OStrich: Fair Scheduler for Burst Submissions of Parallel Jobs

Krzysztof Rzadca
Institute of Informatics,
University of Warsaw, Poland

joint work with:
Filip Skalski (U Warsaw / Google)

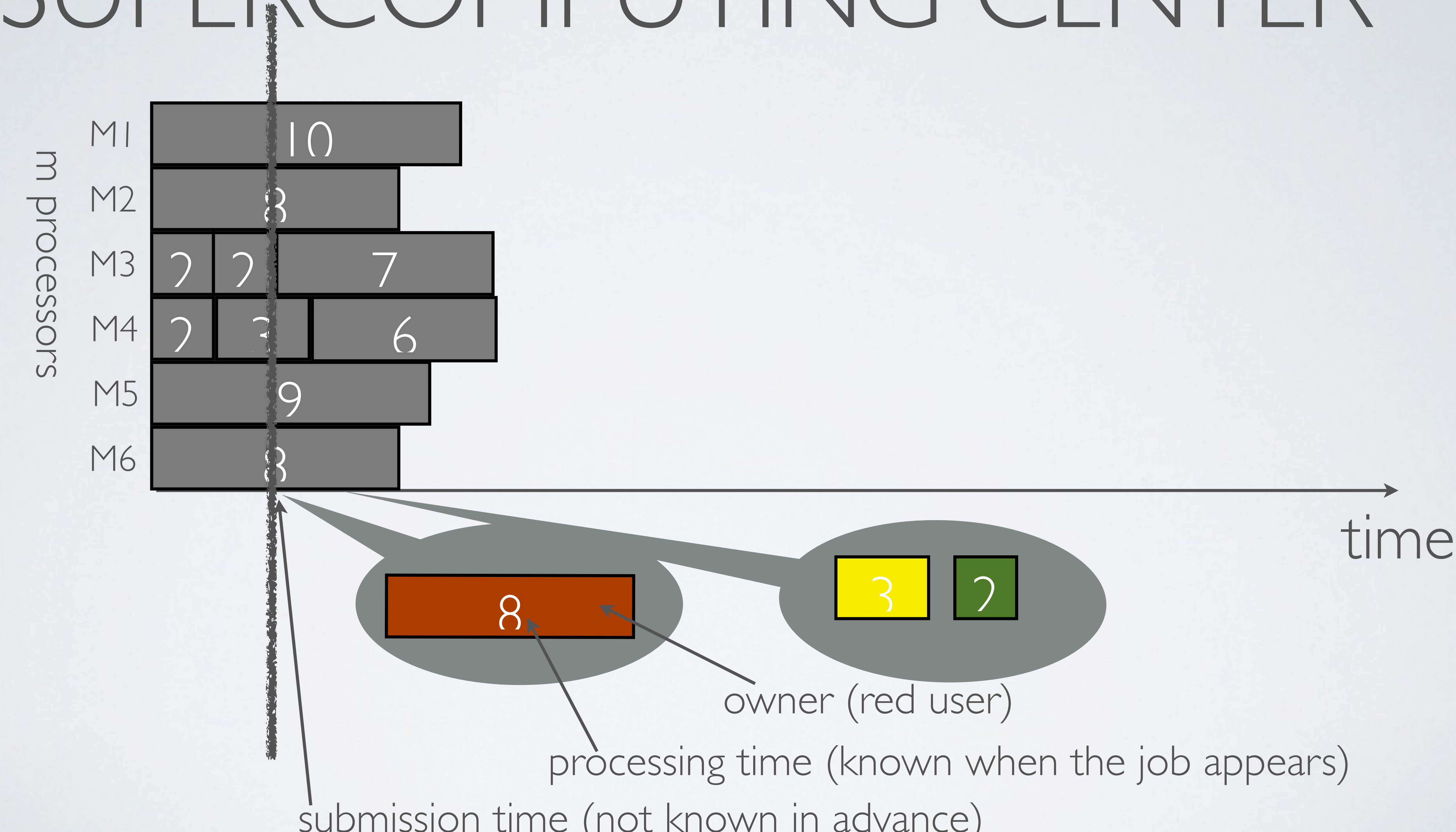
based on work with:
Vinicius Pinheiro (Grenoble)
Denis Trystram (Grenoble)

http://www.flickr.com/photos/bobjagendorf/3456836120/

KEY MESSAGE: A FAIR, MULTIUSER ONLINE SCHEDULING ALGORITHM

- Online problem with **multiple users** sharing a supercomputer
- Workload composed of **campaigns** (~job arrays): jobs independent to execute; the owner wants to finish all jobs as soon as possible
- OStrich: an algorithm with **a guarantee on worst-case slowdown (stretch)** for each user (OStrich ~ per-User Stretch)
- The **slowdown depends on the total number of users**, and not the total system load
- Implementation as a **SLURM scheduler used in a production cluster**

MODEL: A TYPICAL SUPERCOMPUTING CENTER



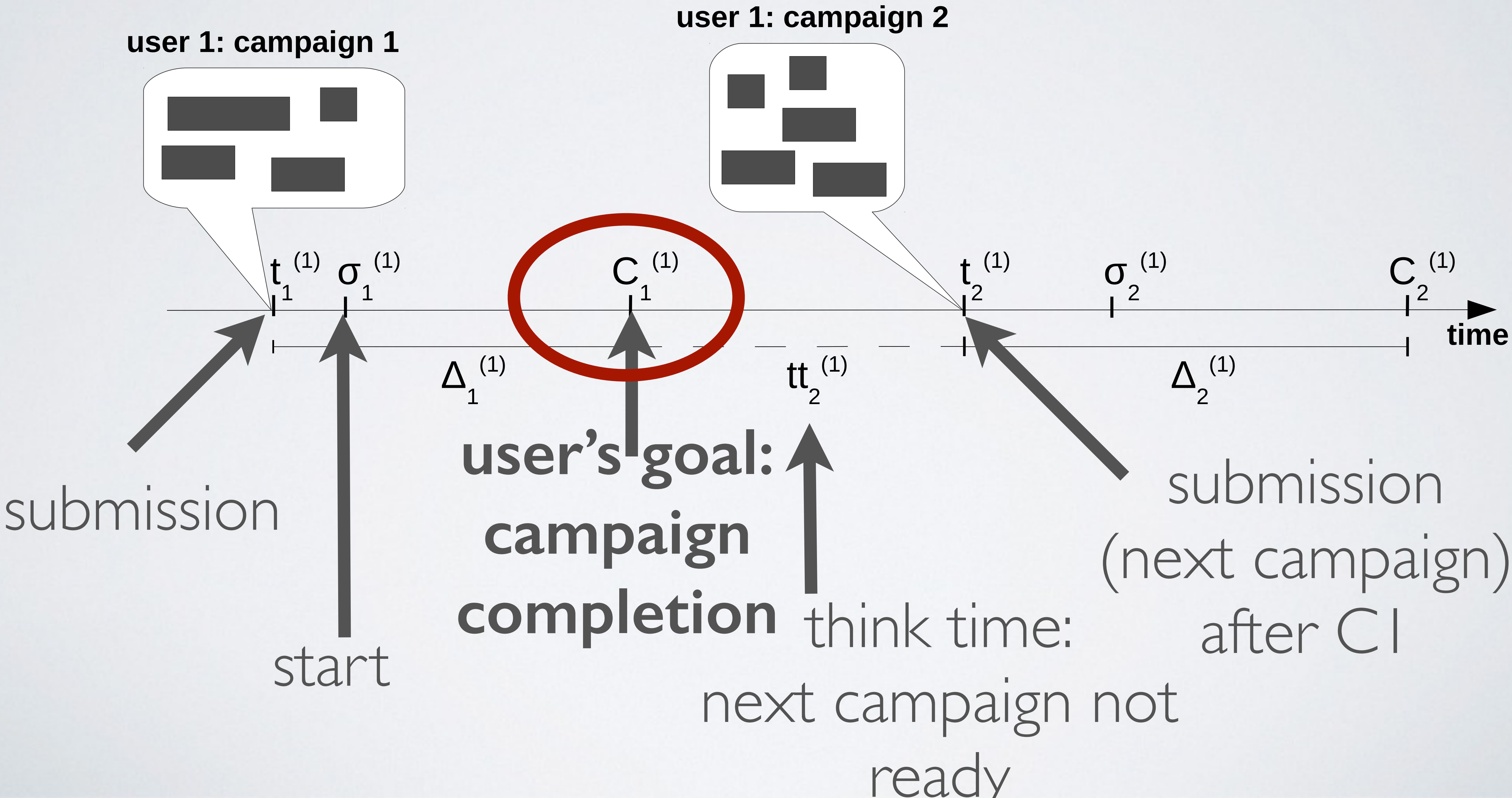
WHY CAMPAIGNS?

