McGill

Microsoft Research

Performance Inconsistency in Large Scale Data Processing Clusters

Mingyuan Xia, Nan Zhu, Yuxiong He, Sameh Elnikety, Xue Liu

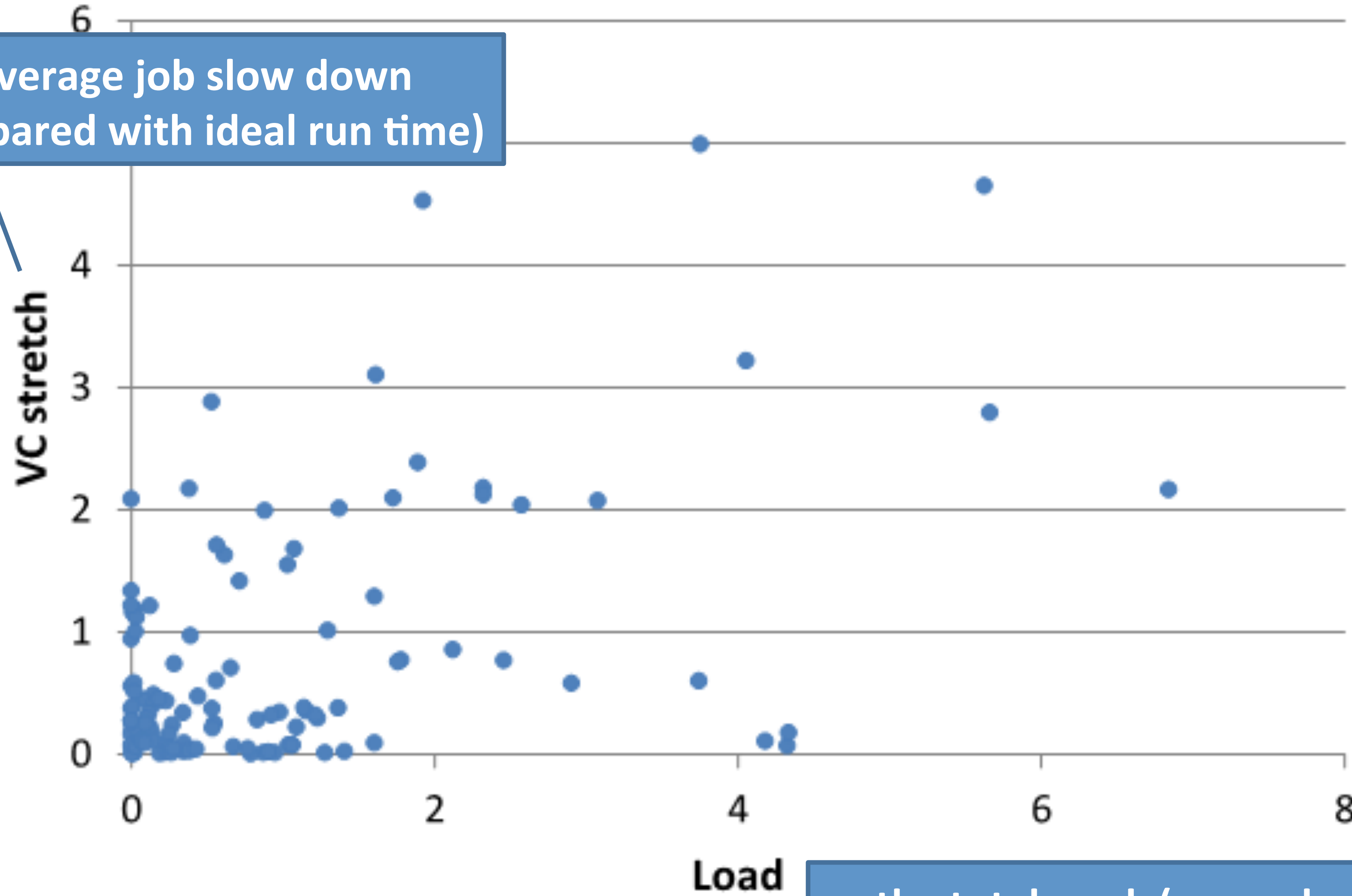
Large Computing Clusters

- e.g., MapReduce, Hadoop, Cosmos
  - Enable large data processing applications

- Sharing
  - Each user pays for using a fraction of the entire cluster (virtual cluster)
  - Fixed capacity, but resources can be shared among VCs to promote utilization

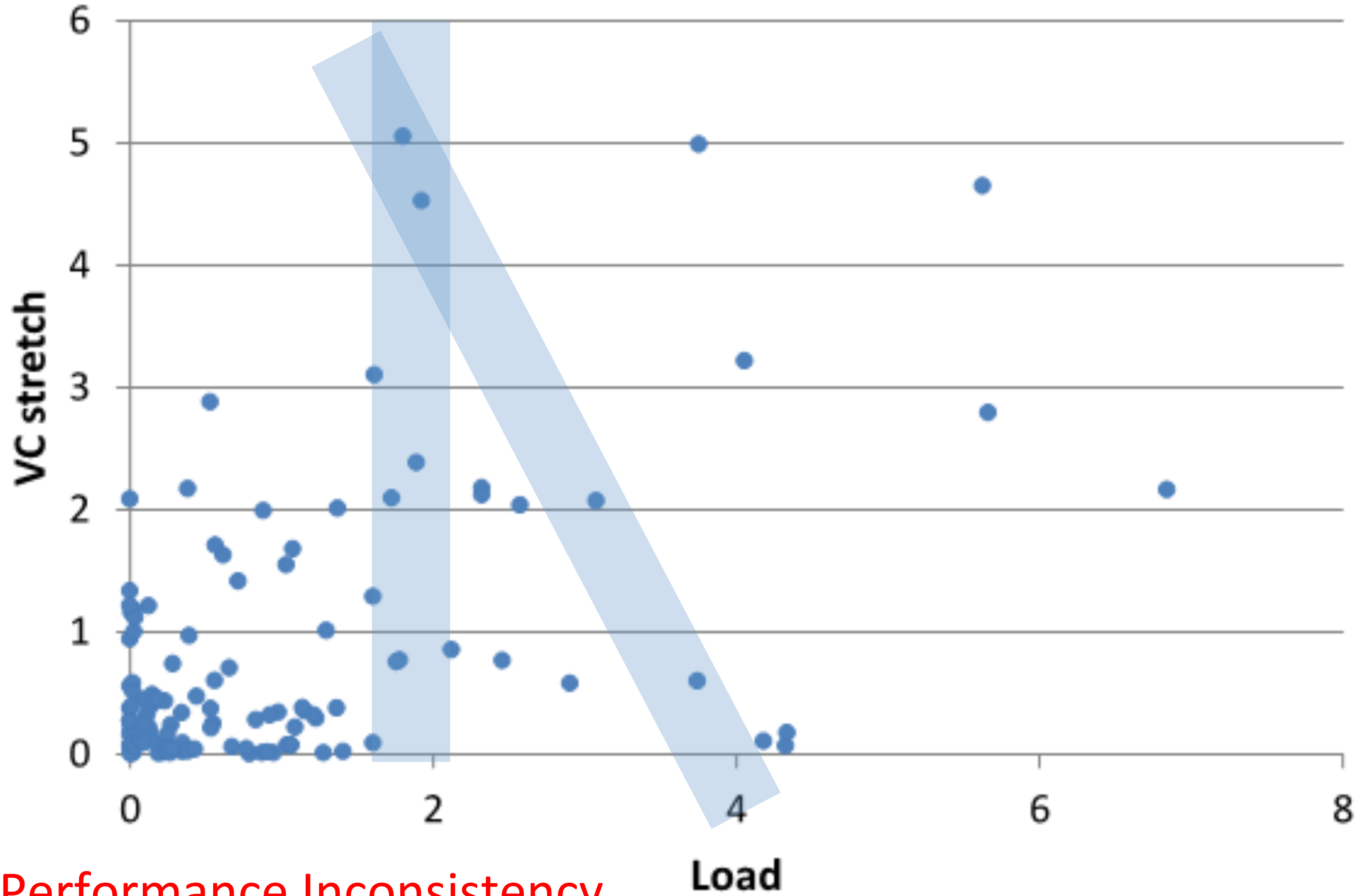
Production trace

average job slow down
(compared with ideal run time)



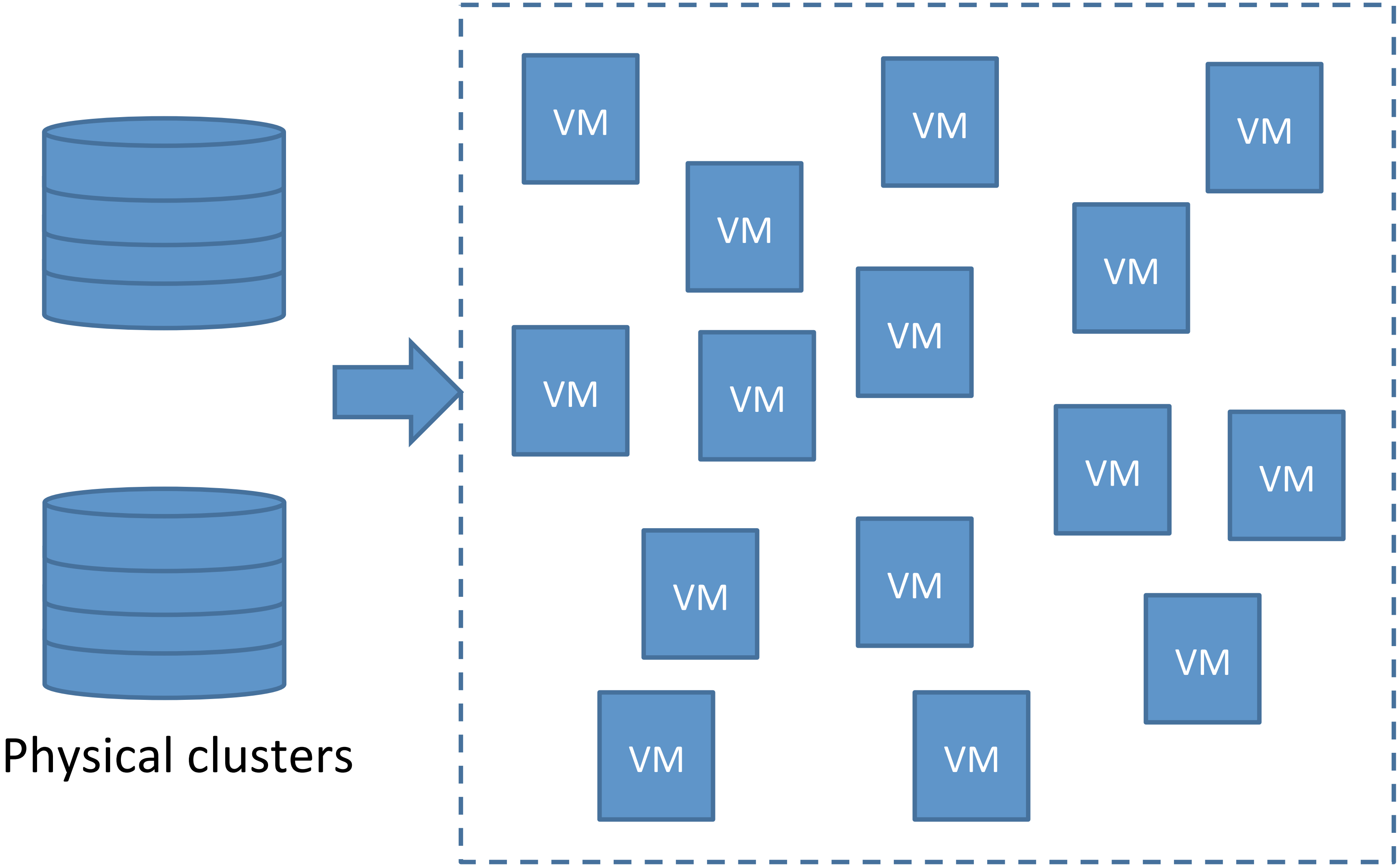
the total work (server hours)
/VC capacity (number of servers)

