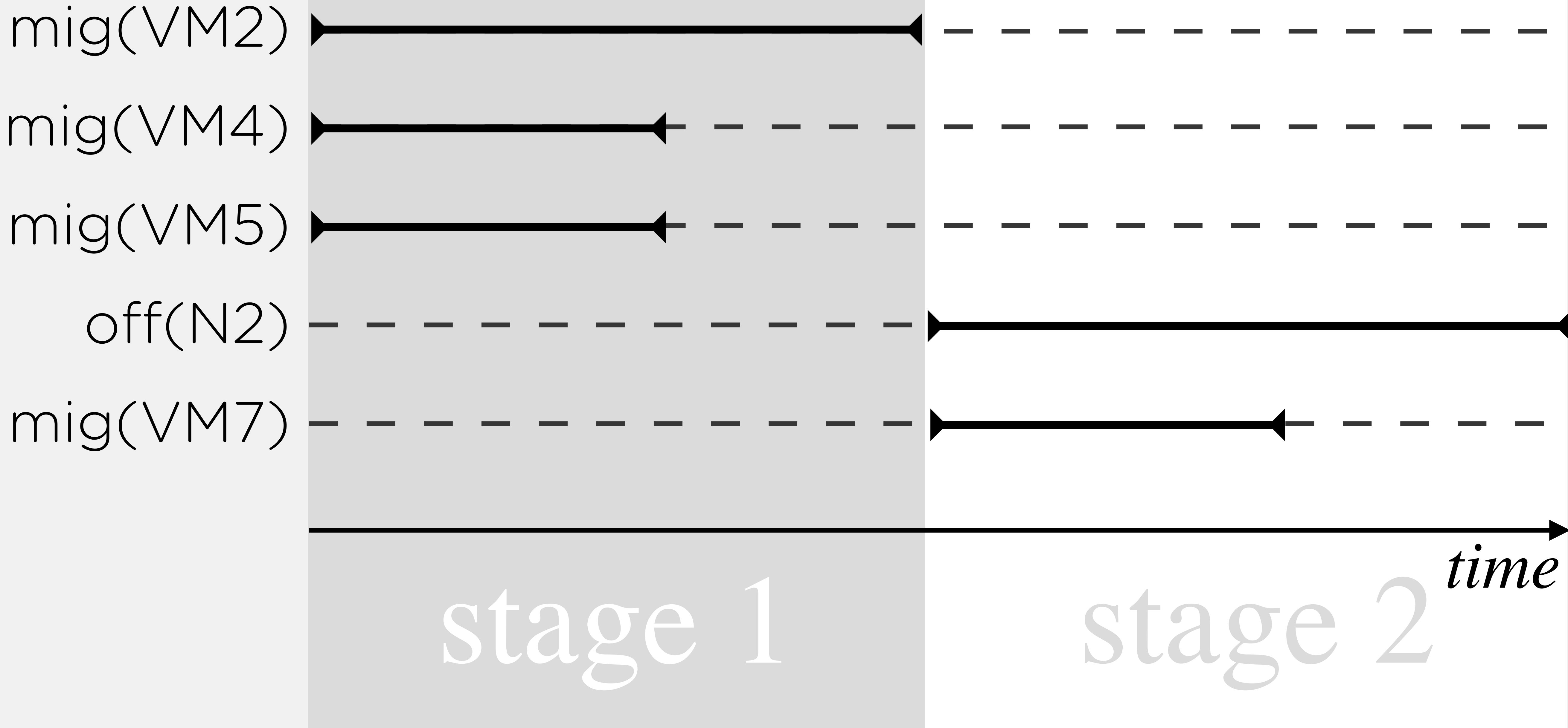
offline(N2) + no CPU sharing



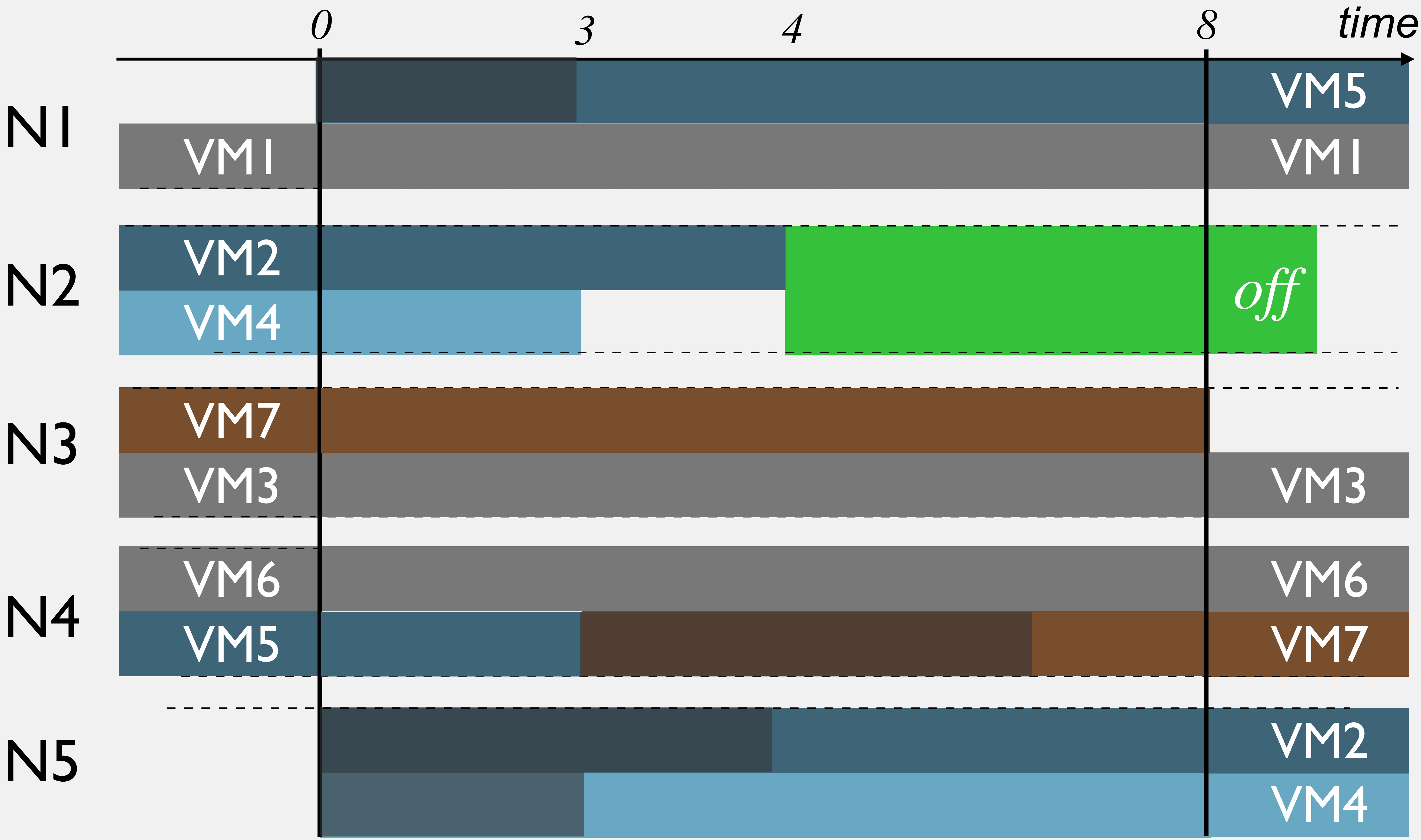
Dependency management

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

coarse grain staging delay actions



Resource-Constrained Project Scheduling Problem [14]