- Modern applications submit many related computing jobs
  - Map/Reduce
  - parameter sweep workflows
- SLURM makes such submissions easier by job arrays (max job array size increased to 1M, so it's useful)
- But cluster schedulers treat such jobs as independent

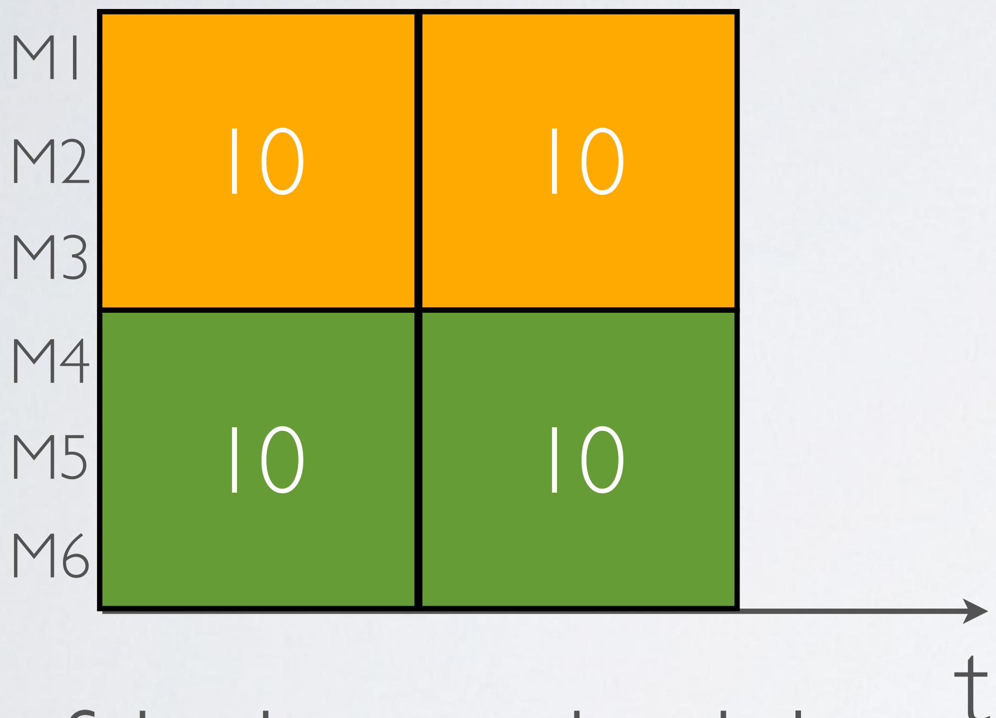
WHY A WORST-CASE BOUND FOR EACH USER?

- Many policies based on First-Come-First-Served
- New jobs are put at the end of the queue
- Thus, users with large workloads slow down everyone else
- Hard to manage partial solutions:
  - Limits on number of jobs in the queue,
  - Karma points, priority queues, etc.
  - Fair-share