Production trace

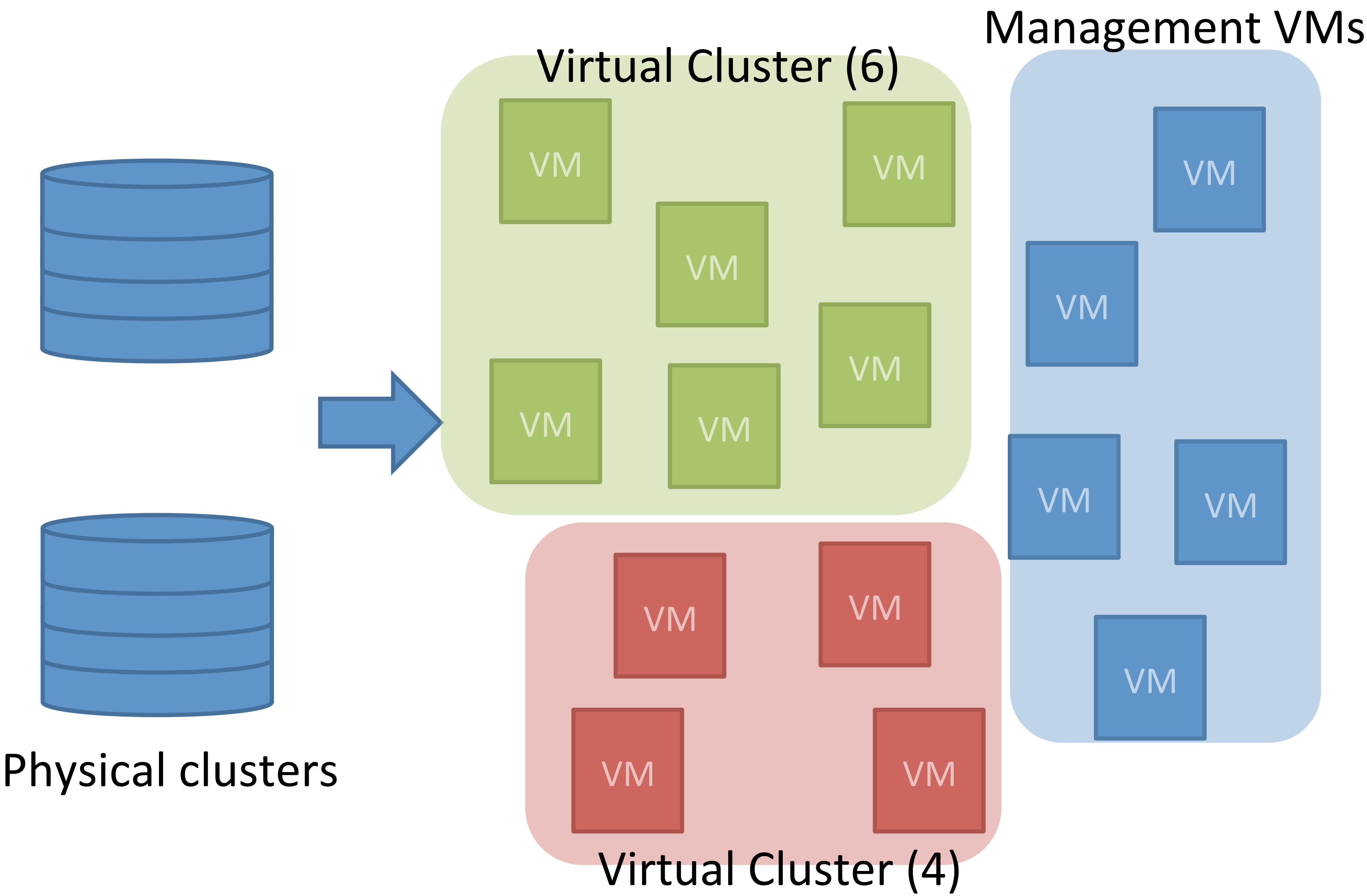


Performance Inconsistency

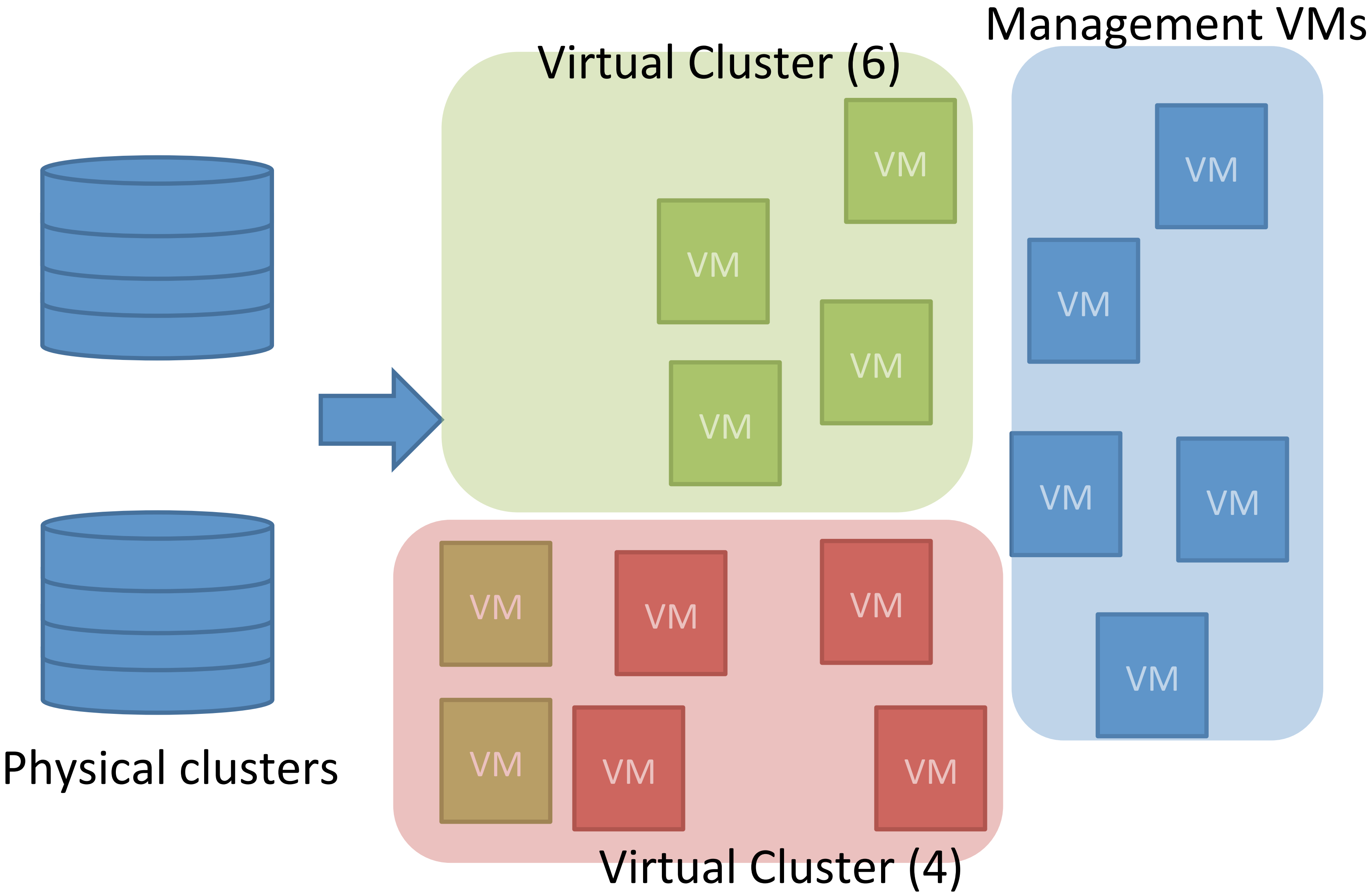
Cosmos



Cosmos


Management VMs
Virtual Cluster (6)
VM
VM
VM
VM
VM
VM
VM
VM
VM
VM
VM
VM
VM
VM
VM
Physical clusters
Virtual Cluster (4)

Cosmos

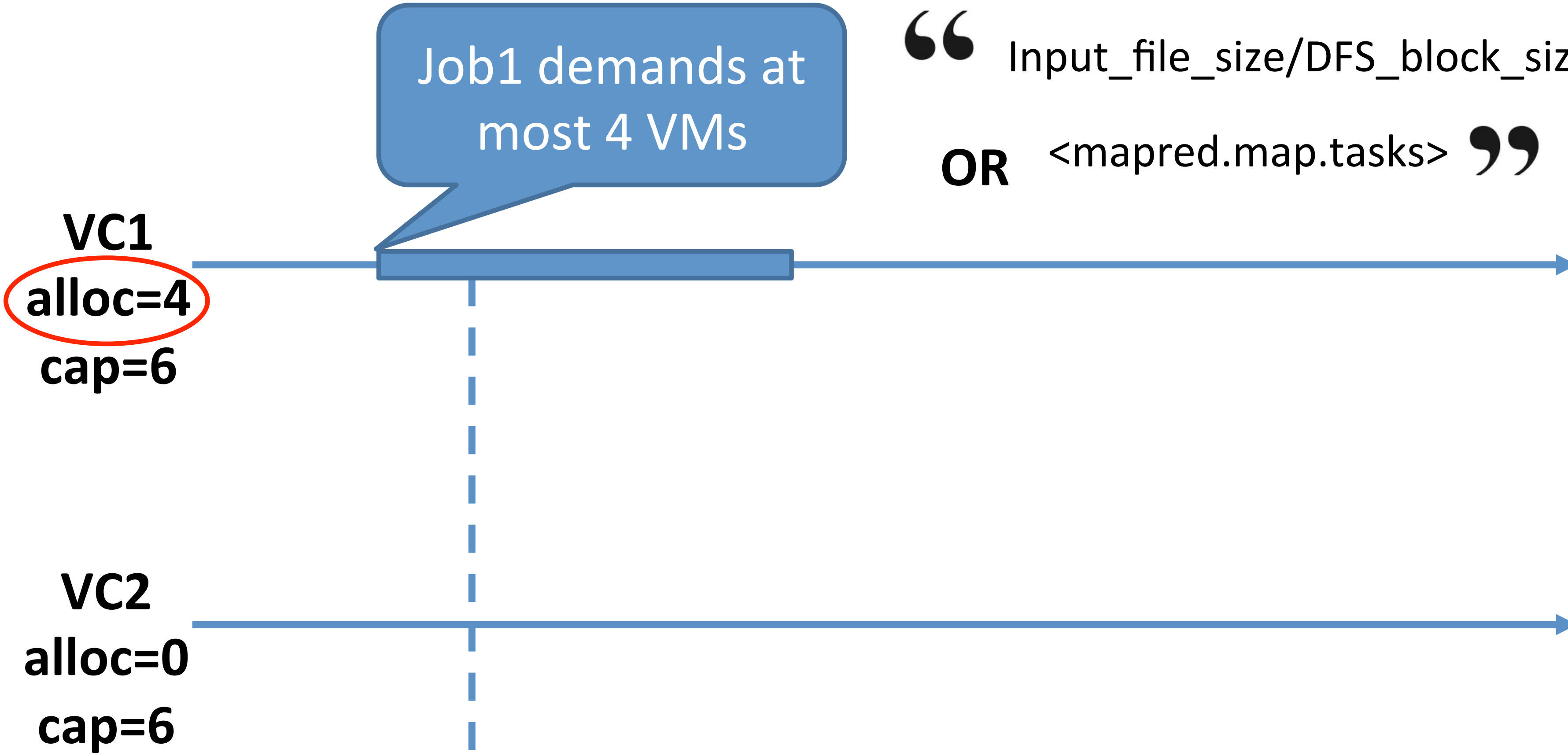

Management VMs
Virtual Cluster (6)
VM
VM
VM
VM
VM
VM
VM
VM
VM
VM
VM
VM
VM
VM
VM
VM
VM
Physical clusters
Virtual Cluster (4)