Resource-Constrained Project Scheduling Problem

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

NP-hard problem

From a theoretical to a **practical** solution

duration may be longer

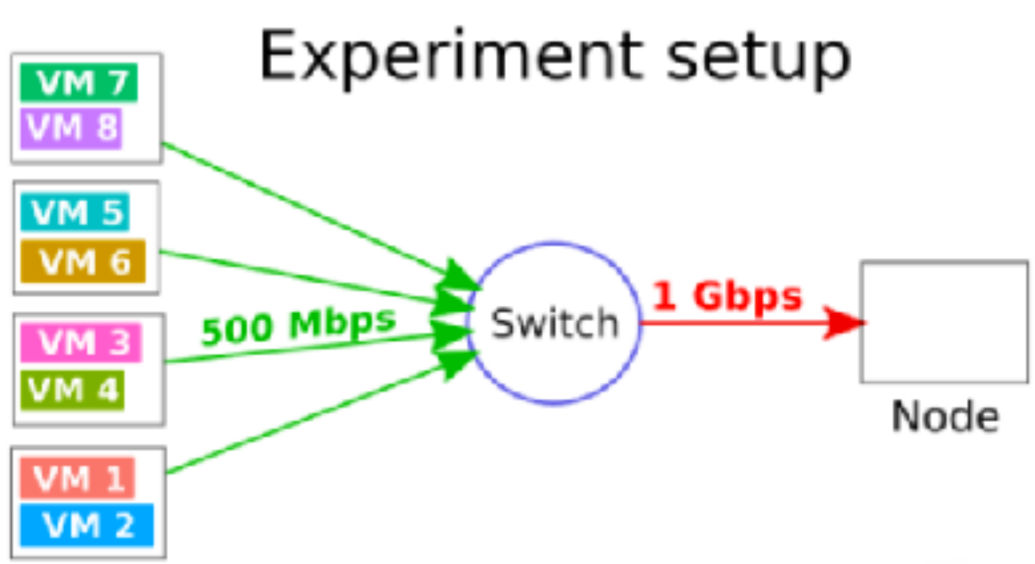
convert to an event based schedule

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

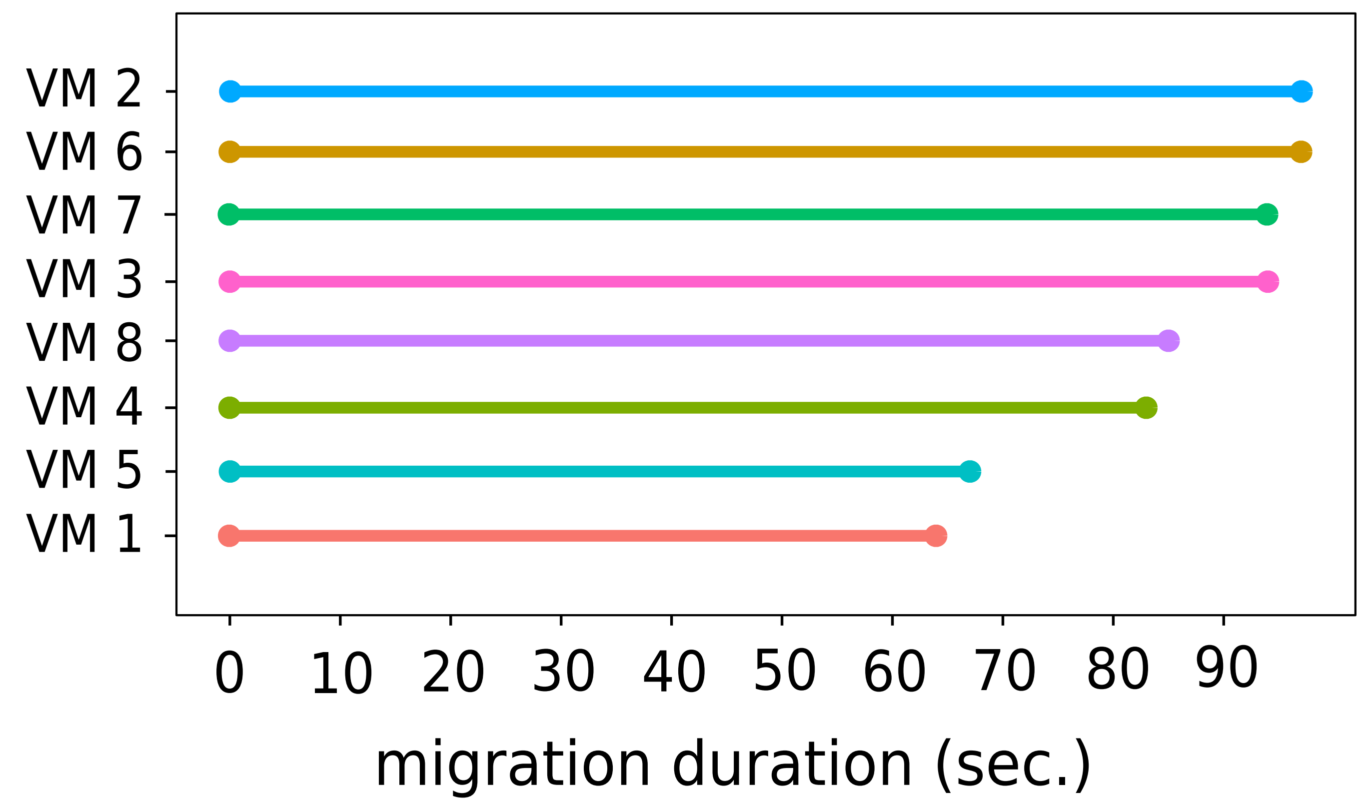
- : migrate VM4
- : migrate VM5
- : migrate VM2
!migrate(VM2) & !migrate(VM4): shutdown(N2)
!migrate(VM5): migrate VM7