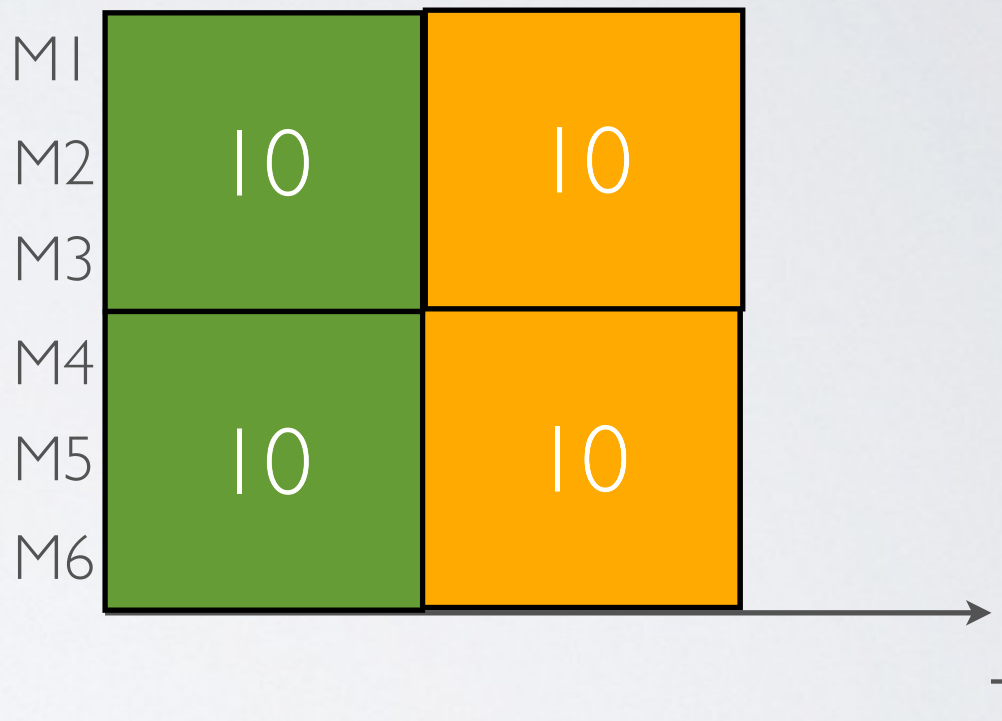
A CAMPAIGN: A BAG OF INDEPENDENT TASKS



PRINCIPLE OF THE ALGORITHM: PARETO-OPTIMALITY



a fair-share schedule

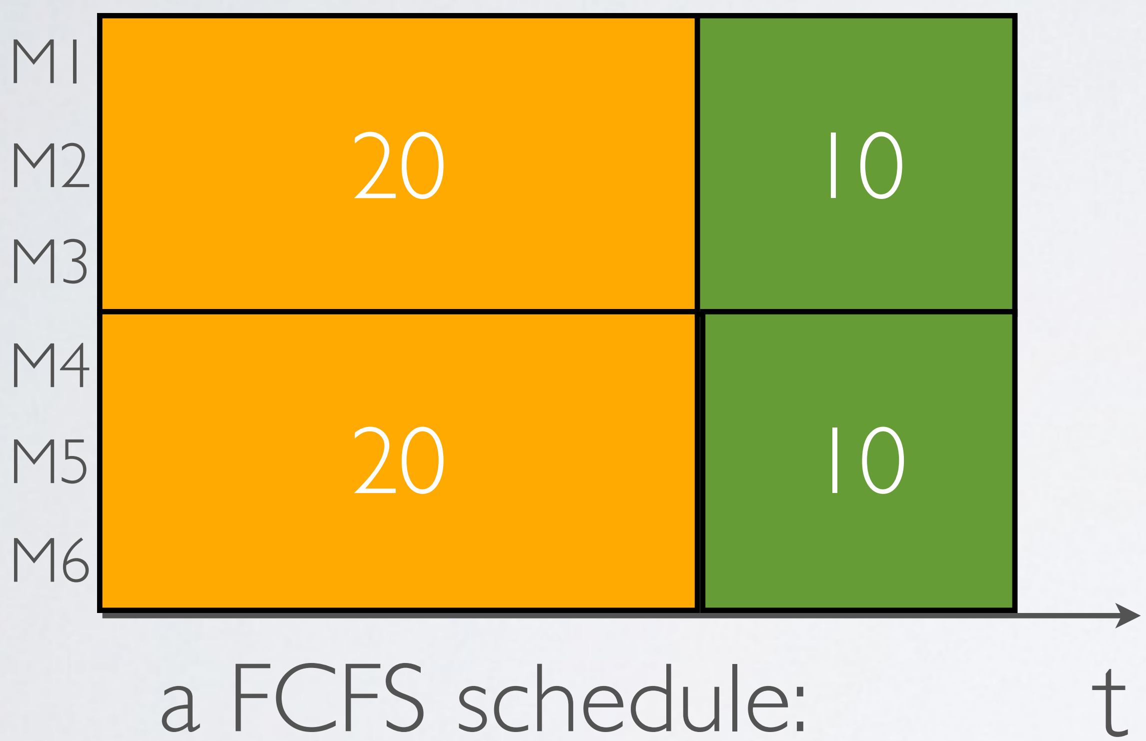


a Pareto-optimal schedule

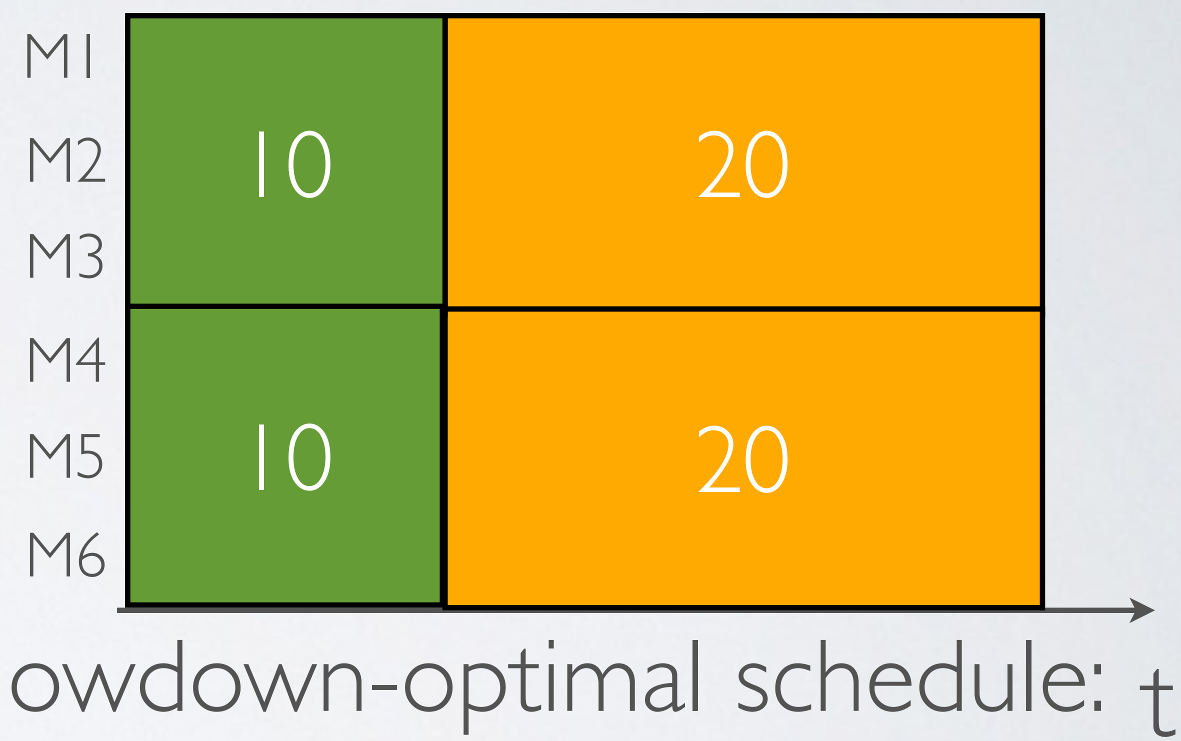
completion times:
(20,20)

completion times:
(10,20)

PRINCIPLE OF THE ALGORITHM: OPTIMIZE SLOWDOWN (BUT NO STARVATION)



a FCFS schedule:

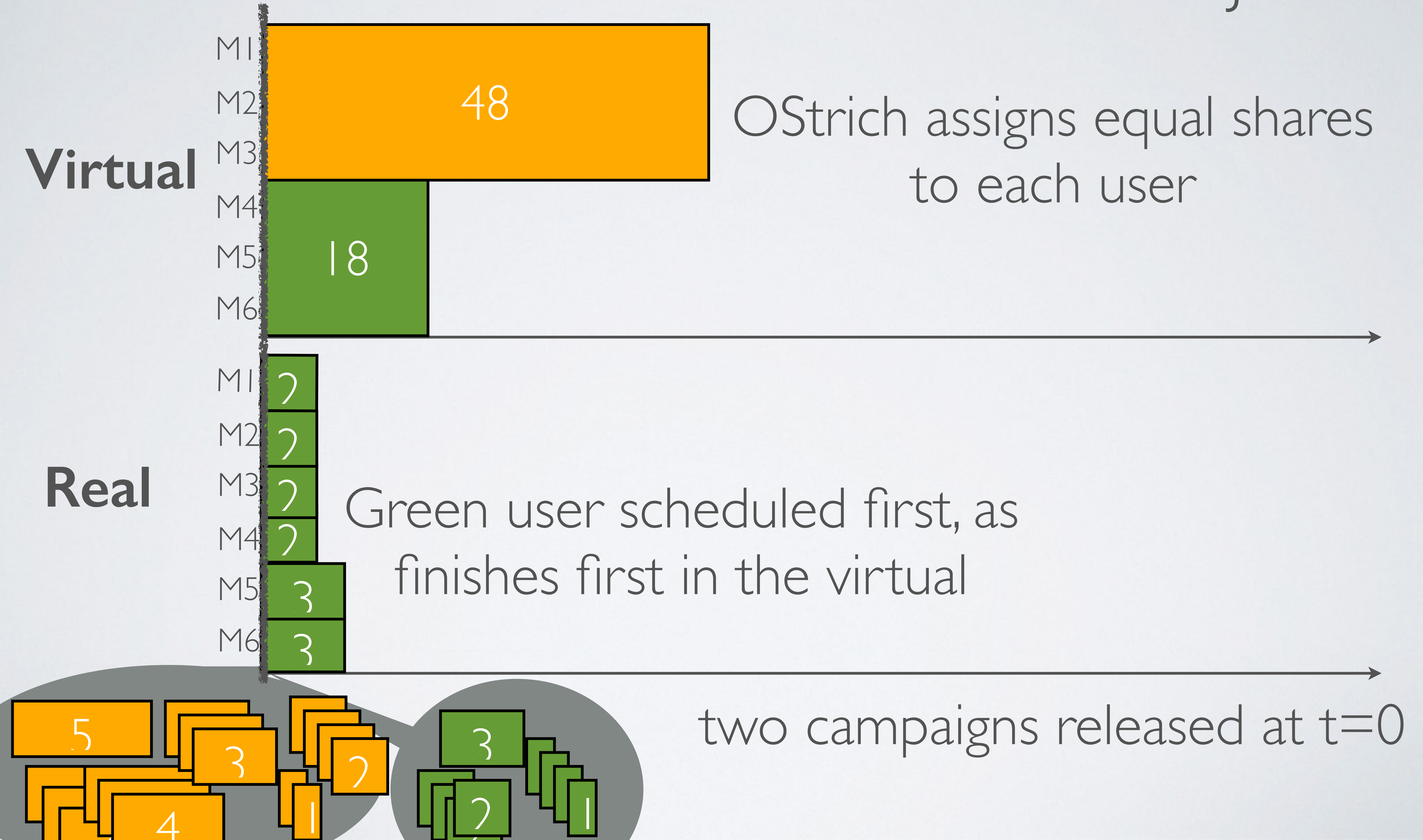


a slowdown-optimal schedule:

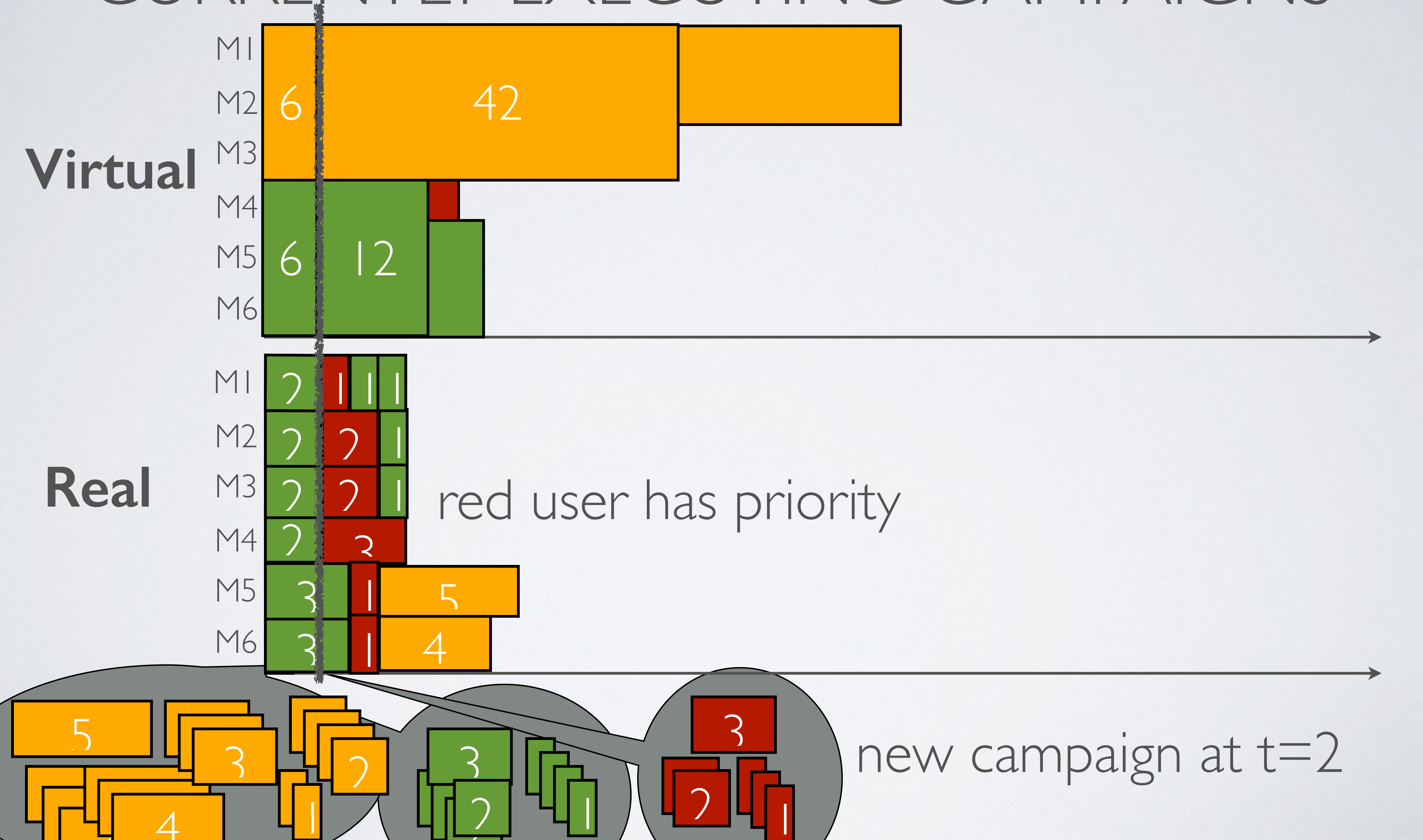
completion (30,20)
slowdown (3,1)