Job execution model

VC1
alloc=0
cap=6

VC2
alloc=0
cap=8

Job execution model

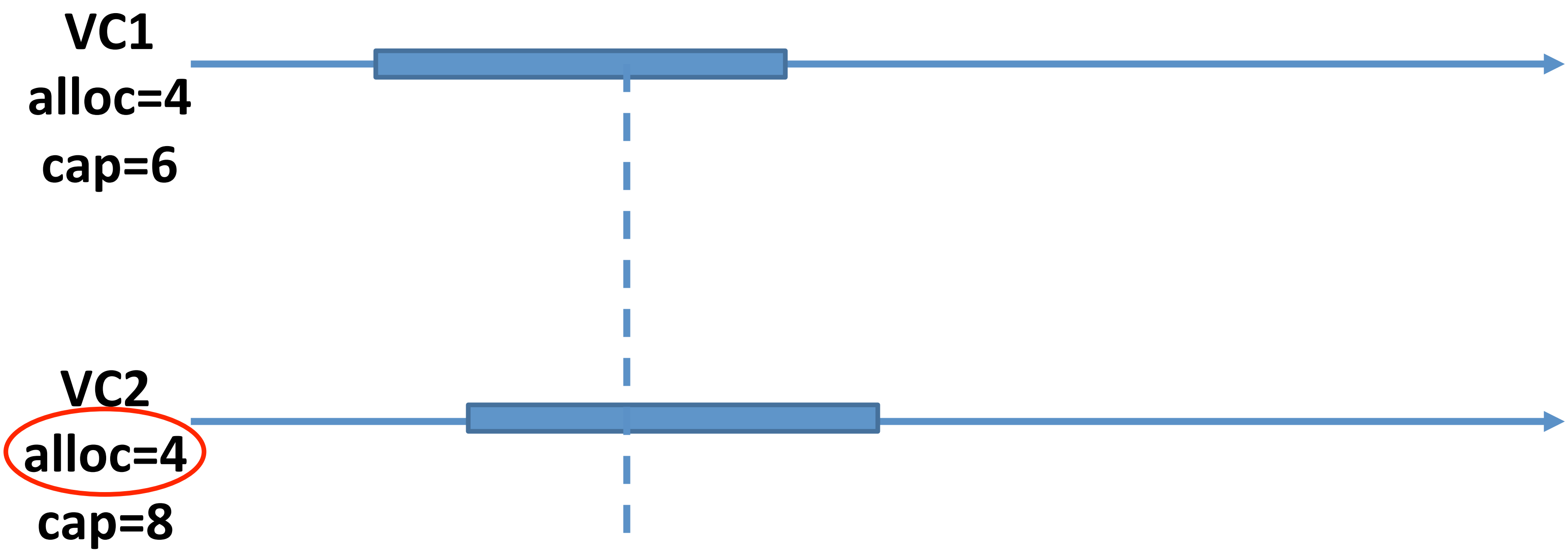


“ Input_file_size/DFS_block_size

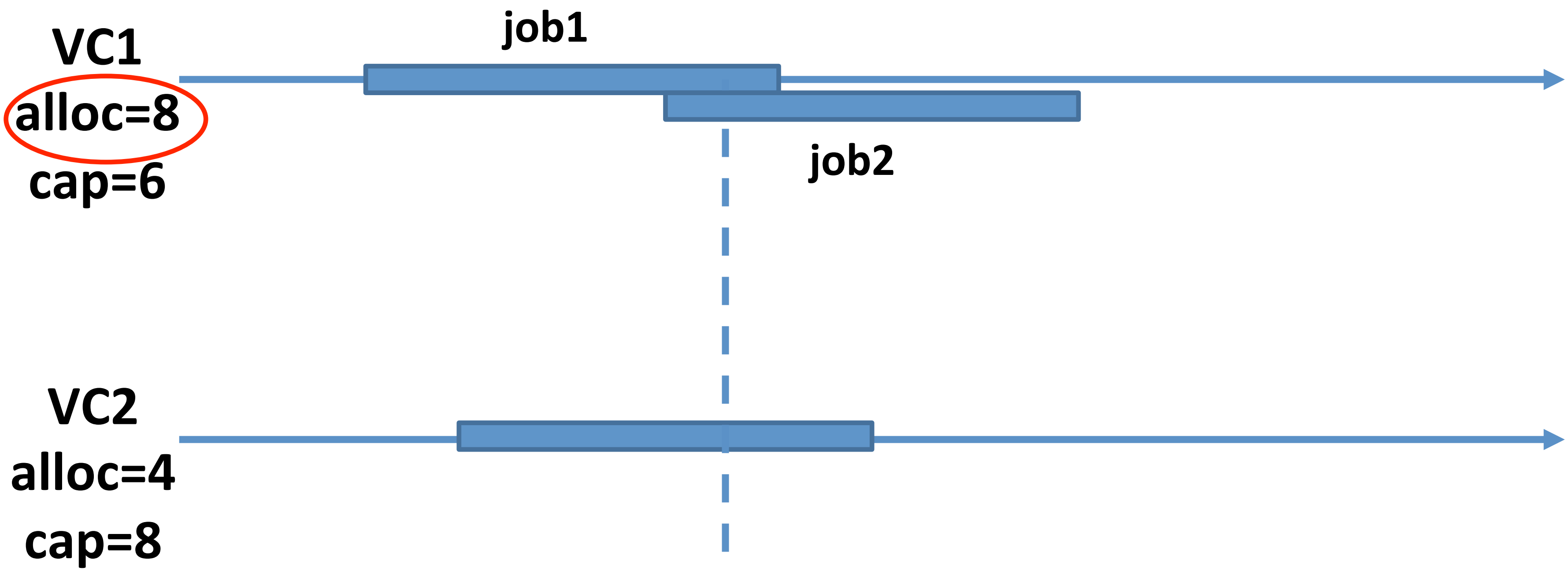
OR <mapred.map.tasks> ”

[1] http://wiki.apache.org/hadoop/HowManyMapsAndReduces

Job execution model

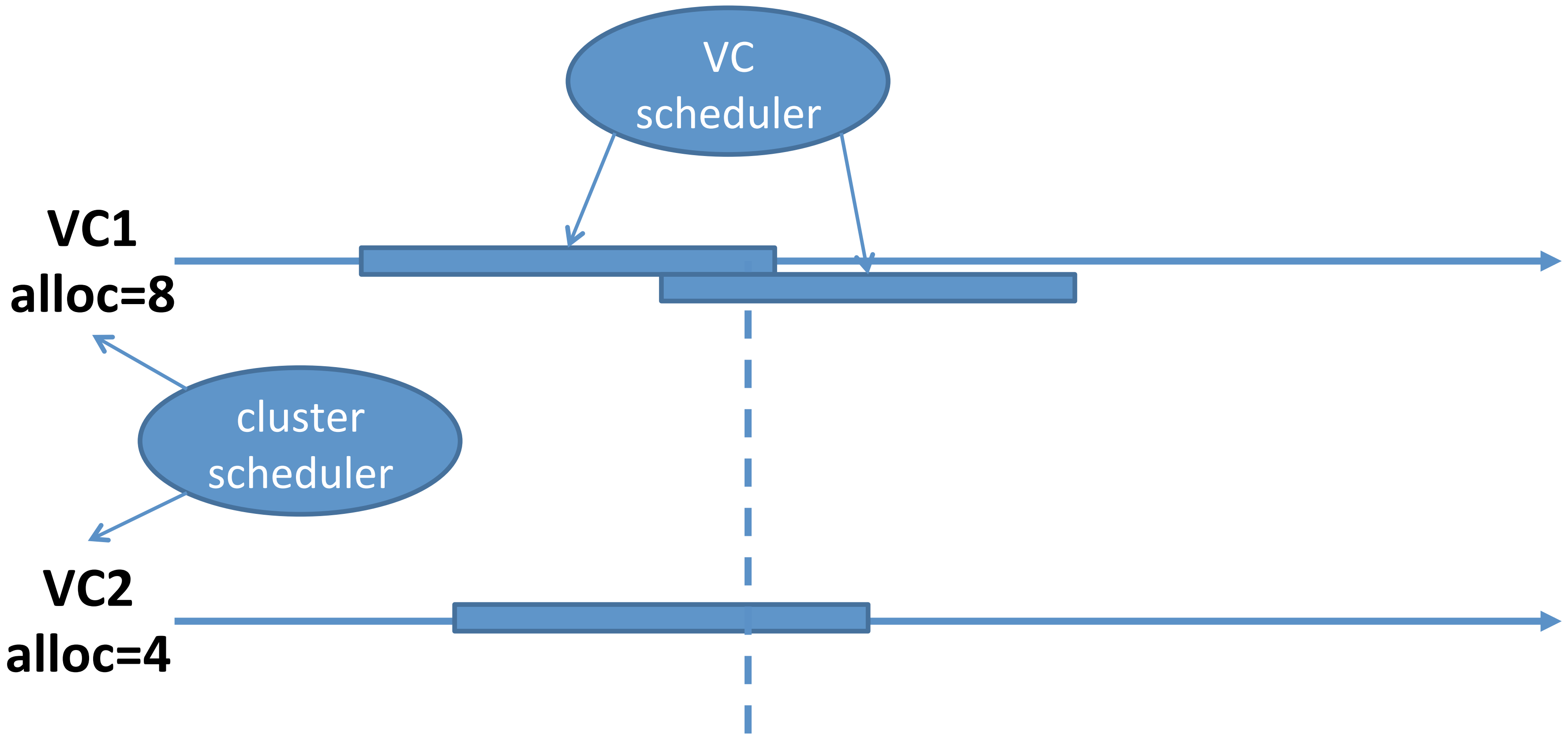


Job execution model



Sharing promotes overall system utilization

Job execution model



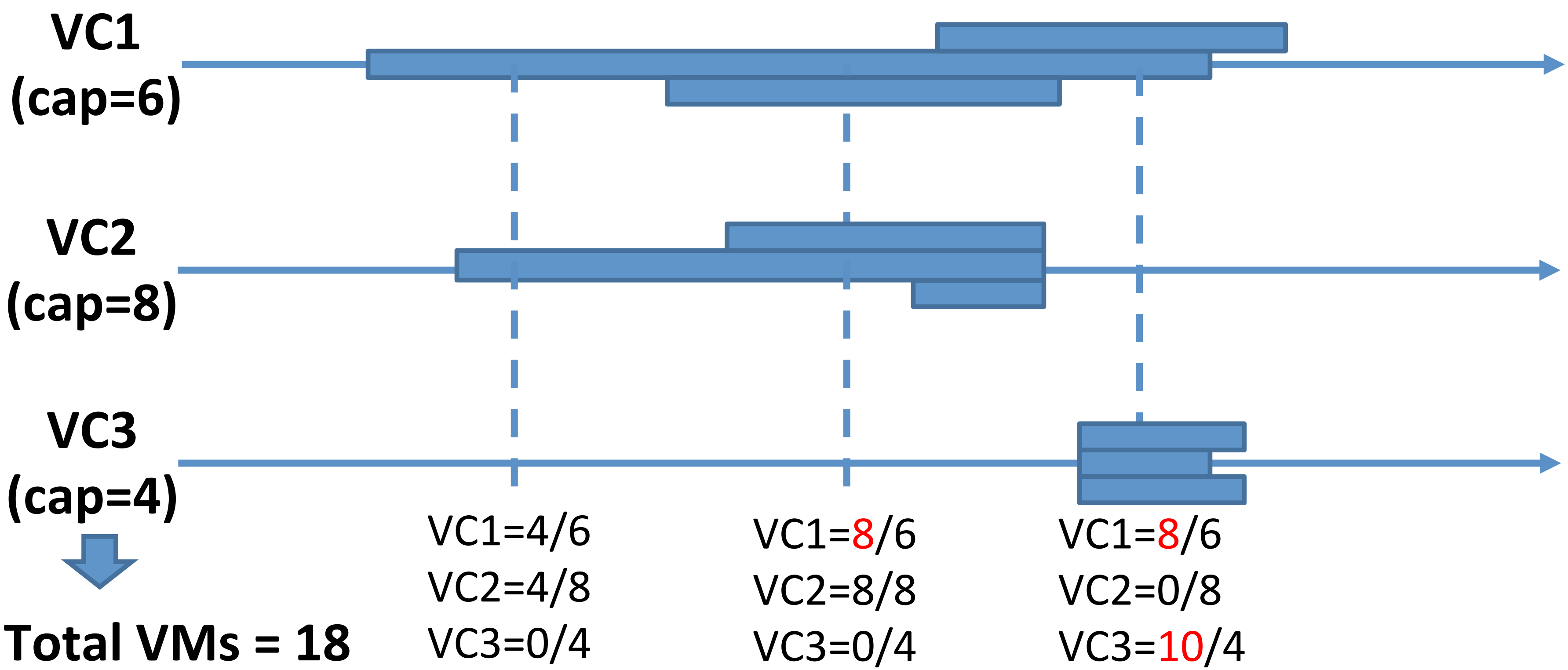
Our work tackles fairness at the cluster scheduler level.