Extensibility in practice

looking for a better migration scheduler



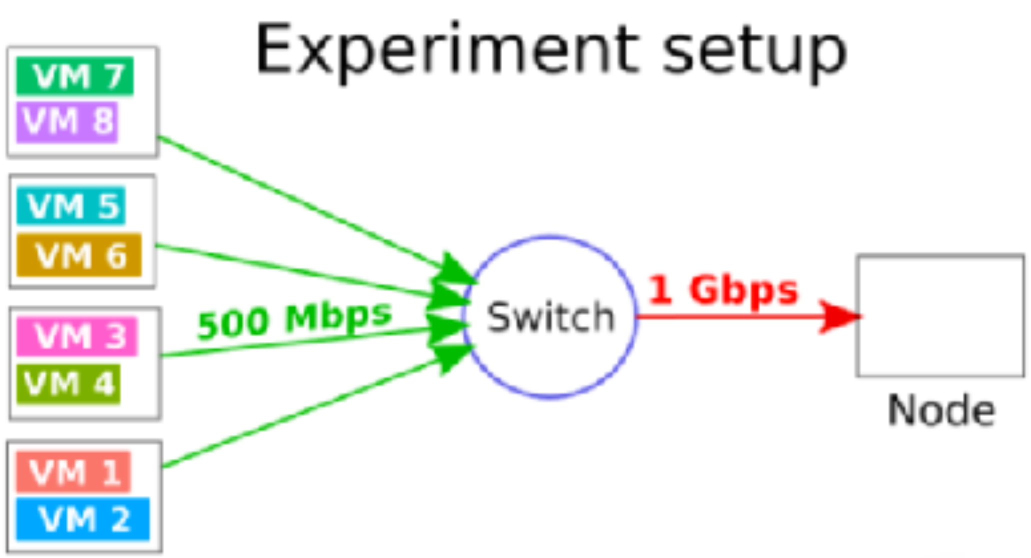
[btrplace vanilla, entropy, cloudsim, ...]



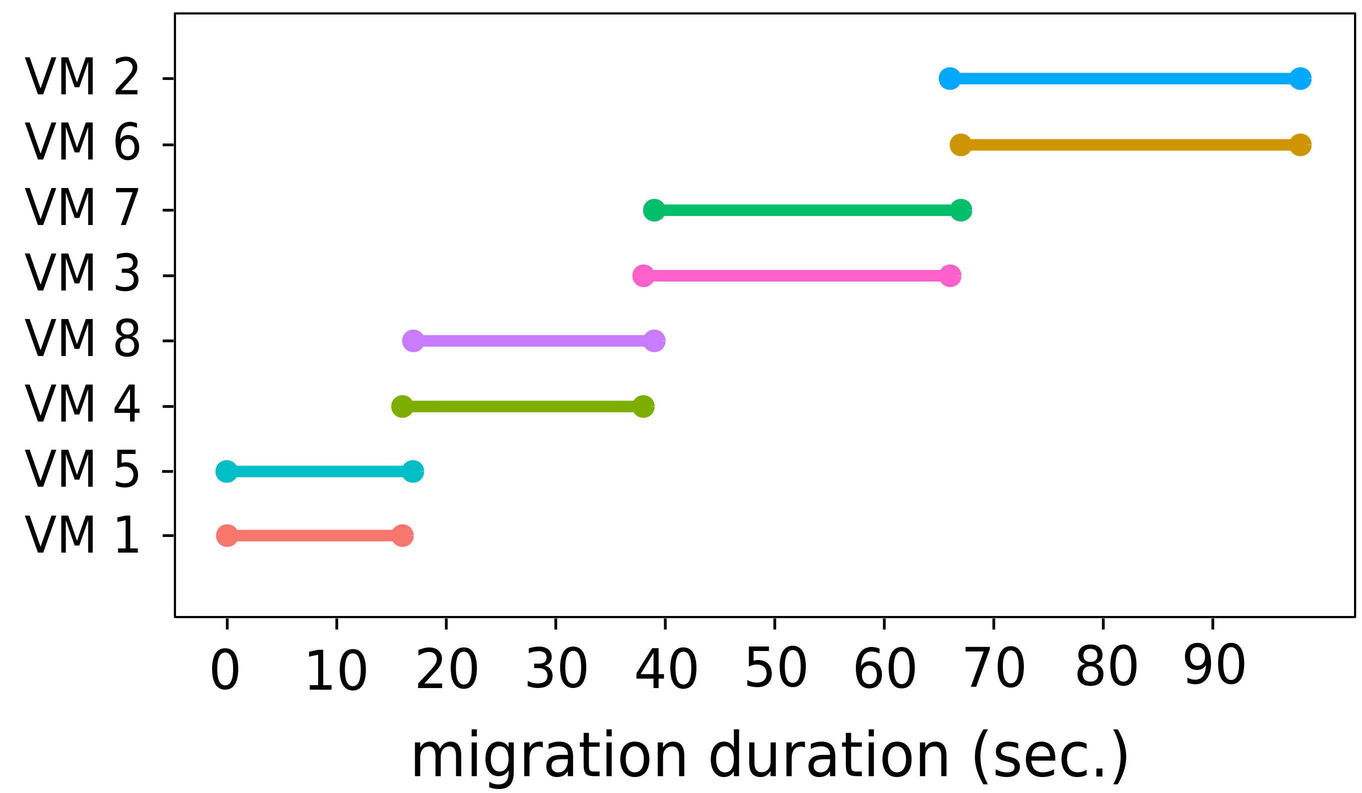
network and workload blind

Extensibility in practice

looking for a better migration scheduler



btrplace + migration scheduler [16]



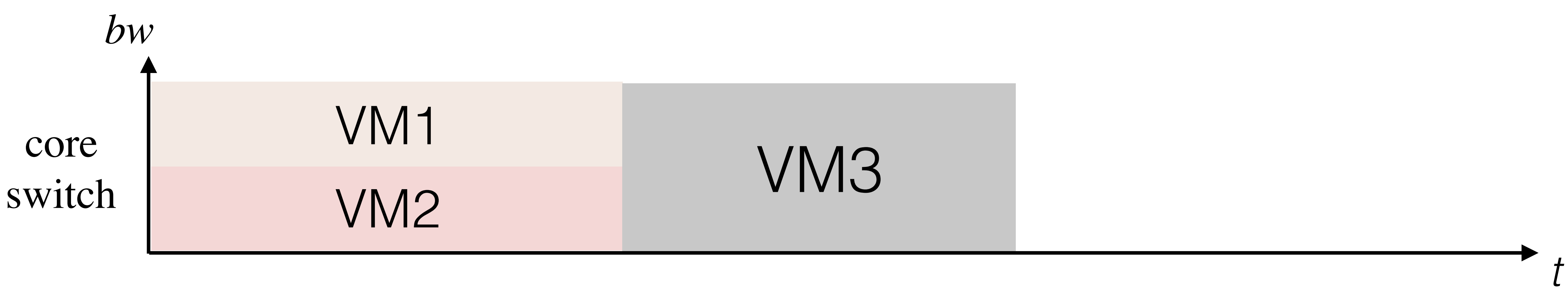
network and workload aware

Extensibility in practice
solver-side

Network Model — heterogeneous network
cumulative constraints; +/- 300 sloc.