completion (10,30)
slowdown (1,3/2)

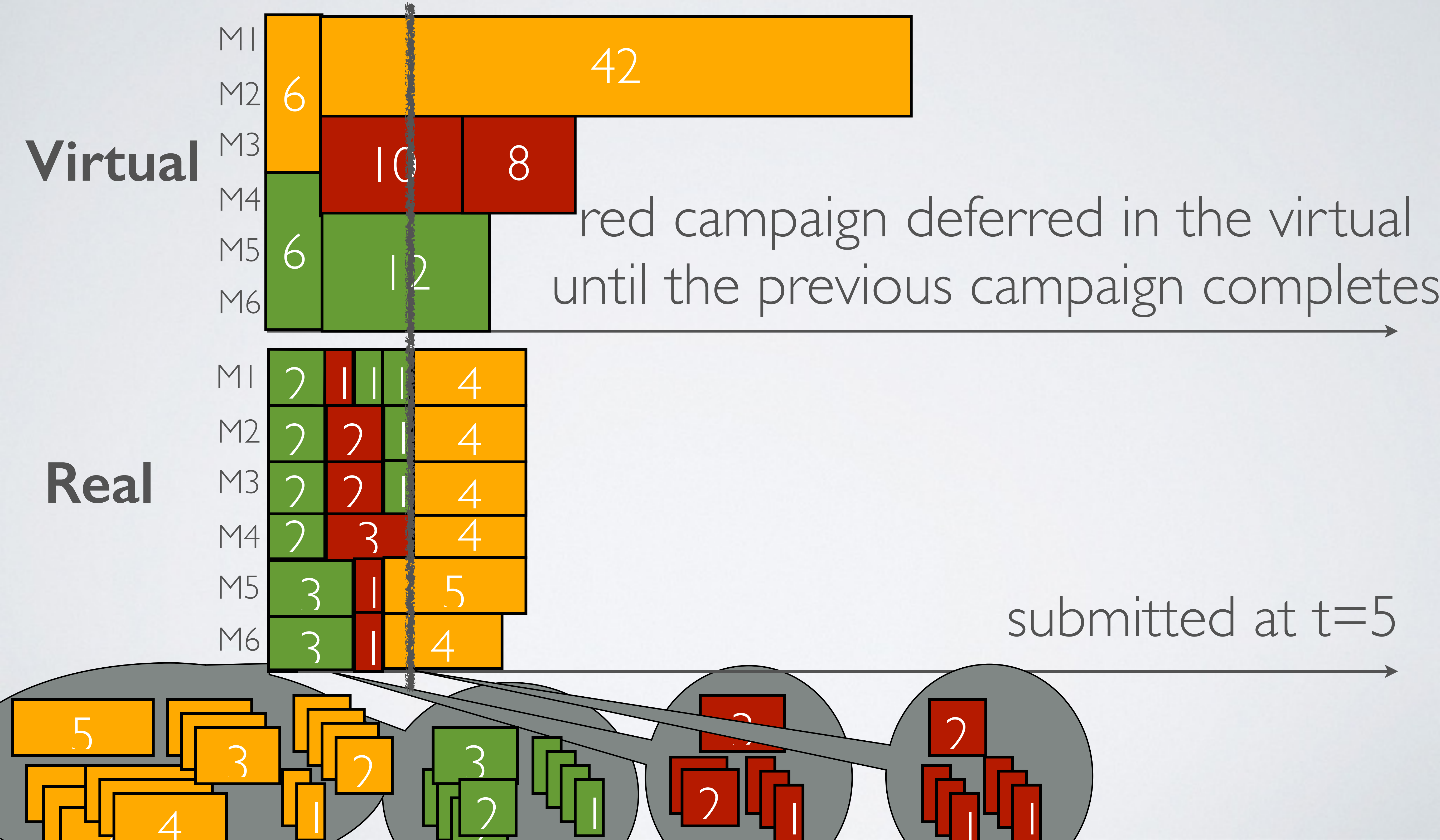
OSTRICH ALGORITHM: A VIRTUAL FAIR-SHARE SCHEDULE DEFINES PRIORITIES FOR CHOOSING JOBS