Instantaneous fairness

- Consider parameters at a given time point
- MaxMin: Maximize the minimum allocation
  - Hadoop's fair scheduler is a variation

MaxMin: how it works

- Maximize the minimum allocation



MaxMin

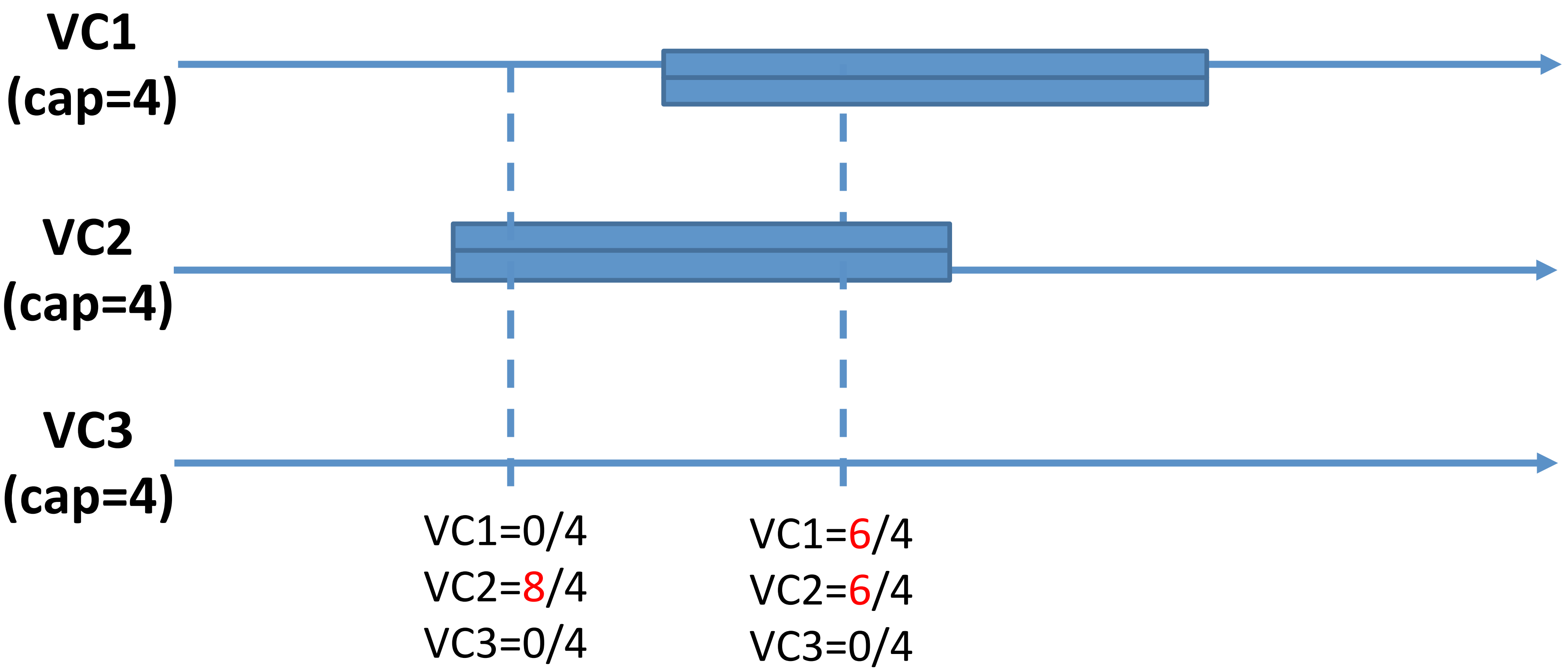
	VC1	VC2	VC3
Allocation	0	0	0
Demand	8	0	12
Capacity	6	8	4

Stage I (8 remaining)			
Allocation	6	0	4
Demand	2	0	8
Capacity	6	8	4

Stage II (4 remaining)			
Allocation	6+2	0	4+2
Demand	0	0	6
Capacity	6	8	4

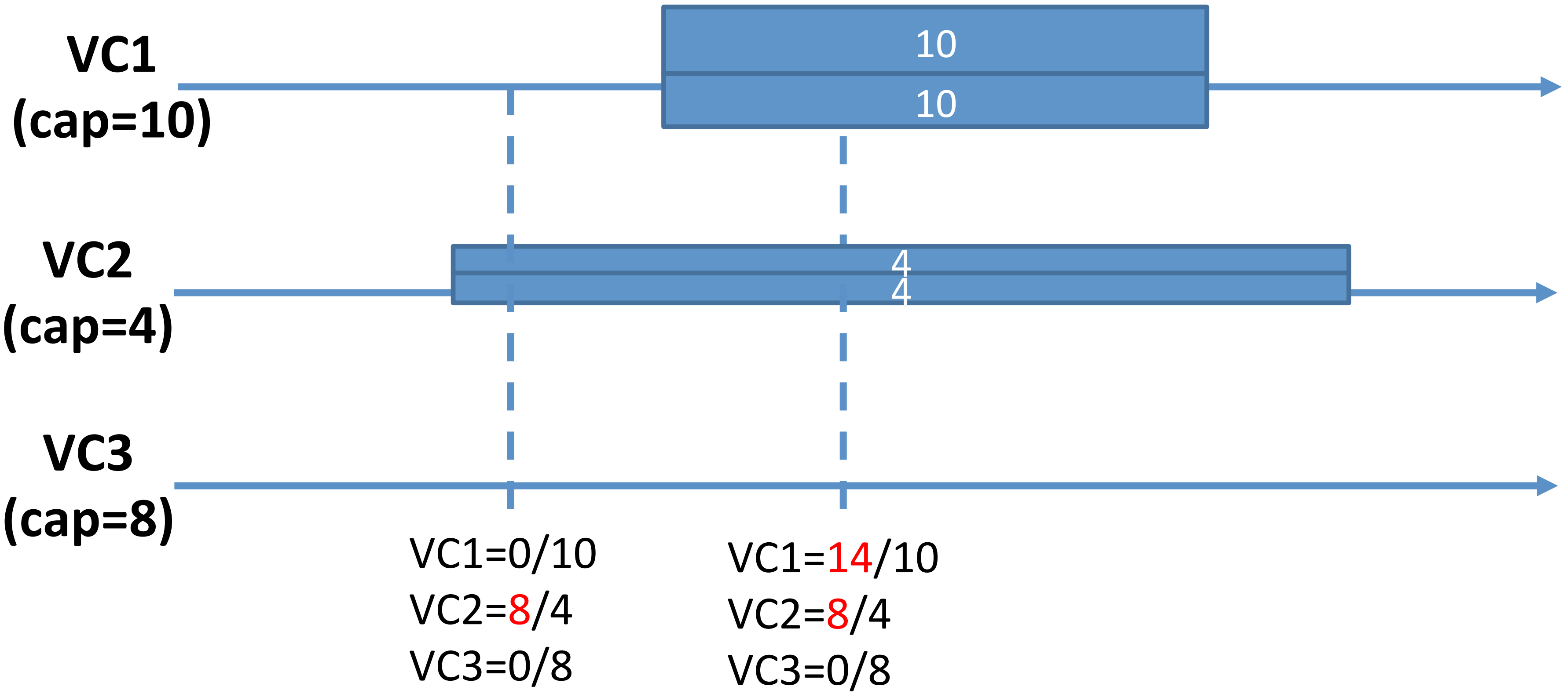
Long-term?

- Previous contribution is not rewarded



Long-term?

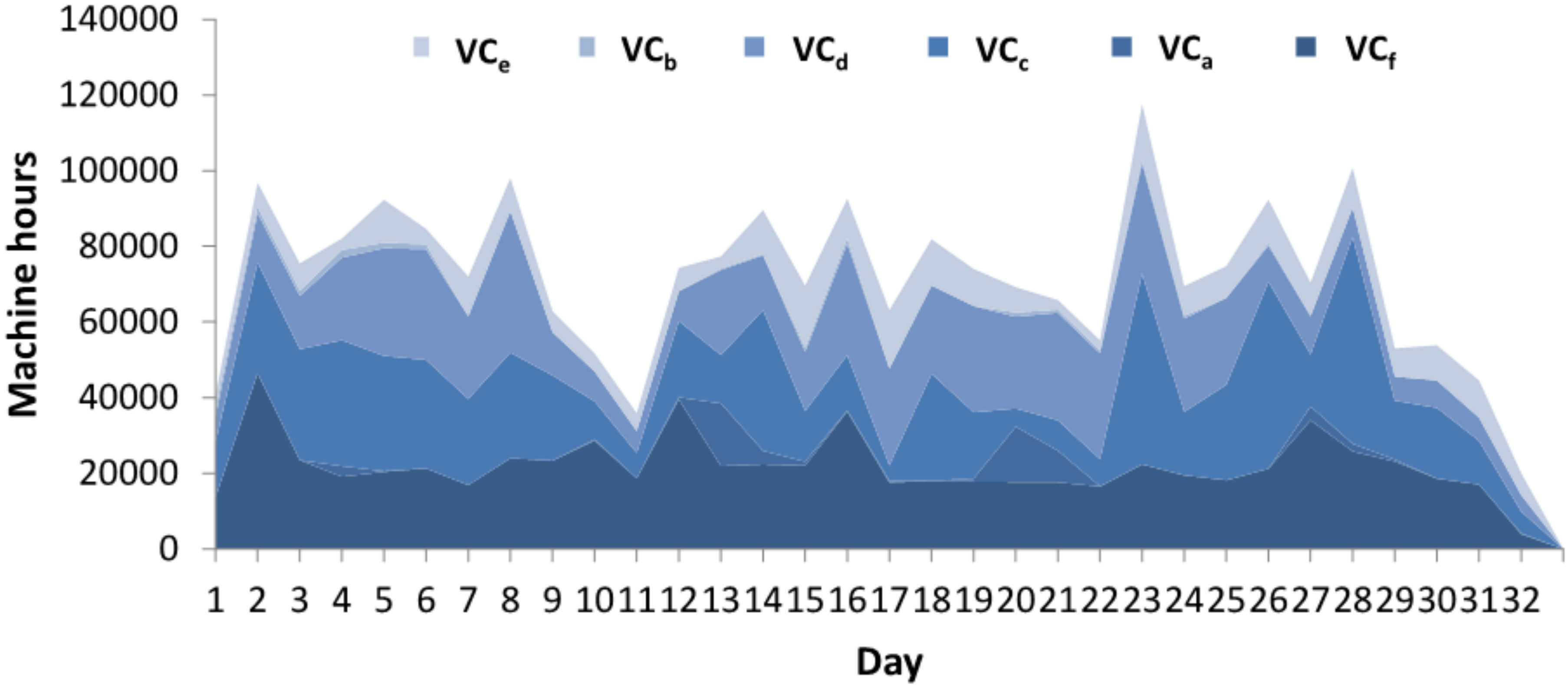
- Large VCs win fewer resources



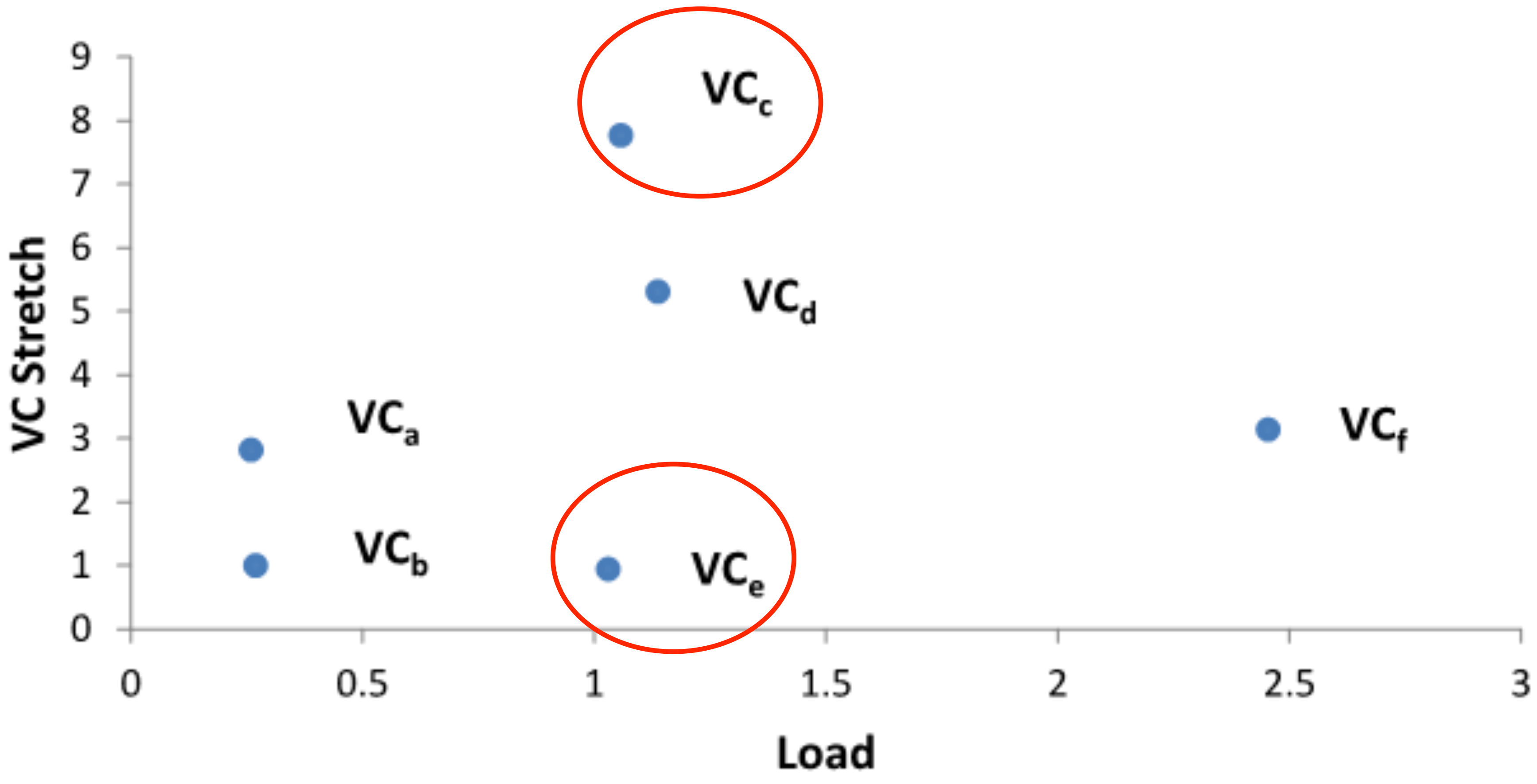
Trace-driven simulation

- Build a simulator
- Production trace from a commercial cluster
  - 50,000 servers shared by 115 VCs
  - One month period
  - Job submit time, size, etc
- Pick 6 VCs (two under-loaded, three full-loaded and one over-loaded) to assess the performance inconsistency