Migration Model — memory and network aware
+/- 200 sloc.

Constraints Model — restrict the migration models
+/- 100 sloc.



Nobody's perfect

placement — vector packing problem

scheduling — multi-mode resource-constrained project scheduling problem

exact approaches:

1000 VMs / 10 nodes -> 10^{1000} assignments

heuristics approaches: fast but approximatives

scaling

NP-hard problems

```
.[1/2] relocatable(vm#0).dSlice_hoster = {31}
..[1/2] relocatable(vm#1).dSlice_hoster = {31}
...[1/2] relocatable(vm#2).dSlice_hoster = {31}
....[1/2] relocatable(vm#3).dSlice_hoster = {31}
.....[1/2] relocatable(vm#4).dSlice_hoster = {31}
......[1/2] relocatable(vm#5).dSlice_hoster = {31}
```

the search heuristic

per objective

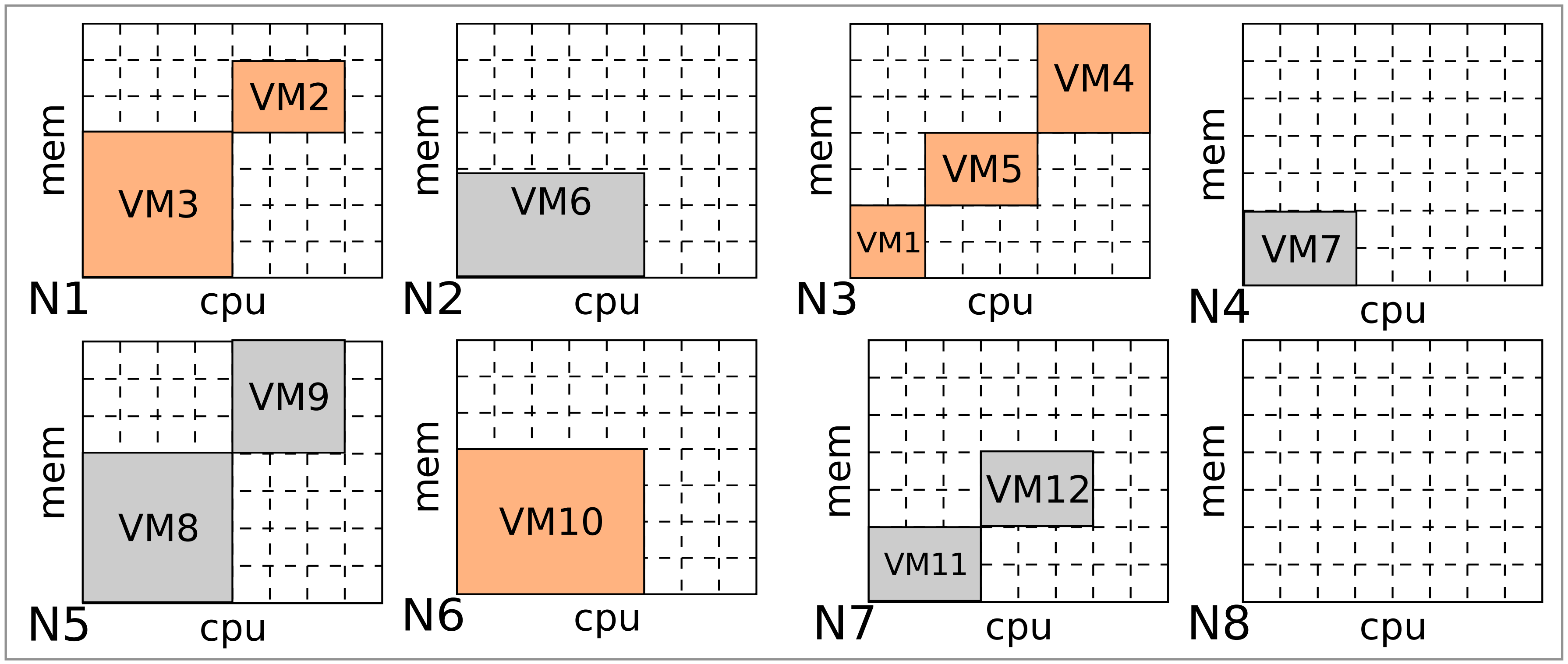
guide choco to instantiation of interest at each search node

1. which of the variables to focus
2. which value to try

do not alter the theoretical problem

```
.........[1/2] shutdownableNode(node#3).start = {0}
..........[1/2] shutdownableNode(node#2).start = {0}
...........[1/2] shutdownableNode(node#1).start = {0}
............[1/2] shutdownableNode(node#0).start = {0}
..............[1/2] relocatable(vm#97).cSlice_end = {1}
..................[2/2] relocatable(vm#202).cSlice_end \ {2}
...................[1/2] relocatable(vm#202).cSlice_end = {4}
                    [1/2] relocatable(vm#203).cSlice_end = {2}
```