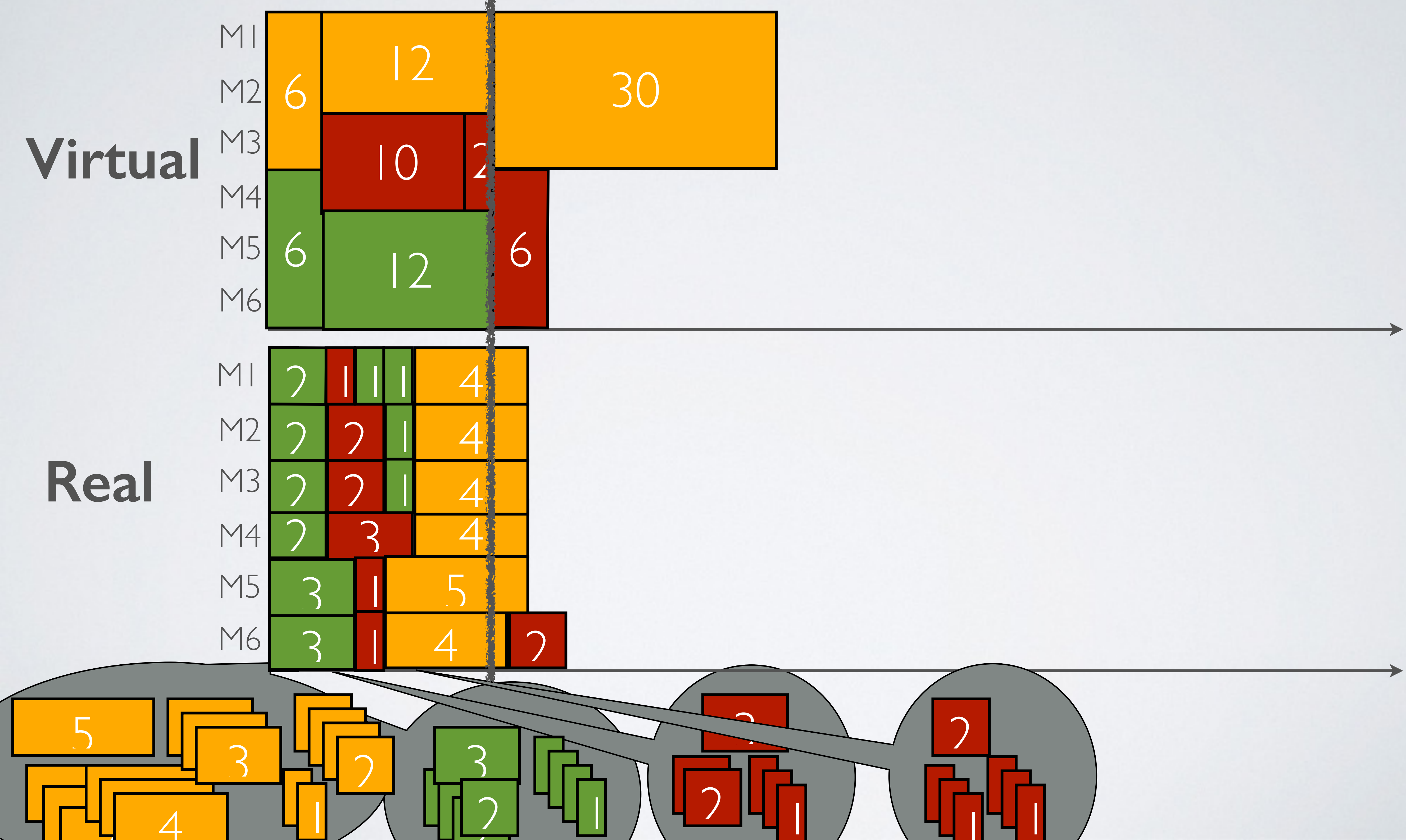
OSTRICH ALGORITHM: NEW SUBMISSIONS "PREEMPT" CURRENTLY EXECUTING CAMPAIGNS



OSTRICH ALGORITHM: NEXT CAMPAIGN DEFERRED UNTIL PREV CAMPAIGN VIRTUAL COMPLETION



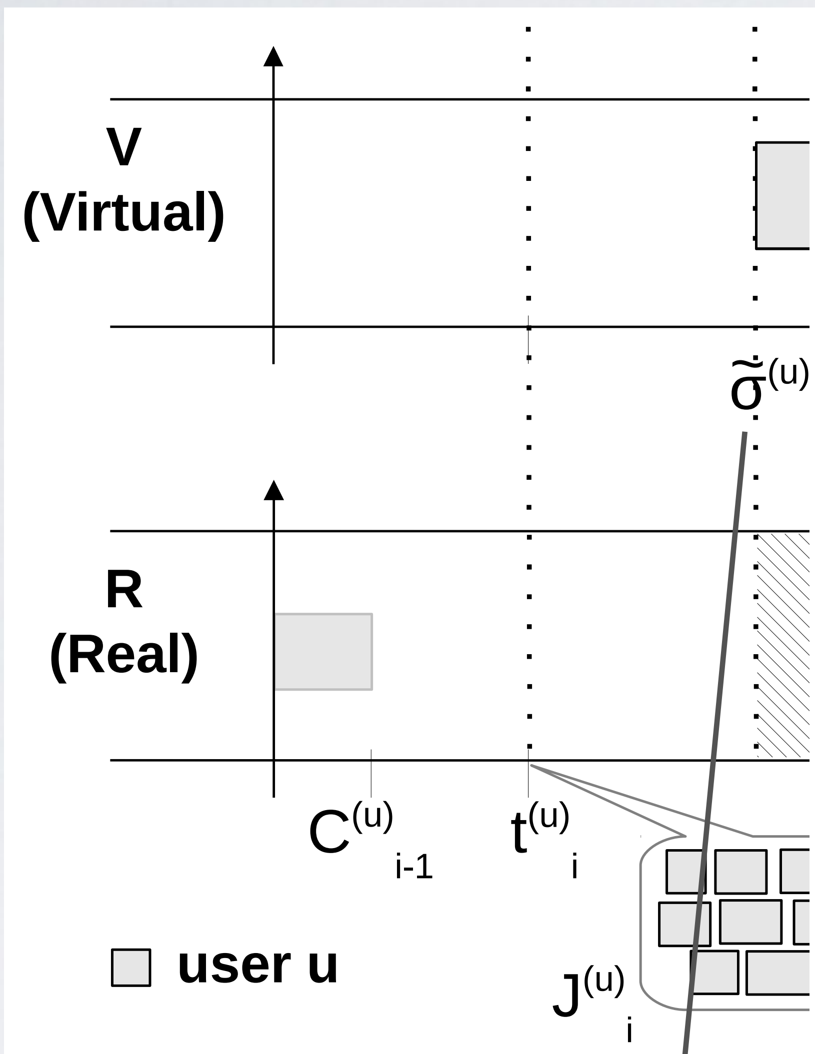
OSTRICH ALGORITHM: NEXT CAMPAIGN DEFERRED UNTIL PREV CAMPAIGN VIRTUAL COMPLETION



http://www.supercoloring.com/

SOME PROOFS?

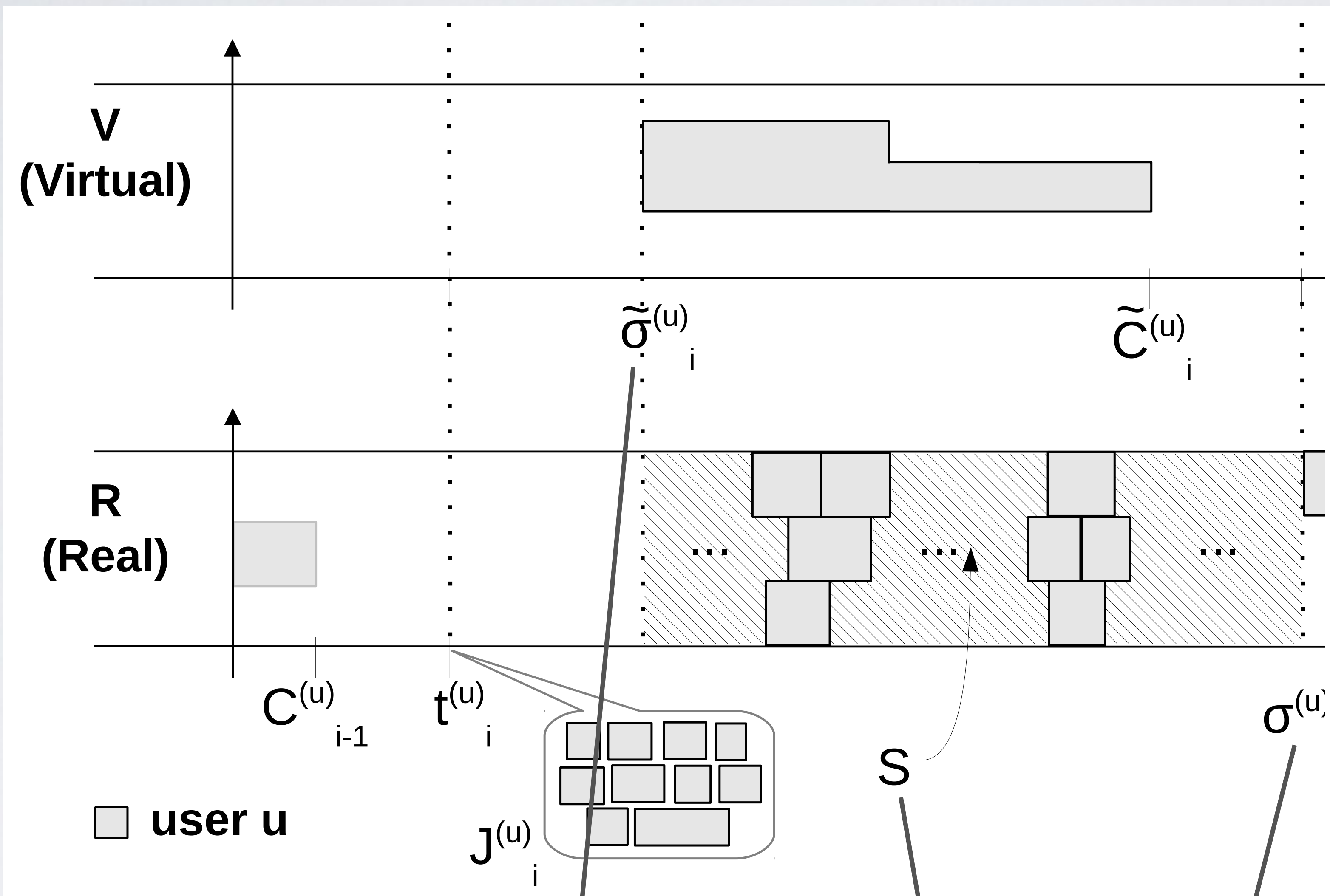
AN UPPER BOUND ON THE CAMPAIGN'S COMPLETION TIME