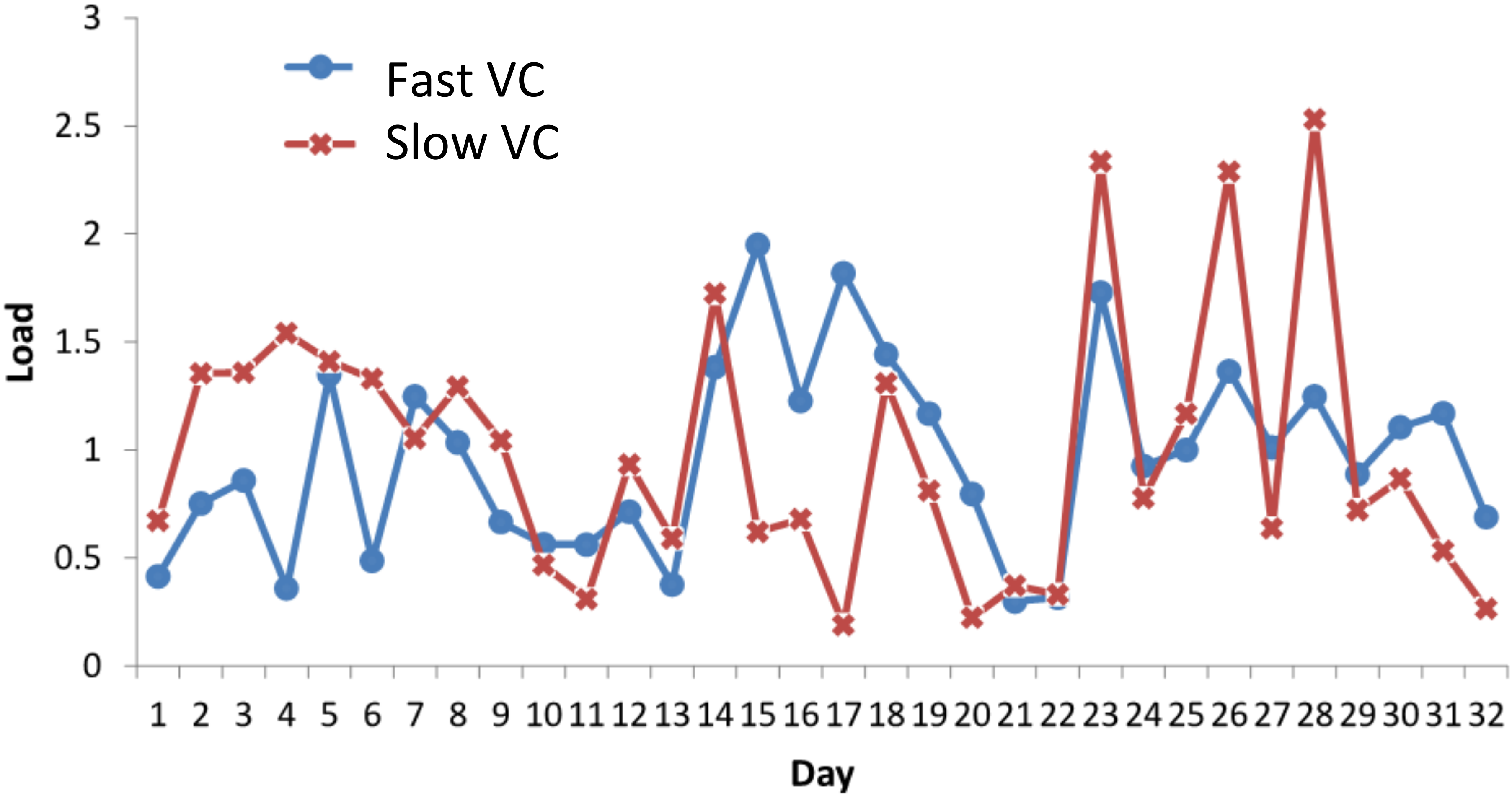
Overall load fluctuation



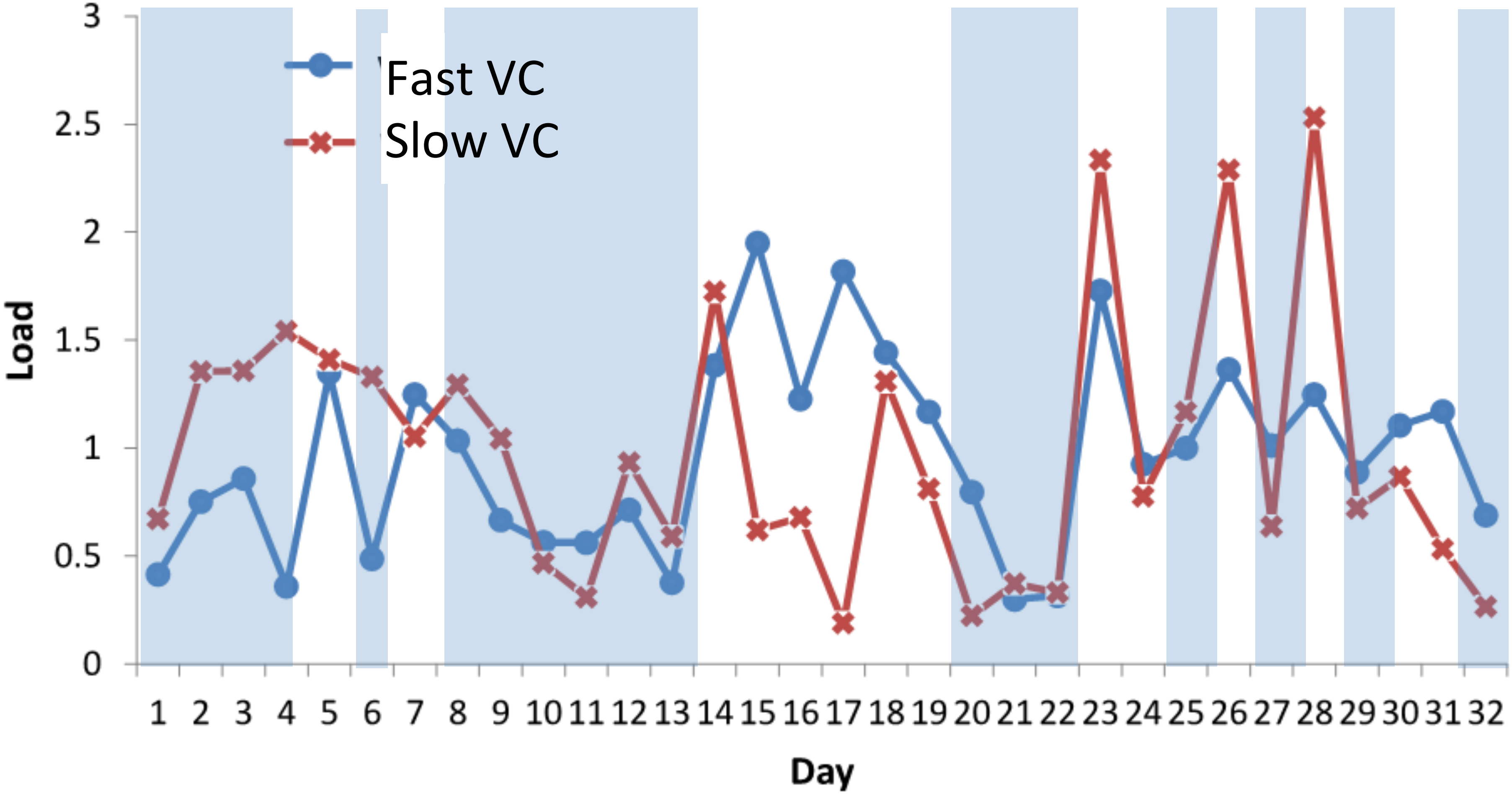
Overall VC performance



Load fluctuation

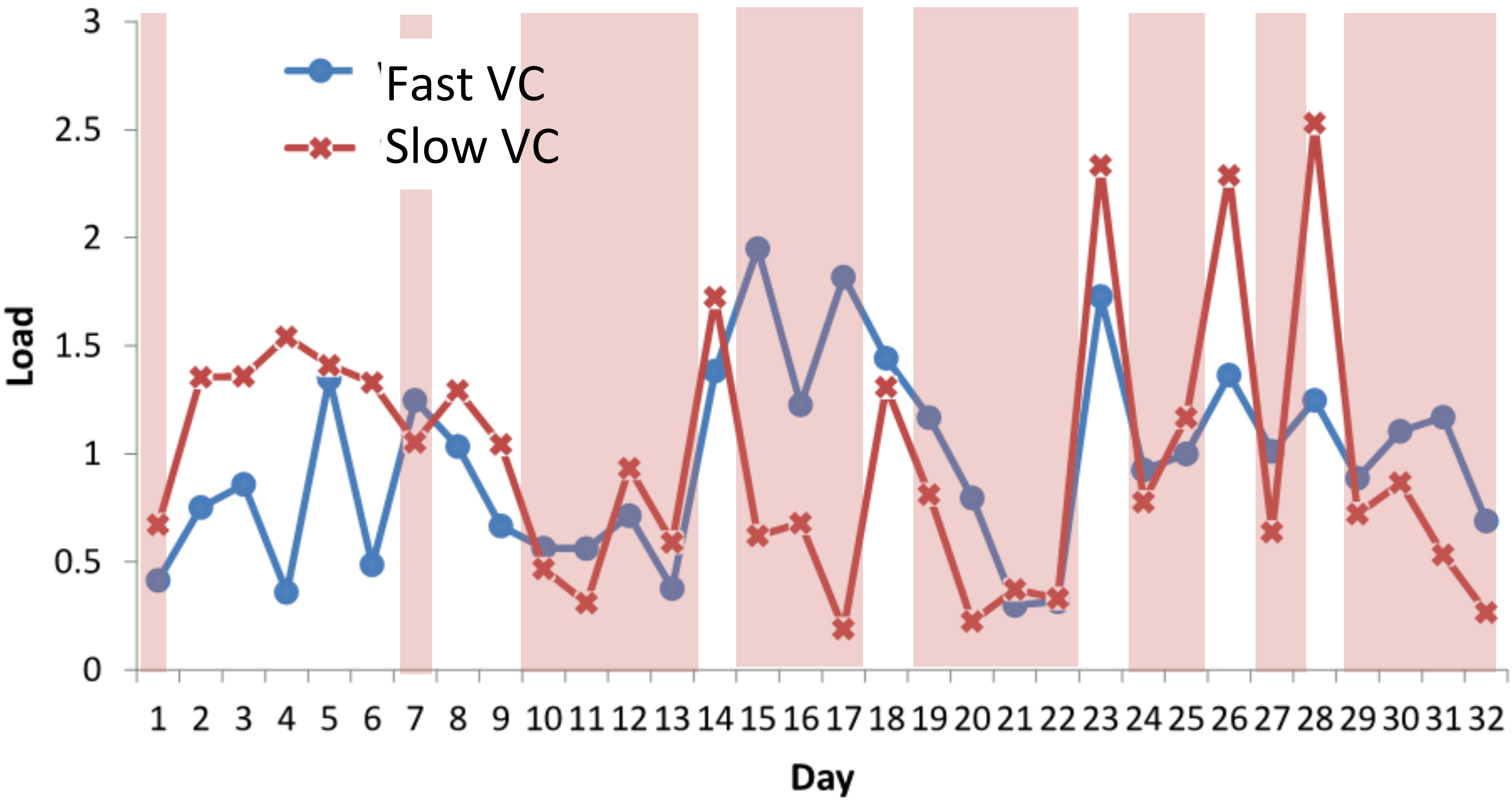


Fast VC: under-loaded days