static model analysis 101

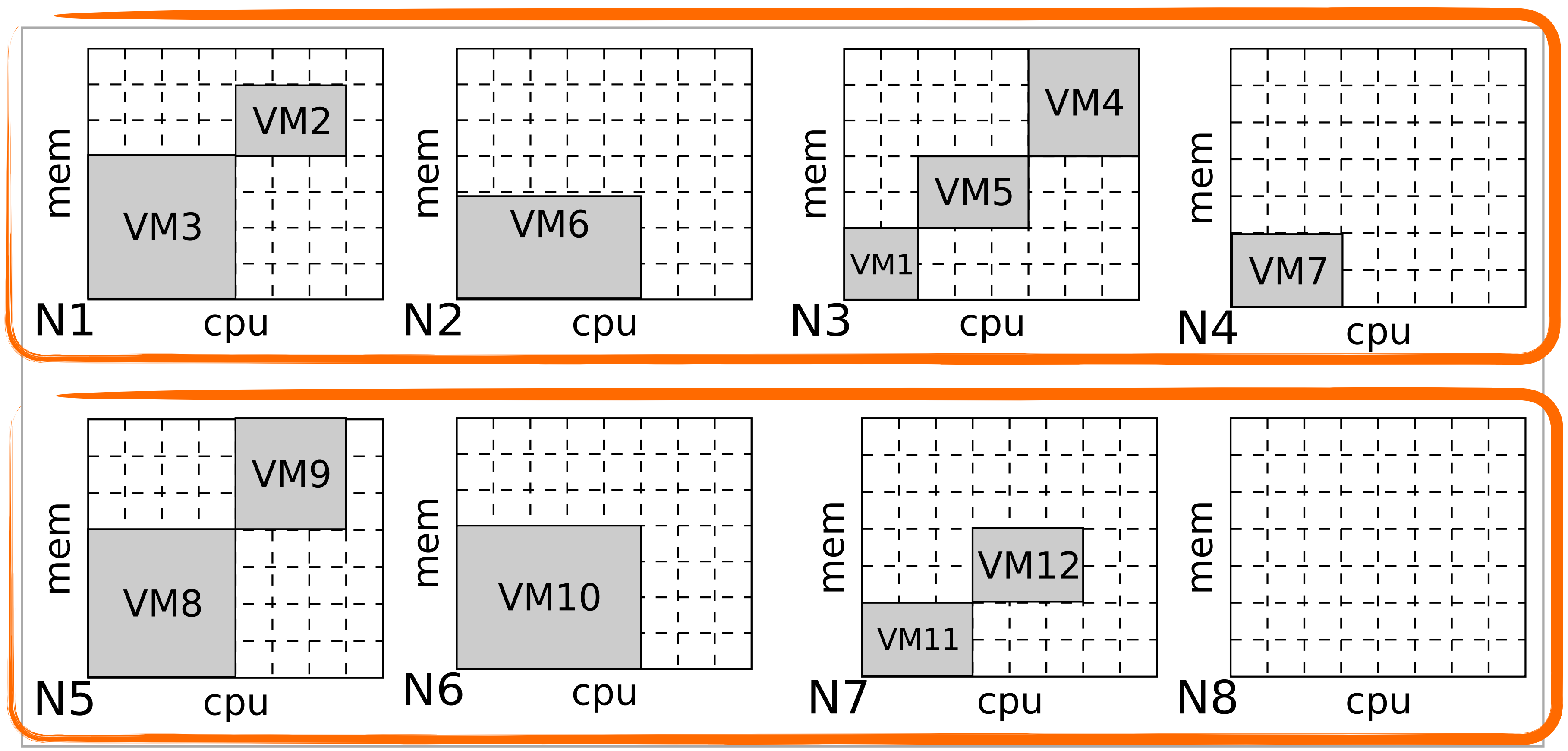



```
spread({VM3,VM2,VM8});
lonely({VM7});
preserve({VM1},'ucpu', 3);
offline(@N6);
ban($ALL_VMS,@N8);
fence(VM[1..7],@N[1..4]);
fence(VM[8..12],@N[5..8]);
```

```
scheduler.doRepair(true)
```

manage only supposed mis-placed VMs

beware of under estimations !



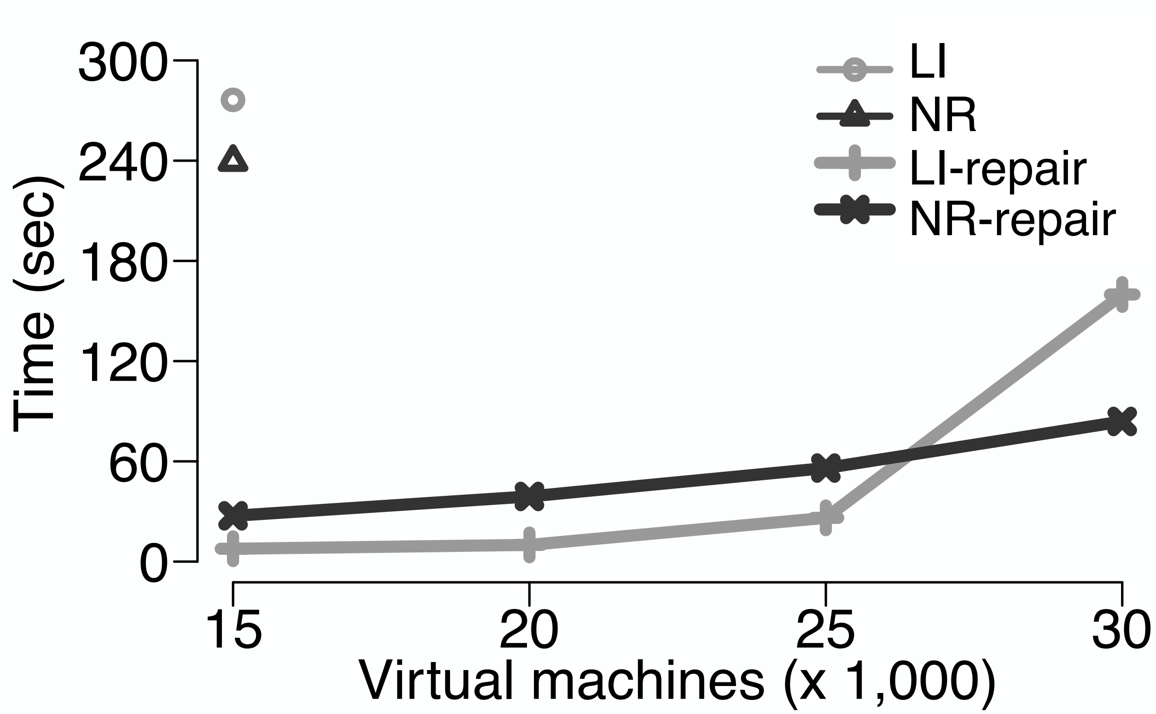

```
spread({VM3,VM2,VM8});
lonely({VM7});
preserve({VM1},'ucpu', 3);
offline(@N6);
ban($ALL_VMS,@N8);
fence(VM[1..7],@N[1..4]);
fence(VM[8..12],@N[5..8]);
```

```
s.setInstanceSolver(
  new StaticPartitioning())
```