$$C_{i,q} \leq t_i^{(u)} + k\frac{W_{i-1}^{(u)}}{m} + \dots$$

wait until the prev campaign completes in virtual

AN UPPER BOUND ON THE CAMPAIGN'S COMPLETION TIME

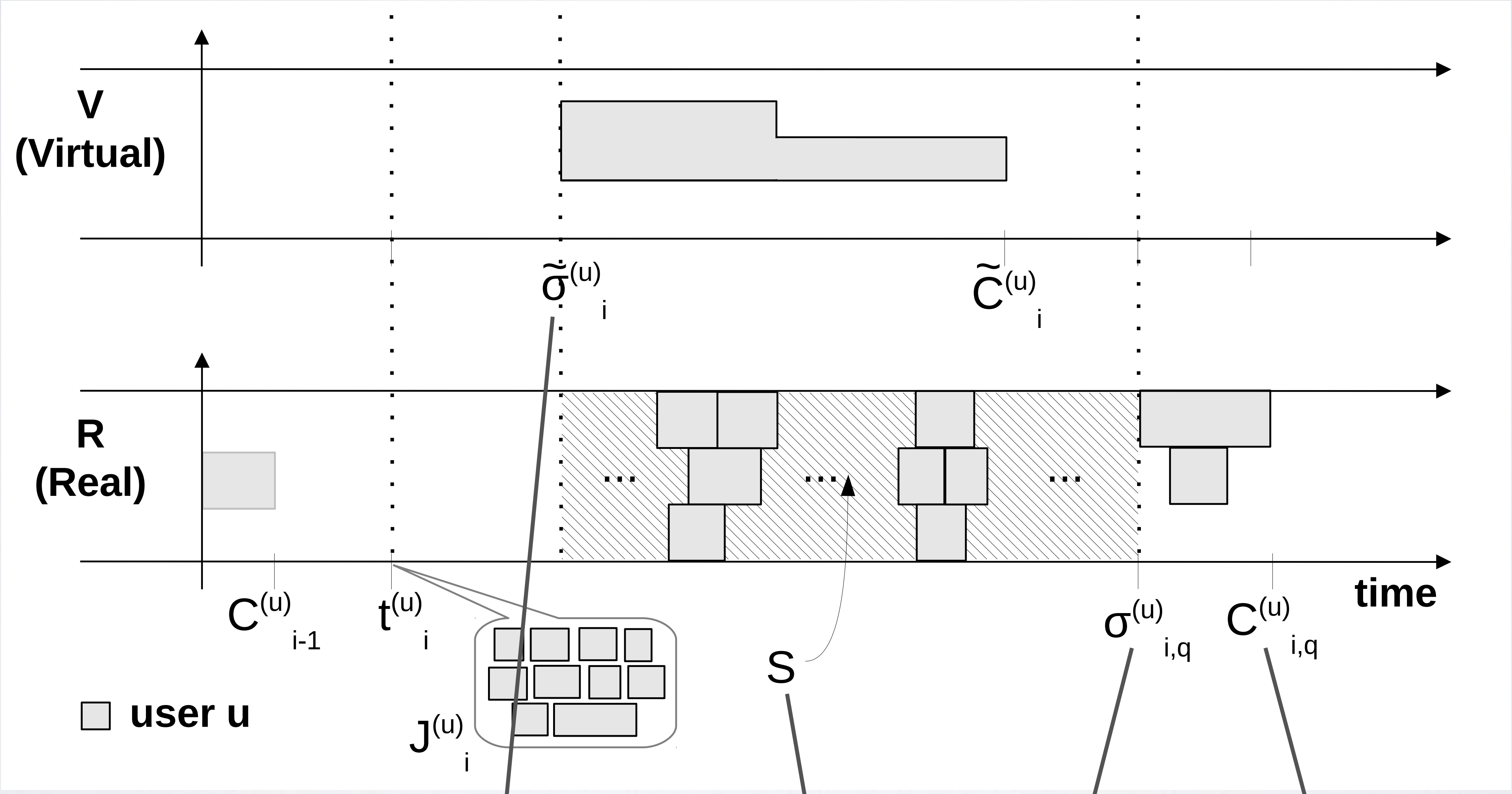


$$C_{i,q} \leq t_i^{(u)} + k\frac{W_{i-1}^{(u)}}{m} + p_{\max} + (k-1)\frac{W_i^{(u)}}{m} + p_{\max} + \frac{W_i}{m}$$

wait until the prev campaign completes in virtual

upper bound on the surface that can preempt while campaign is executing in virtual

AN UPPER BOUND ON THE CAMPAIGN'S COMPLETION TIME



$$C_{i,q} \leq t_i^{(u)} + k\frac{W_{i-1}^{(u)}}{m} + p_{\max} + (k-1)\frac{W_i^{(u)}}{m} + p_{\max} + \frac{W_i^{(u)}}{m} + p_{\max}^{(u)}$$

wait until the prev campaign completes in virtual

upper bound on the surface that can preempt while campaign is executing in virtual

standard upper bounds for the current campaign executing on all resources

EACH CAMPAIGN'S SLOWDOWN IS BOUNDED

- campaign slowdown: flow time weighted by the surface

$$D_i^{(u)} = \frac{C_i^{(u)} - t_i^{(u)}}{W_i^{(u)}}$$

- OStrich guarantee:

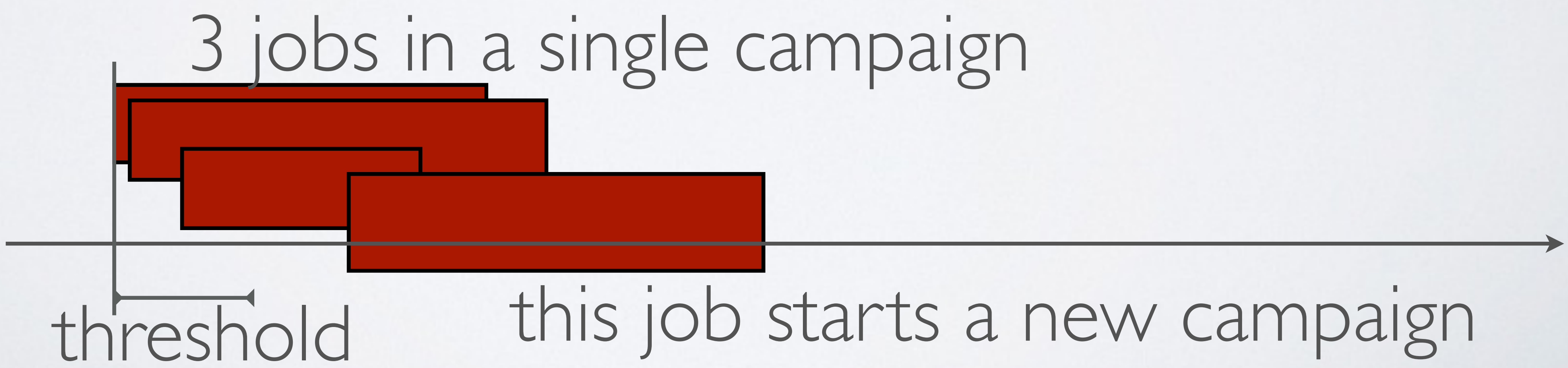
$$D_i^{(u)} \in O(k(1 + \frac{W_{i-1}^{(u)}}{W_i^{(u)}}))$$

- k is the number of active users
- we treat pmax as constant (and small compared to campaign's surface)