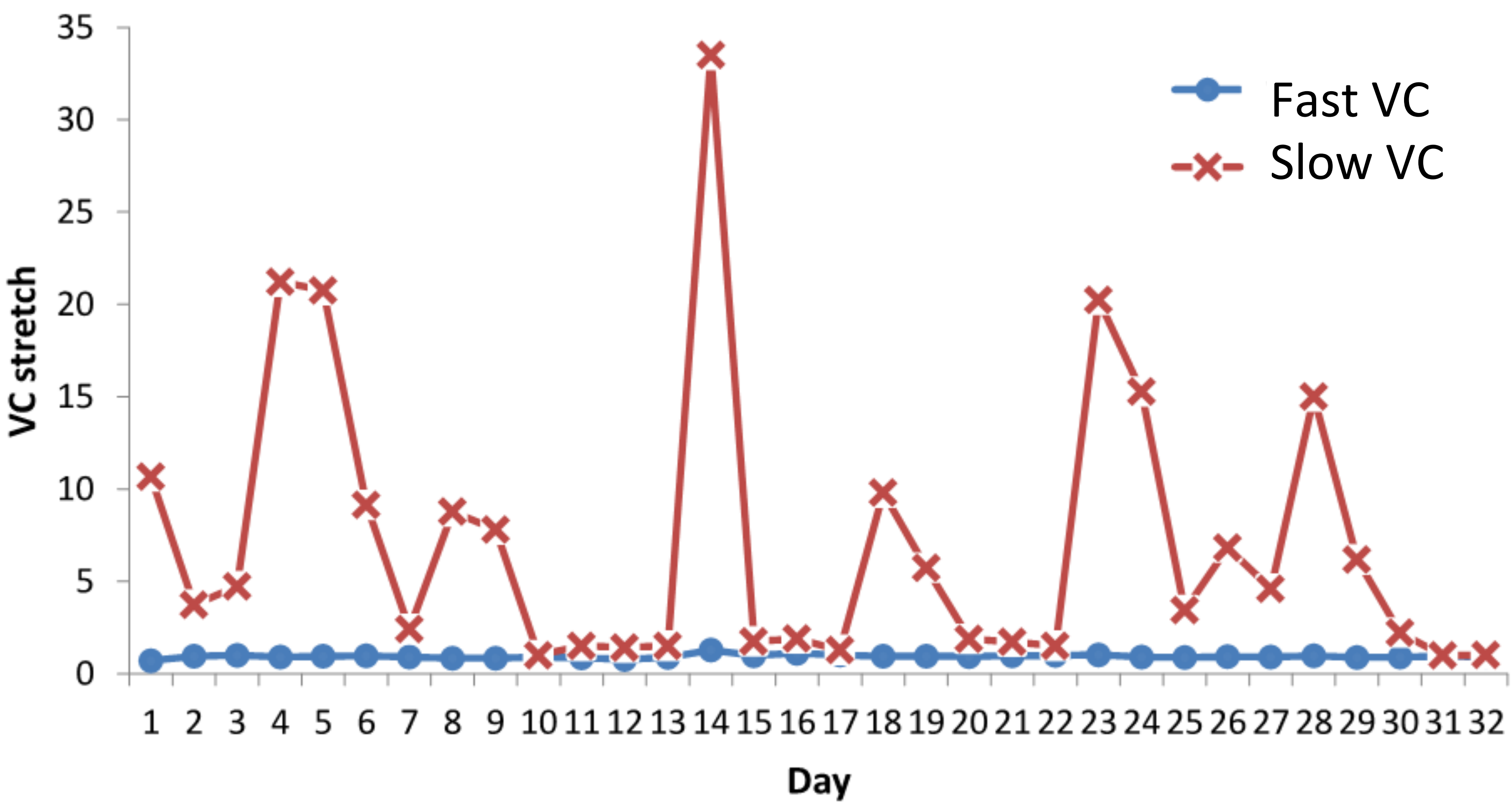
18 under-loaded days

Slow VC: under-loaded days



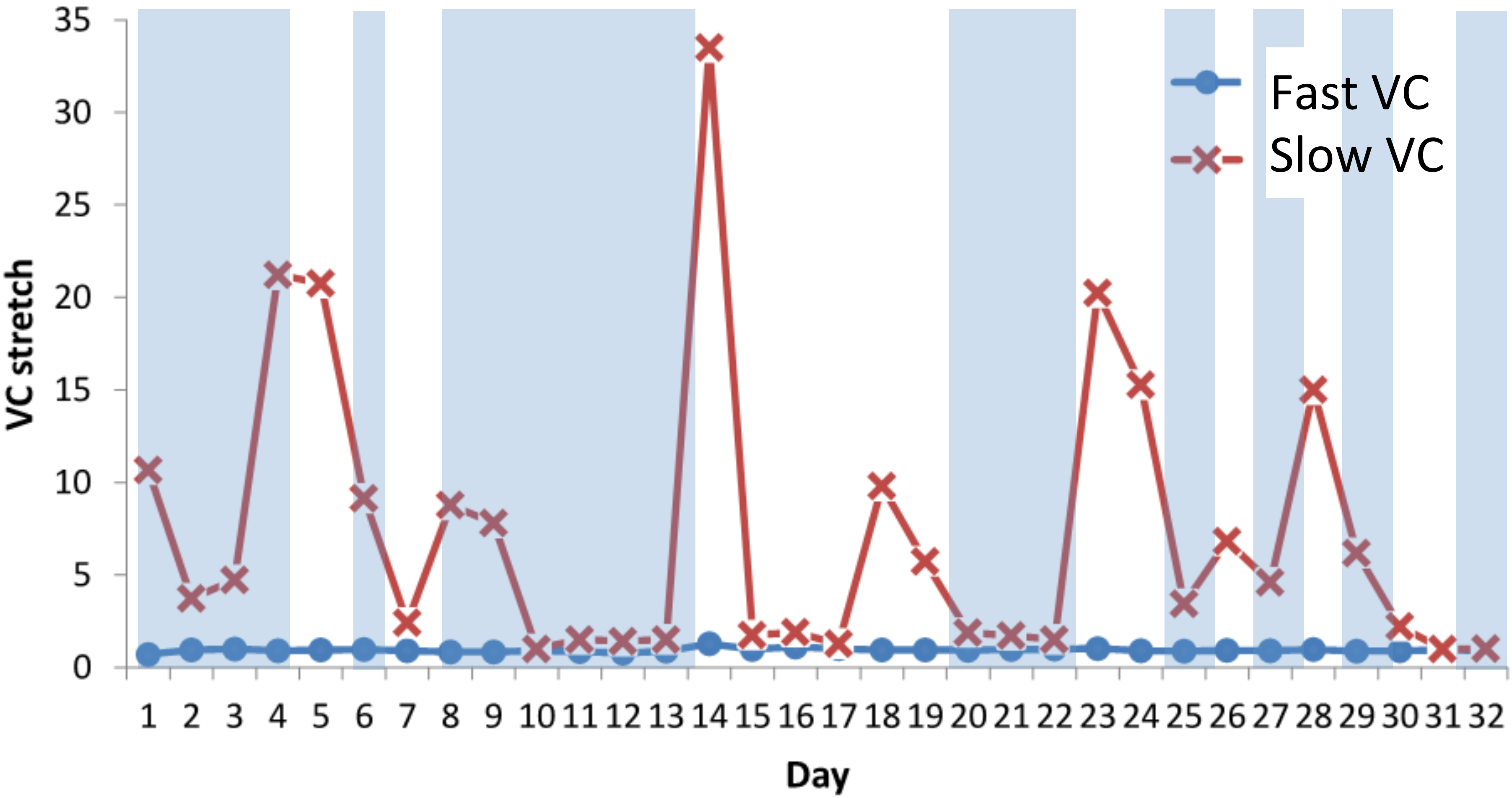
20 under-loaded days

Performance



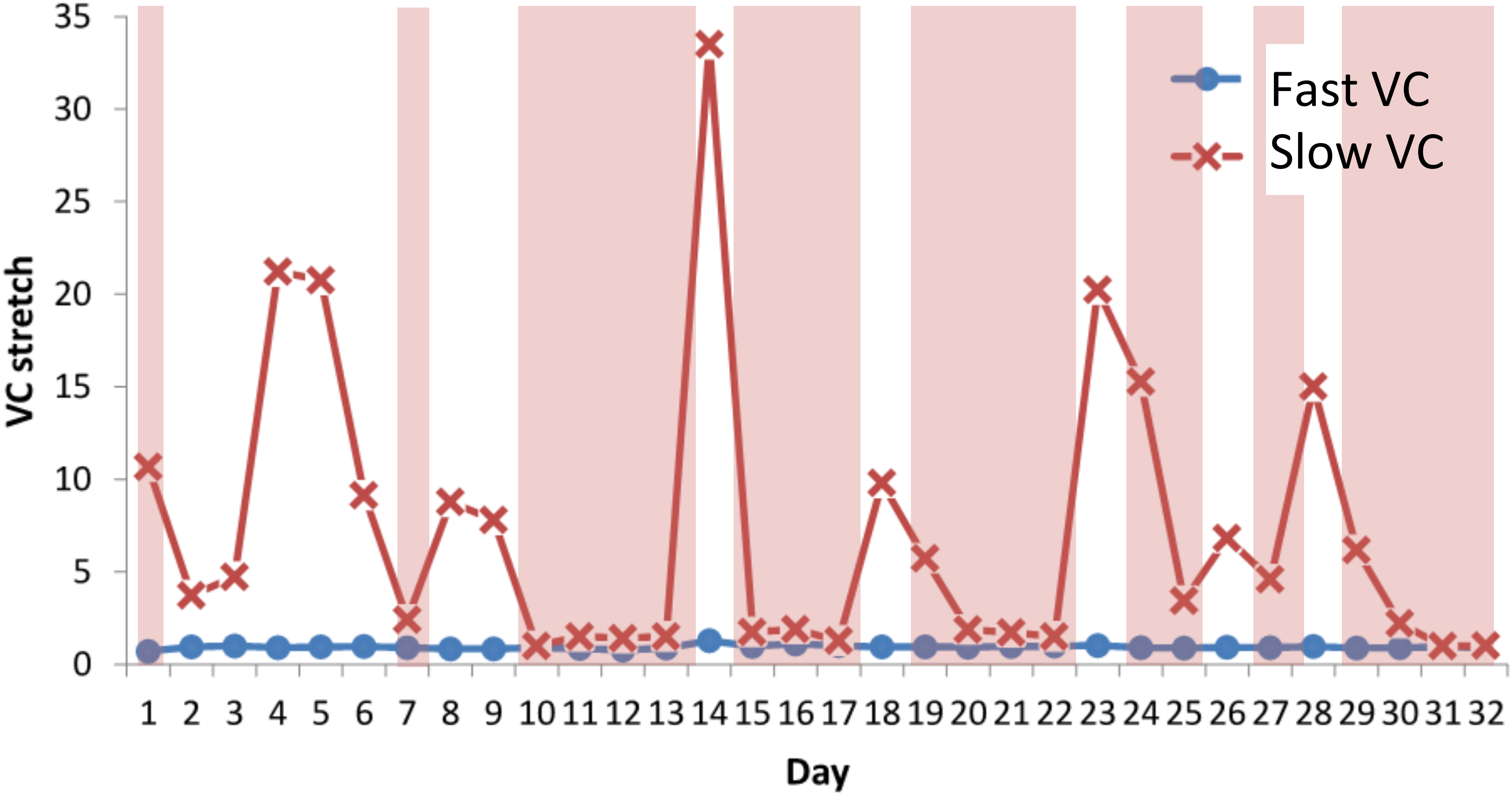
(b) The VC stretch. Lower is better.