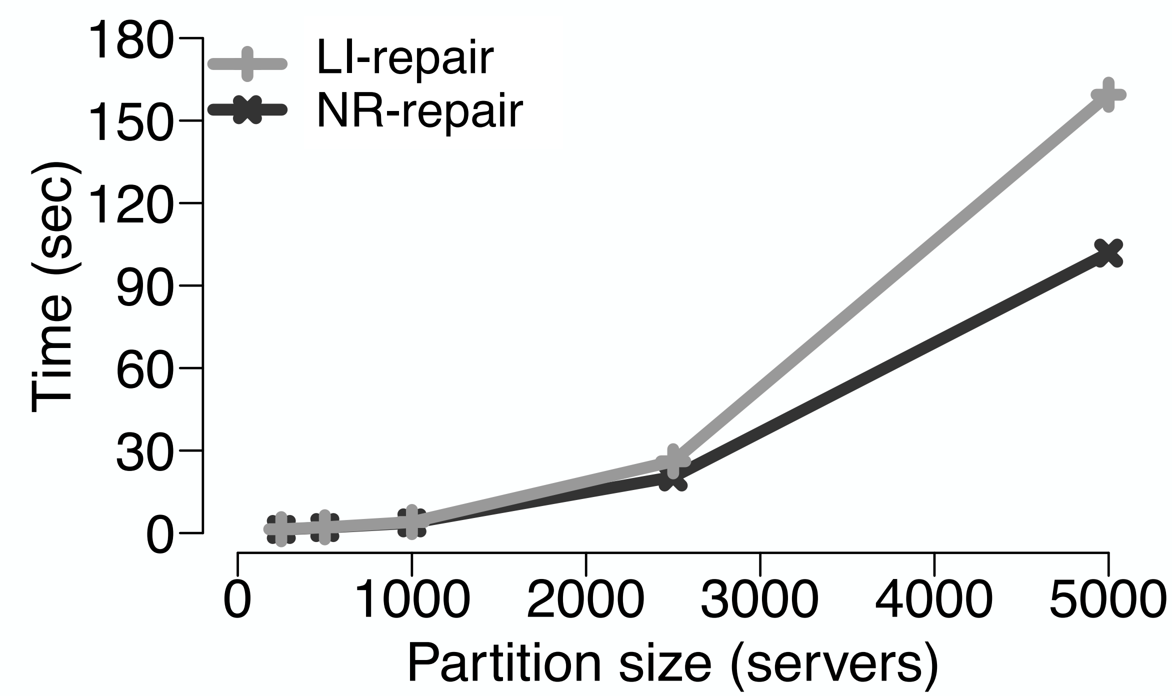
independent sub-problems solved in parallel

beware of resource fragmentation !