IMPLEMENTATION IN SLURM

FROM THEORY TO SLURM

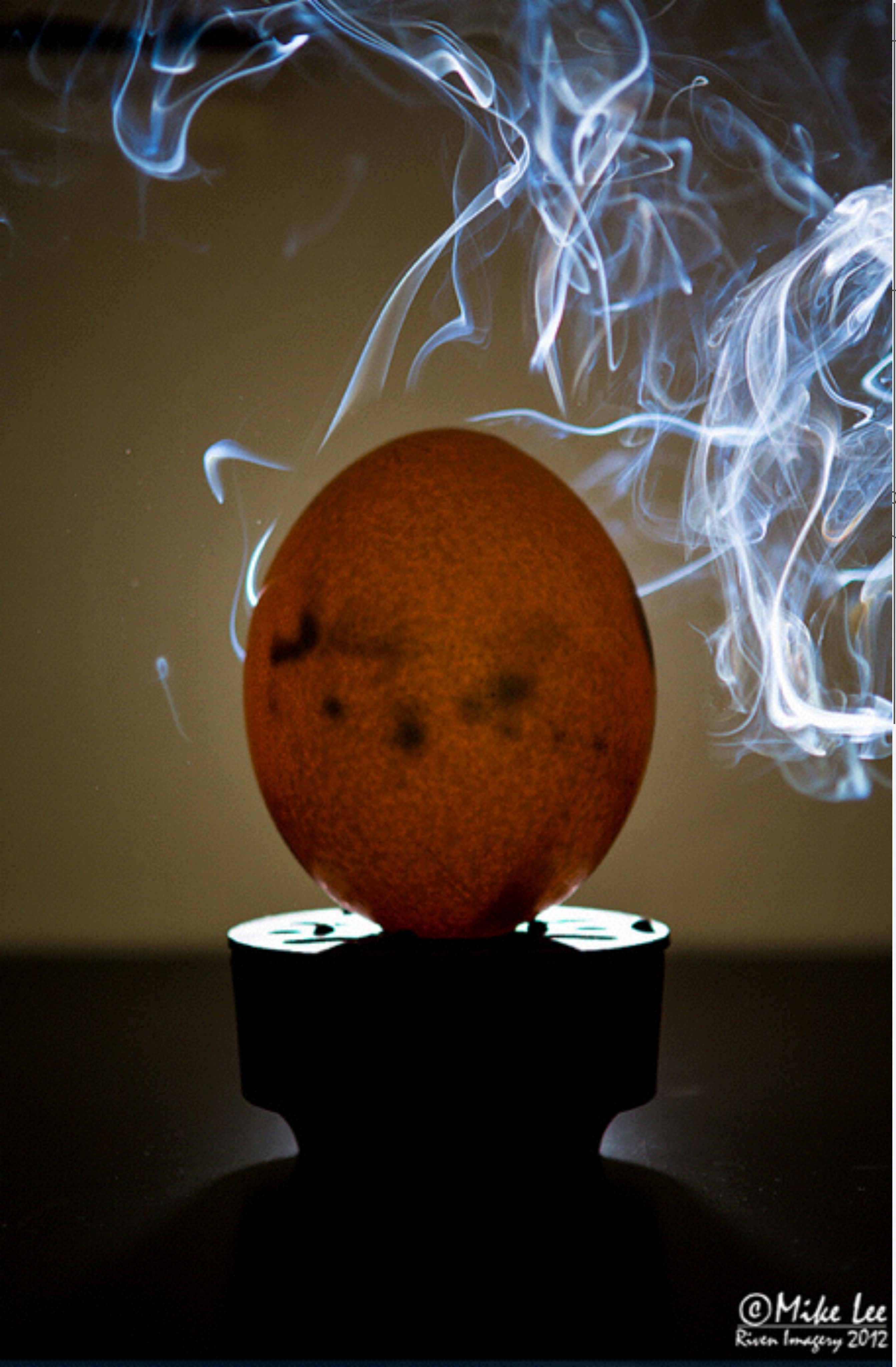
- fixed reservations: as idle time
- partitions: as (perhaps overlapping) sets of processors
- users' estimates are imprecise: simple estimates can be used (not yet implemented!) (in simulations we use the average from 2 last completed jobs)
- campaign from a stream of jobs: we group jobs based on delay from the first submission



A SEMI - ACTIVE SCHEDULER

- OStrich is notified about a newly submitted job; assigns 0 priority to this job
- each 1-10 seconds, OStrich recalculates the virtual schedule (new jobs, completed jobs, changed jobs)
- OStrich assigns decreasing priorities to jobs by campaign order

- the main SLURM daemon uses priorities to order jobs for FCFS/backfill



https://www.flickr.com/photos/rivenimagery/8359976129/

EXPERIMENTS

(still work in progress…)

http://www.flickr.com/photos/steveharris/245780134/

OSTRICH IS FAST!
50K+ JOBS SCHEDULED IN 0.04 SECONDS

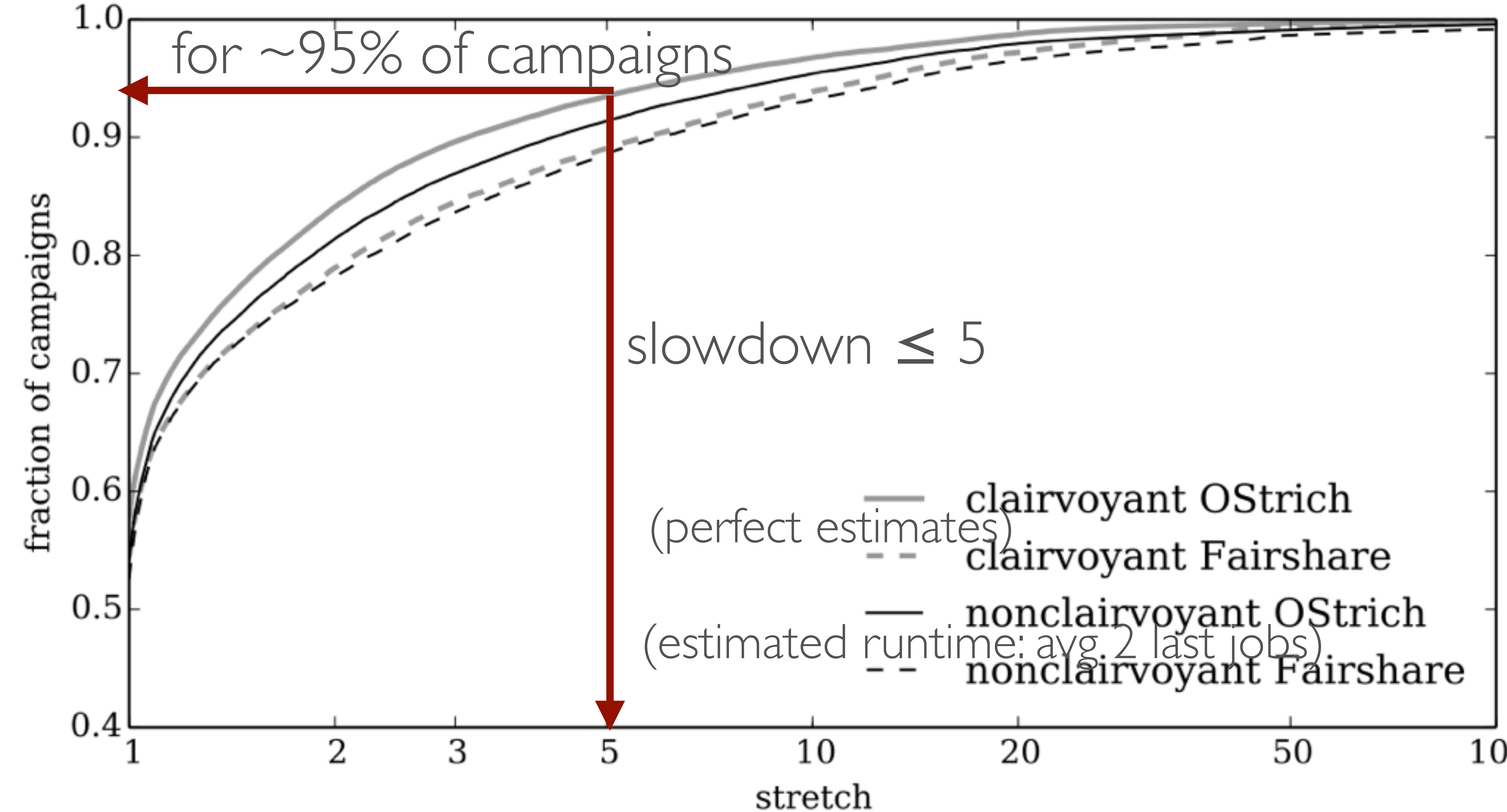
we emulated a cluster head node on a normal PC

IN PRODUCTION: 25K+ JOBS SCHEDULED SINCE JULY 2014 NO MAJOR PROBLEMS

running on a cluster with
262 nodes, 5056 cores,
heterogeneous architecture
(ICM: Warsaw Supercomputing Center
site report tomorrow at14:05)