Fast VC: Under-loaded day performance



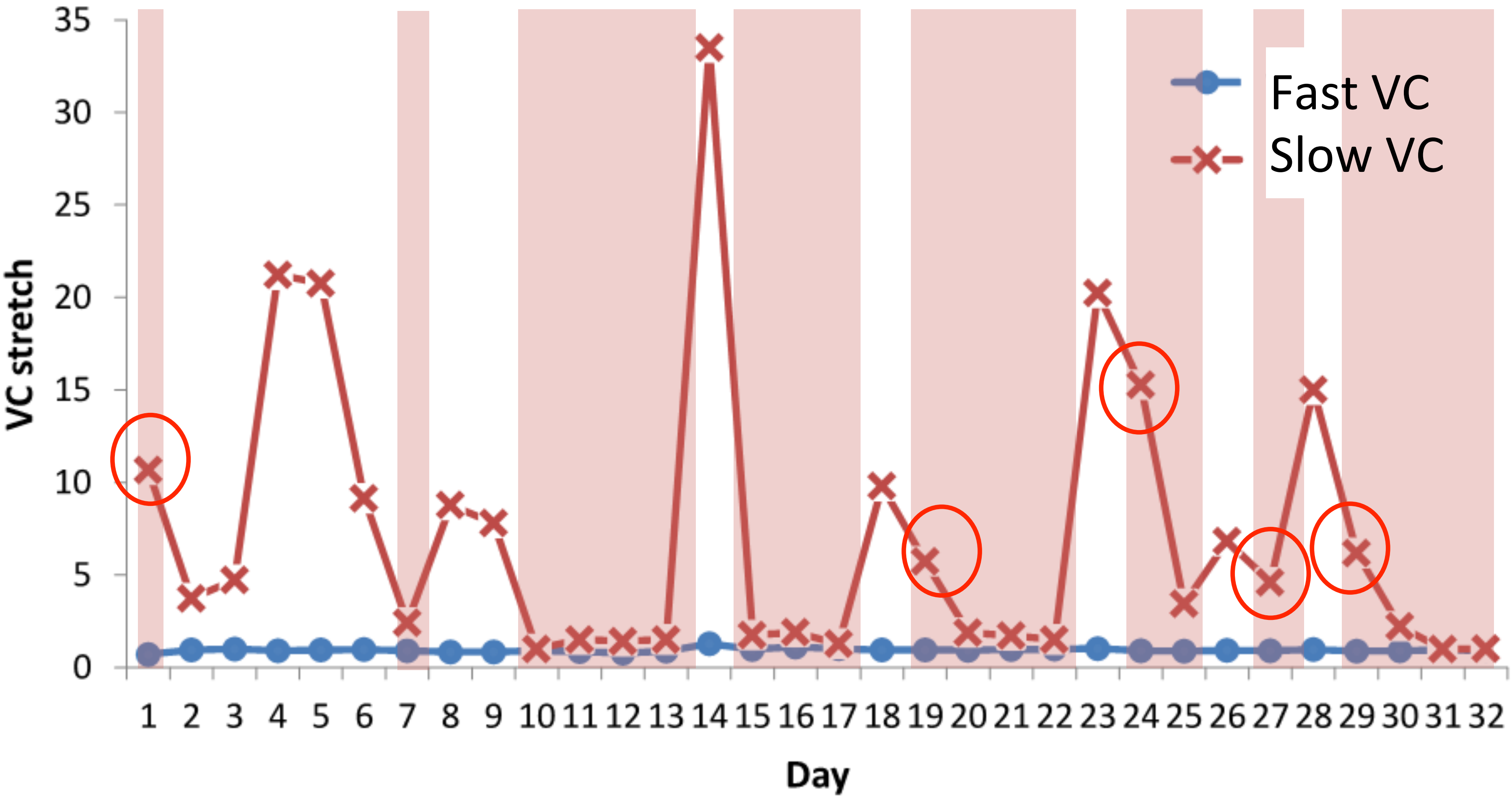
The fast VC performs optimally when it is under-loaded

Slow VC: Under-loaded day performance



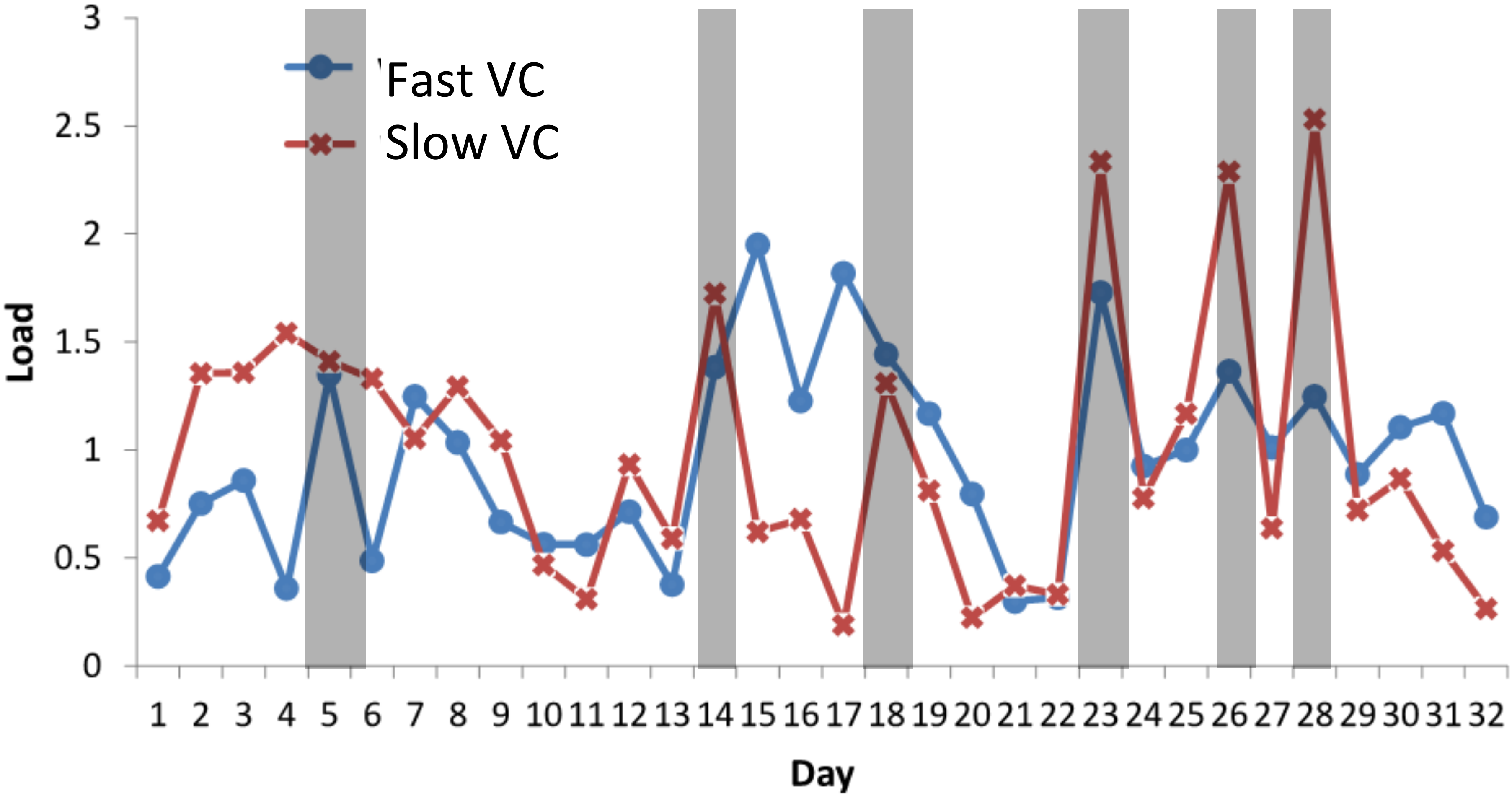
The slow VC performs optimally in most under-loaded days

Congestion



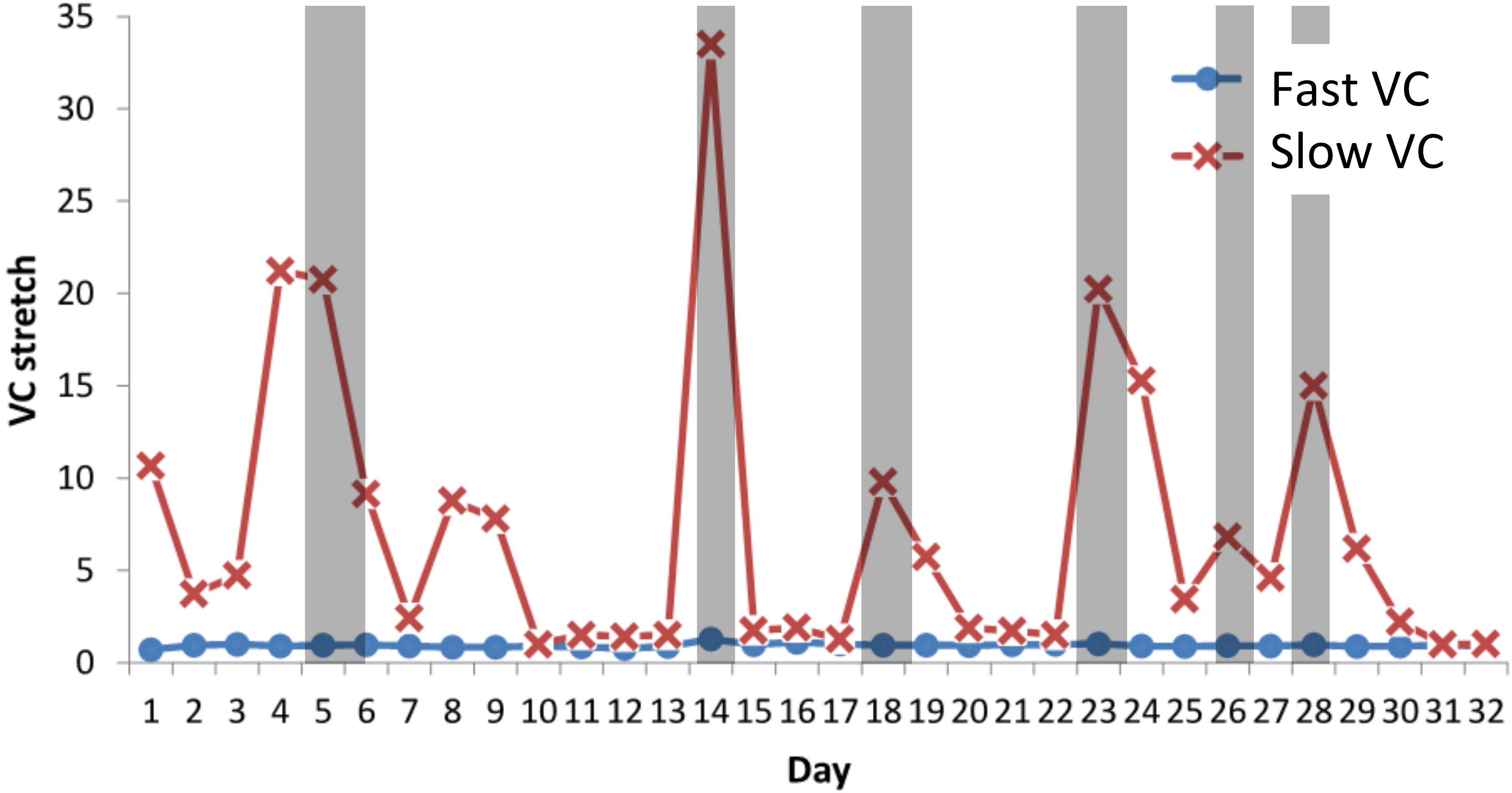
Congestion: a slow day may affect the next upcoming day

Contending days



Six contending days: both VCs are overloaded

Contending day performance



Slow VC loses totally to the fast VC

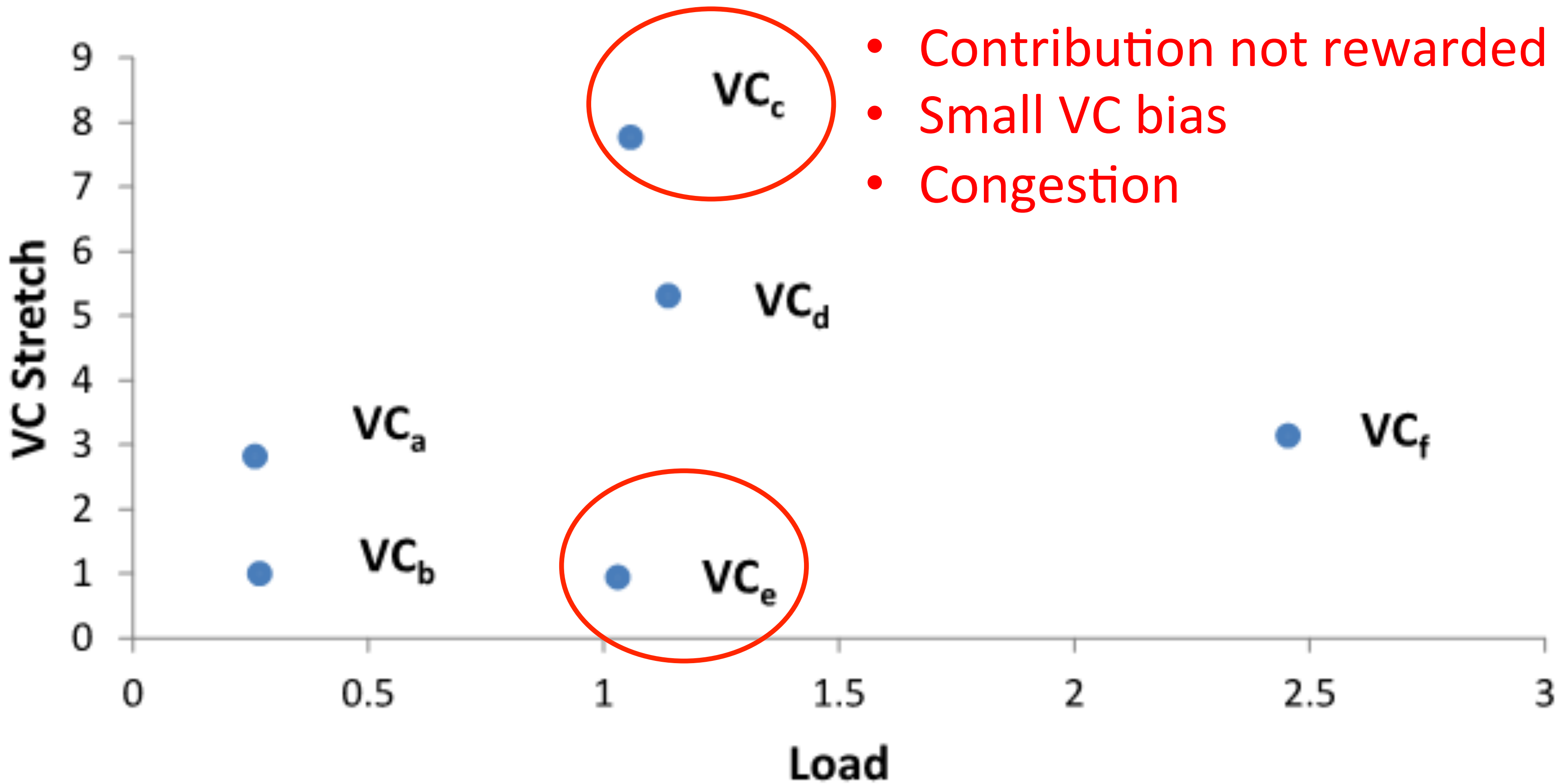
Summary

	Slow VC	Fast VC
Capacity	900	350
Load characteristic (both~=1)	20 under, 11 over (bursty)	18 under, 13 over (smooth)
Under-loaded day performance	- 14 days stretch~=1 - 6 day stretch >1 due to congestion	Stretch~=1

Summary (cont.)