Repair benefits



Partitioning benefits

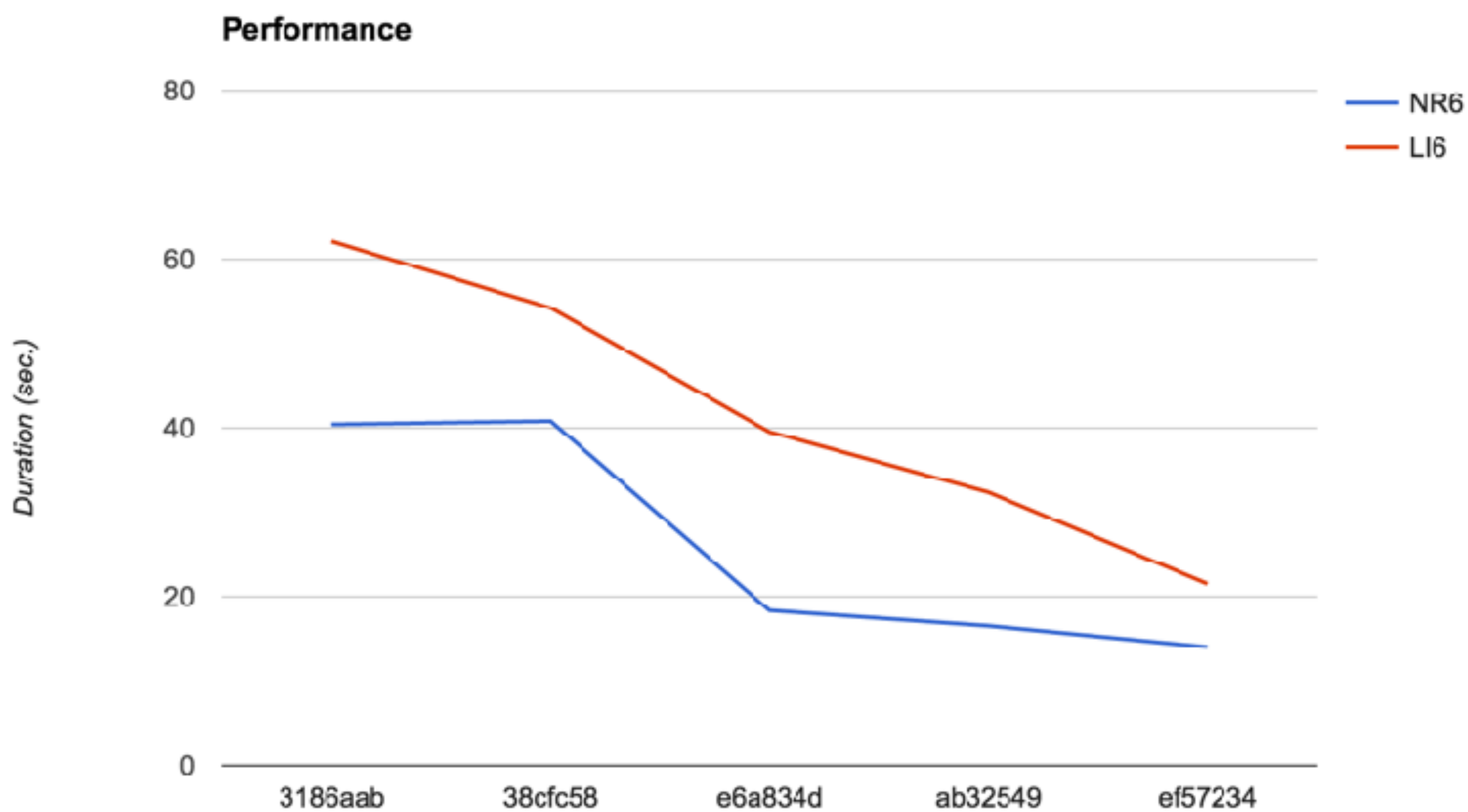


2013 perf numbers…

/!\ non Nutanix workloads

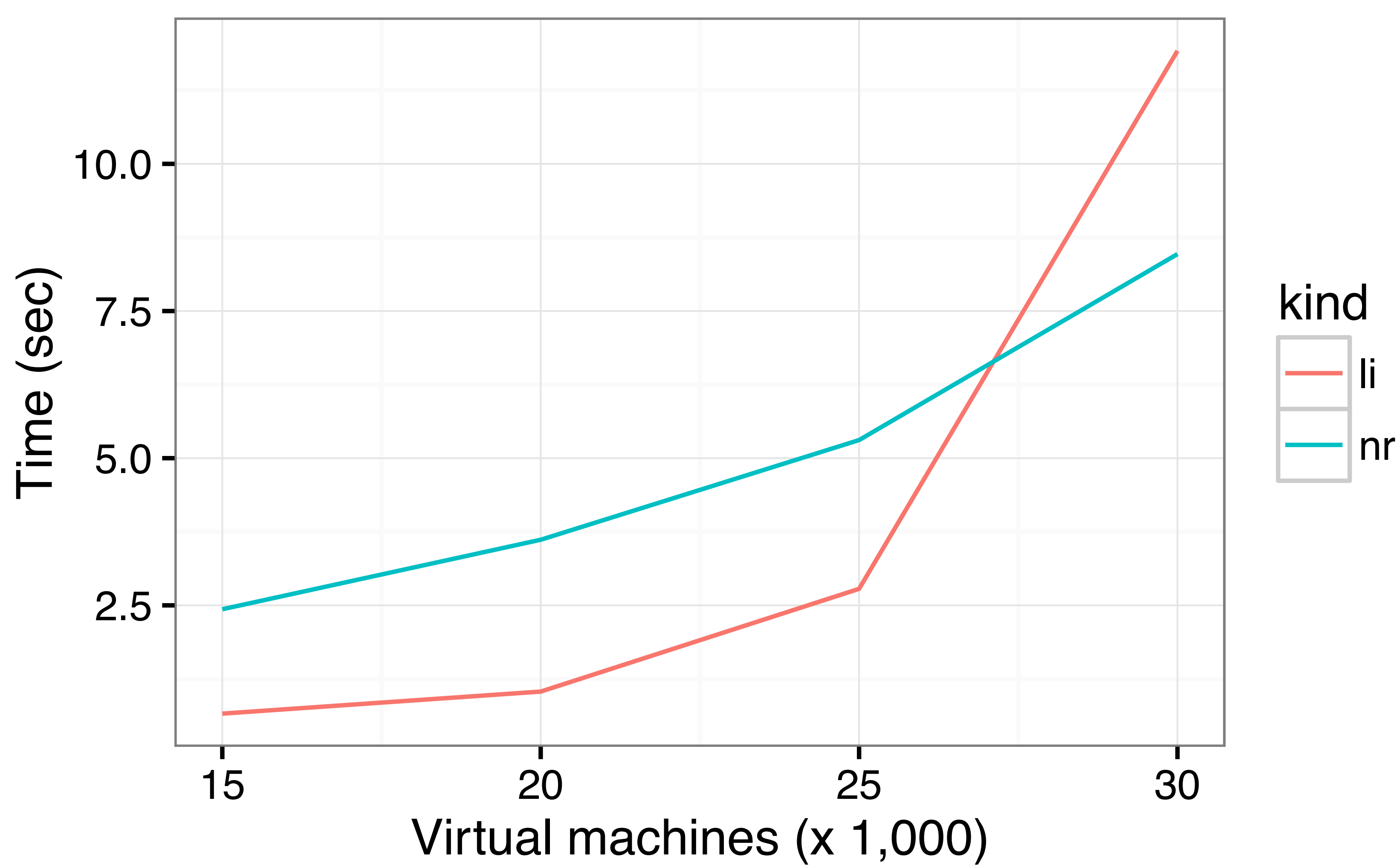
Master the problem

understand the workload,
tune the model, tune the solver, tune the heuristics



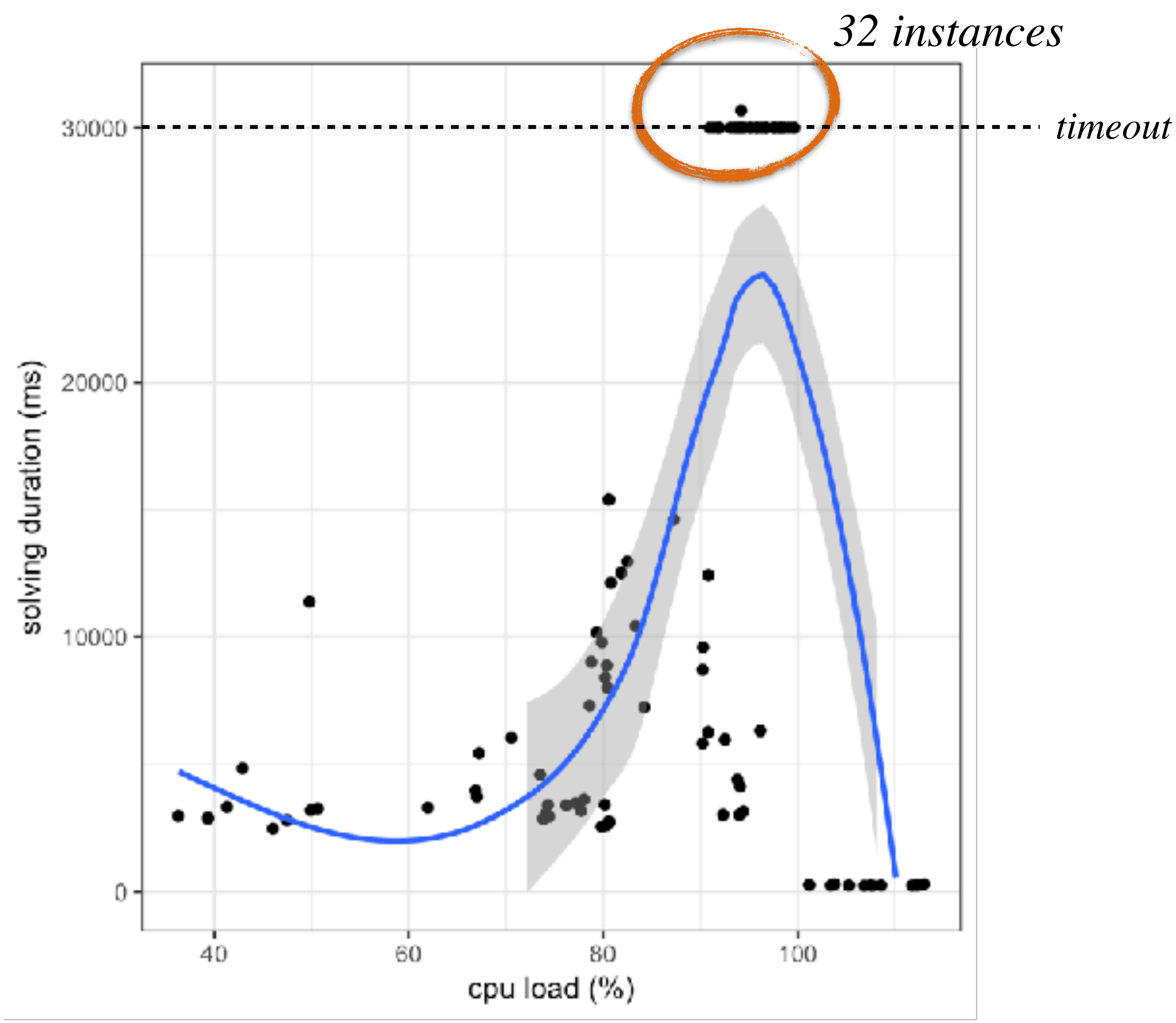
(benching on my laptop)
/!\ non Nutanix workloads