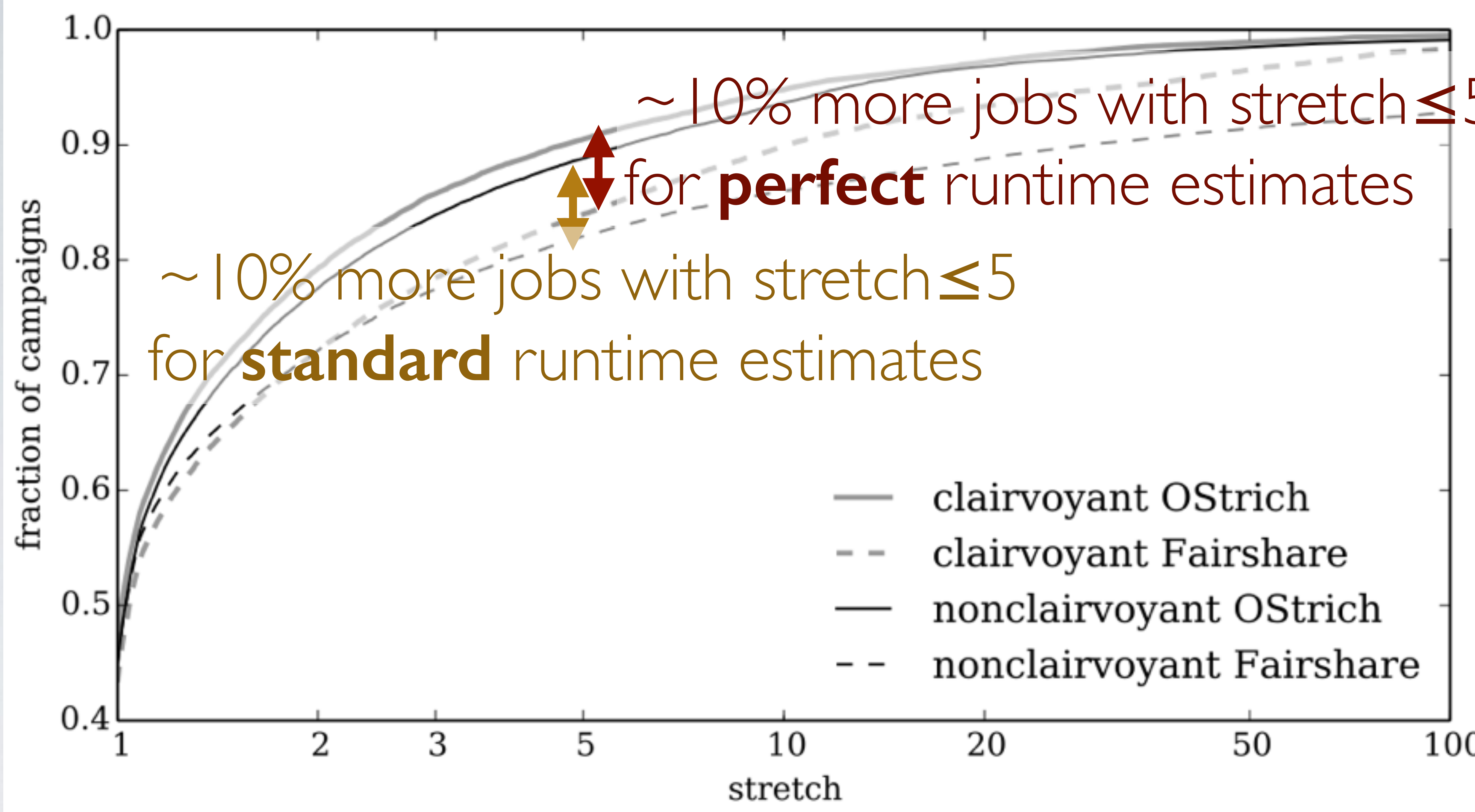
HOW GOOD IS THE ALGORITHM FROM USERS' PERSPECTIVE?

tests on a simulator
using recorded logs
from Dror Feitelson's archive



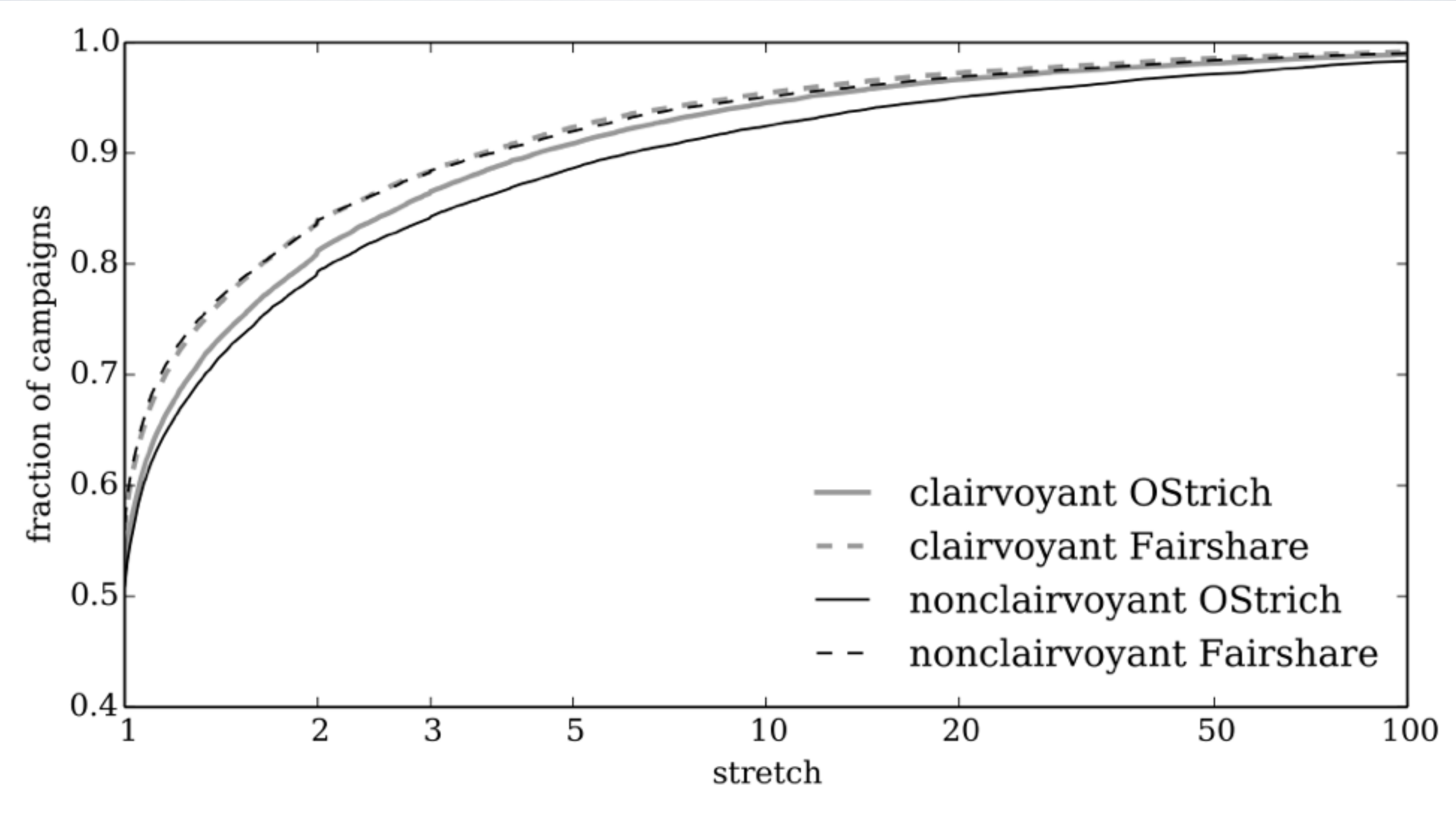
OSTRICH IS MORE EFFICIENT THAN FAIRSHARE (FOR **SOME** LOGS !)

Log from ANL Thunder BlueGene/P, 160k cores, 0.9x time compression



THE MORE CAMPAIGN-LIKE THE LOG, THE LARGER THE DIFFERENCE

Log from ANL Thunder BlueGene/P, 160k cores, 0.8x time compression,
jobs submitted during 30 minutes grouped and submitted together



FOR SOME LOGS, OSTRICH IS WORSE THAN FAIRSHARE

LLNL Thunder, 4k cores

0.95x time compression, 30 minutes job groups

http://www.flickr.com/photos/gravitywave/9146040/

CONCLUSIONS

CONCLUSIONS

- OStrich guarantees that the slowdown of each campaign (burst submission) is proportional to the number of users in the system
- OStrich maintains a virtual, fair-share schedule
- We have a **SLURM scheduling plugin** and a **simulator** available for download: github.com/filipjs/
- with the simulator you're able to test the performance on your workload before running in production
- OStrich can use existing configuration (shares) from multi-factor plugin

ACKNOWLEDGEMENTS

- Work inspired by a problem suggested by Jarosław Żola (SUNY Bufallo)
- The algorithm developed with Vinicius Gama Pinheiro (U. Grenoble) and Denis Trystram (U. Grenoble)
- Joseph Emeras contributed to the experimental evaluation of an earlier version of the algorithm
- Marcin Stolarek and other brave sysadmins from ICM (Warsaw Supercomputing Center) agreed to manage their machines with our scheduler!
- Work supported by Polish National Science Center UMO-2012/07/D/ST6/02440

http://www.flickr.com/photos/kapkaupunki/3110556670/

Thanks and... embrace the OStrich!

Krzysztof Rzadca, krzadca@mimuw.edu.pl

mimuw.edu.pl/~krzadca/ostrich/