	Slow VC	Fast VC
Capacity	900	350
Load characteristic (both~=1)	20 under, 11 over (bursty)	18 under, 13 over (smooth)
Under-loaded day performance	- 14 days stretch~=1 - 6 day stretch >1 due to congestion	Stretch~=1
(Six) Contending days	Stretch = 10 ~ 35 - contribution not rewarded - Small VC bias	Stretch~=1
(Five) Overloaded days	Stretch = 4 ~ 10 - Meets full-utilized days	Stretch ~=1

Summary



- Contribution not rewarded
- Small VC bias
- Congestion

Solutions?

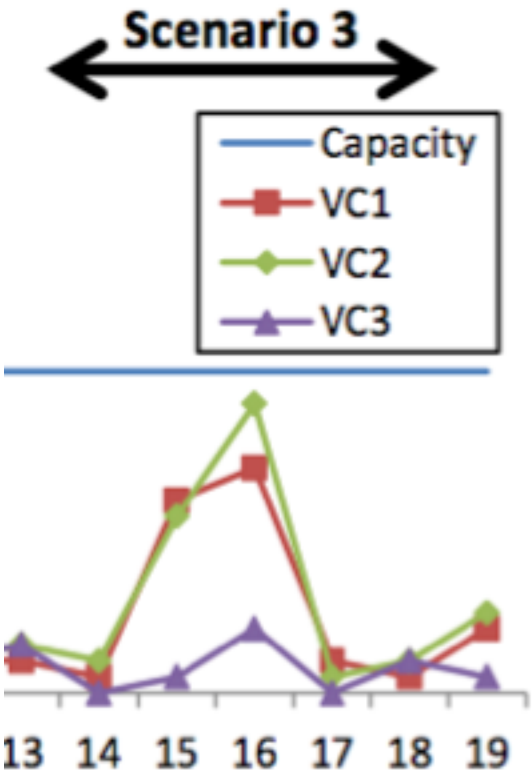
- Consider usage history
  - Idea: gain credits when contributing; lose credits when overusing
  - Higher credits more allocation
  - Deficit Round-Robin (DRR) for network switches
  - Xen credit scheduler
  - Lottery-based scheduler: accumulate “lottery” to increase the chances of wining more resources

A straightforward accounting method

- Open a “saving account” for free resource contribution
  - Contributing free resources -> gain credits
  - Overusing -> lose credits

- Allocate more VMs to VCs with higher credits

Challenges



- Give credits as long as VCs are underutilized
- Make promise for more future allocation
- Users further decrease load to gain more credits
- The system becomes even more underutilized

Challenges

- The counting scheme should provide “good” incentives of user behaviors
- We desire the users to
  - Contribute resources
  - Promote overall system utilization
  - Shape their workload and avoid peaks
- Straightforward scheme may not be enough!

Conclusions

- Show the performance inconsistency with Instantaneous fair schedulers
- Identify three causes
- Usage history should be considered when making scheduling decisions
- Credit counting should enforce fairness while providing incentives to promote overall utilization

McGill

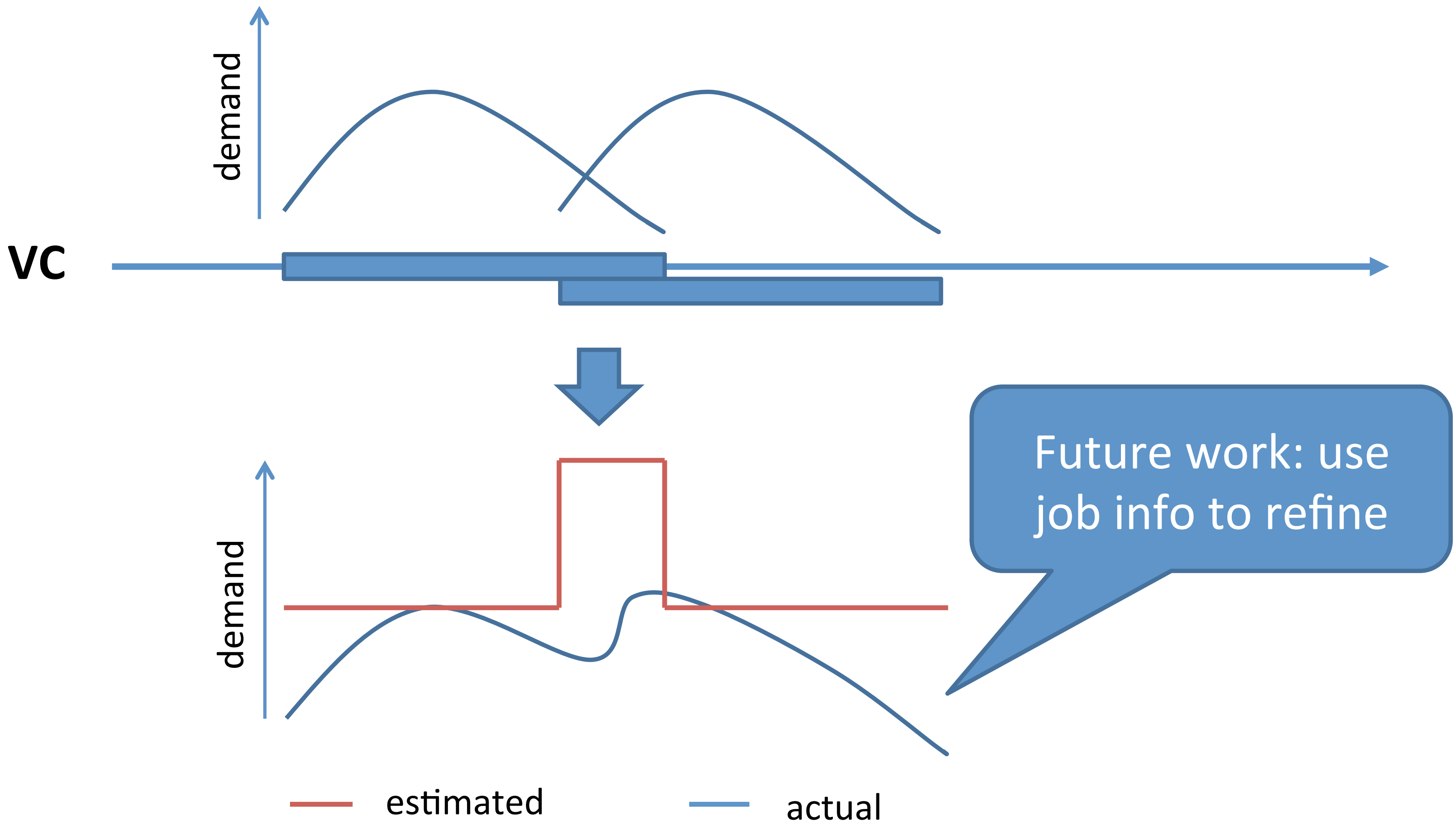
Microsoft Research

Performance Inconsistency in Large Scale Data Processing Clusters

Mingyuan Xia, Nan Zhu, Yuxiong He, Sameh Elnikety, Xue Liu

Backup

Parallelism estimation



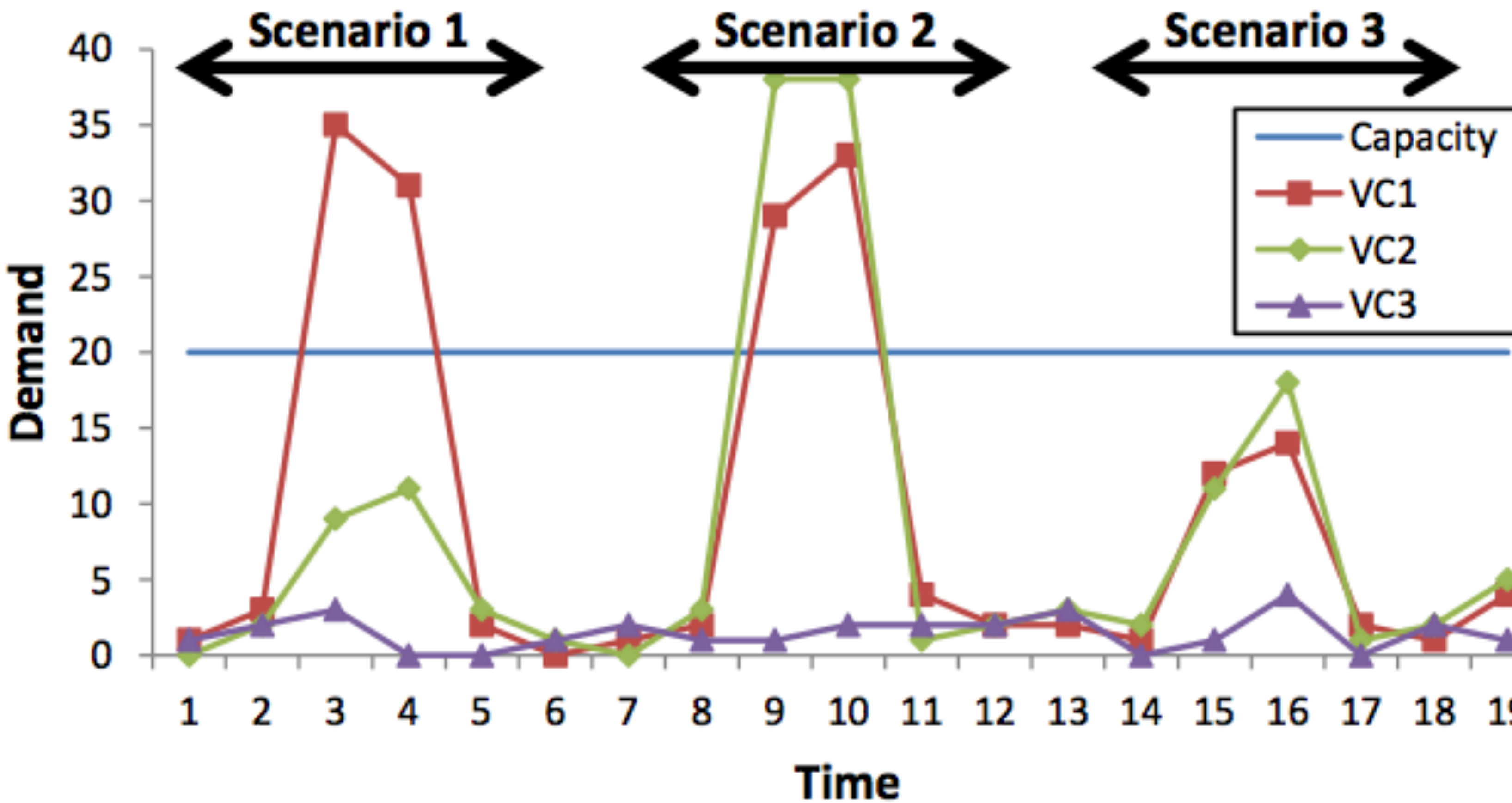
Short-term or long-term fairness

- MapReduce jobs are long (~hours), especially in large clusters like Cosmos
- Scheduling decisions are made on a minute basis
- Short-term fairness -> accumulated effects -> observe unfairness at job level

DRF scheduler

- Dominant Resource Fairness
  - Find the dominant resource type
  - Make MaxMin decision on that type (still only at a given time point)

Challenges



More credits lead to more allocation in the future

Burstiness impact

- MaxMin: If you peak meets others' peak you lose
- Ideal long term fairness: you will be treat better

Dynamic joining/leaving users