“current” performance



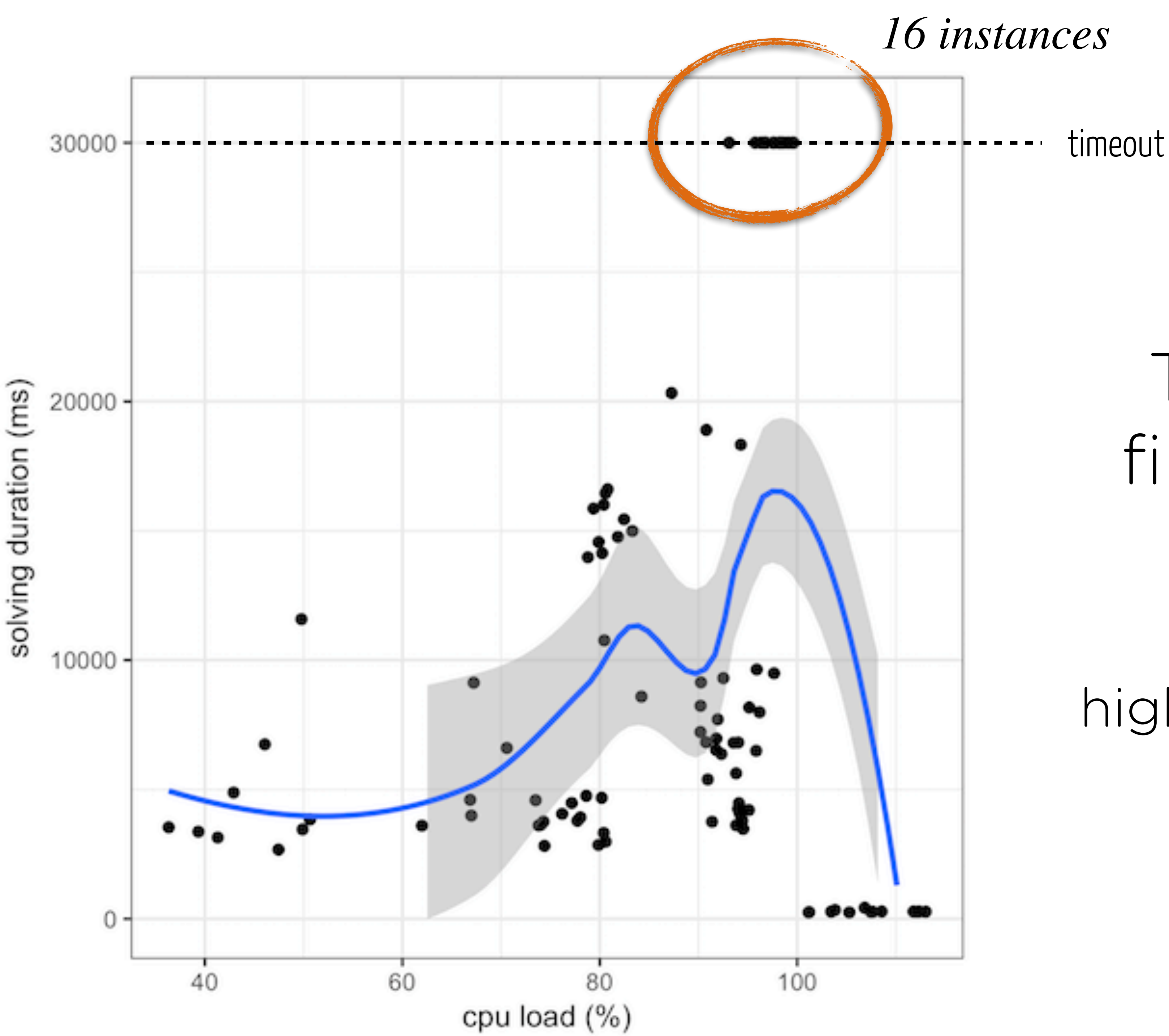
xeon servers
/!\ non Nutanix workloads

The right filtering algorithm for the right workload



very high load
small but hard instances

ok when non-solvable
but no evidence



The costly Knapsack filtering to the rescue

smarter but slower

higher memory consumption

bigger constants

trigger based

RECAP

The VM scheduler makes cloud benefits real

think about what is costly

static scheduling for a peaceful life

dynamic scheduling to cease the day

no holy grail

master the problem

with great power comes great responsibility

http://**BtrPlace**.org

production ready   live demo   stable user API   documented   tutorials

issue tracker   support   chat room

WE WANT YOU

(once graduated)

NUTANIX™

Member of Technical Staff

San Jose, California

2 yrs. postdoc

Sophia, France

resource management in edge computing

PrEsto Cloud

Efficiently connecting **CLOUD** & **EDGE**

References

1. Omega: flexible, scalable schedulers for large computer clusters. Eurosys'13
2. Sparrow: distributed, low latency scheduling, SOSP'13
3. Large-scale cluster management at Google with Borg. Eurosys 15
4. Firmament: fast, centralized cluster at scale. OSDI 16
5. live-migration of virtual machines. NSDI'05
6. VMWare DRS. 2006
7. OpenStack Watcher. 2016
8. Nutanix Acropolis Dynamic Scheduler. 2017
9. Virtual Machine Consolidation in the Wild. Middleware 2014
10. Entropy: a consolidation manager for clusters. VEE 2009
11. pMapper: power and migration cost aware application placement in virtualized systems. Middleware 2009
12. Memory Buddies: exploiting page sharing for smart consolidation in virtualised data centres. VEE 2009
13. Energy-aware resource allocation heuristics for efficient management of data centres for cloud computing. FGCS 2012
14. BtrPlace: a flexible consolidation manager for highly available applications. TDSC 2013
15. Higher SLA satisfaction in datacenter with continuous VM placement constraints. HotDep 2013
16. Scheduling live-migrations for fast, adaptable and energy-efficient relocation operations. UCC 2015
17. Guaranteeing high availability goals for virtual machine placement. ICDCS 2011
18. The Acropolis Dynamic Scheduler. http://nutanixbible.com/