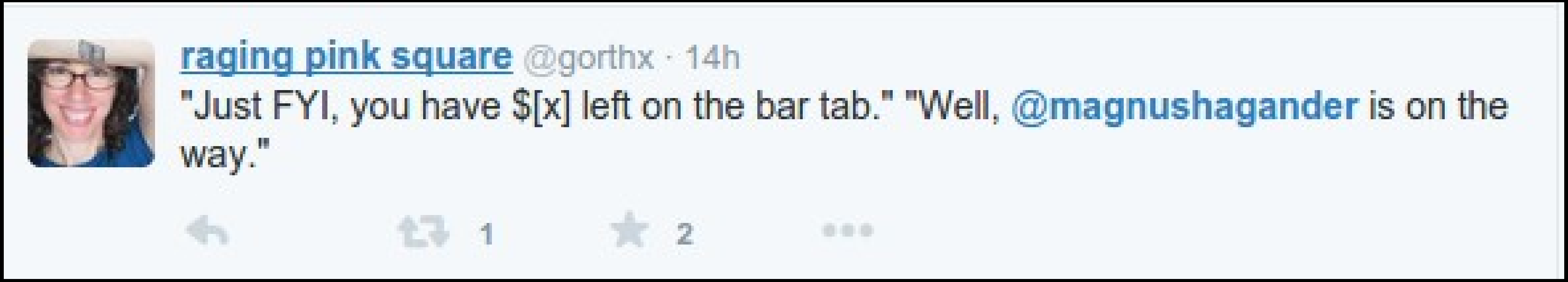
Voting PostgreSQL

Magnus Hagander

A short story...



raging pink square @gorthx · 14h

"Just FYI, you have $[x] left on the bar tab." "Well, @magnushagander is on the way."

1 2

So I got there late...

...but why?

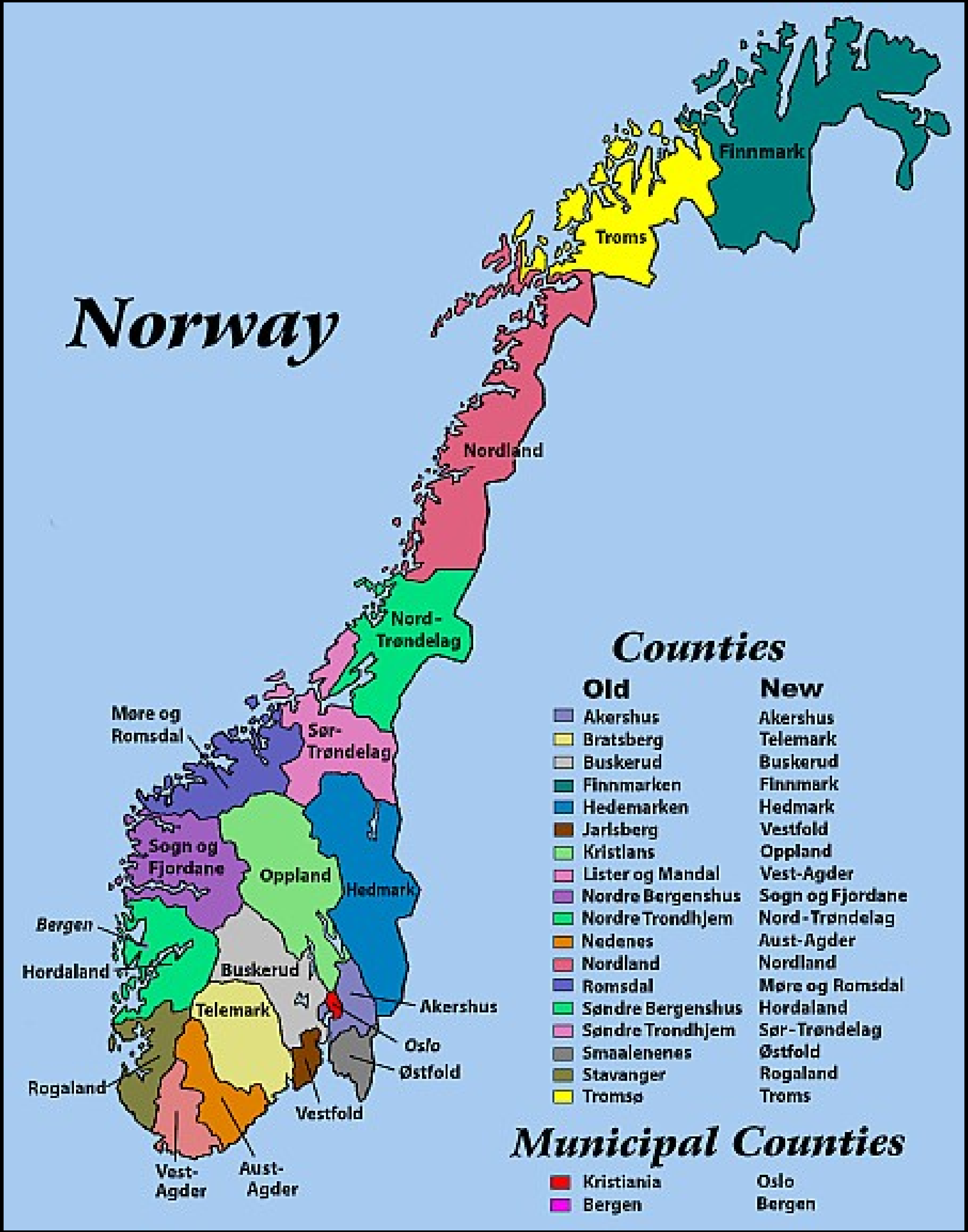
Well, that's the story

Let's start somewhere far away

Norway

Norway

Norway



Counties

	Old	New
🟪	Akershus	Akershus
🟨	Bratsberg	Telemark
⬜	Buskerud	Buskerud
🟩	Finnmarken	Finnmark
🟦	Hedemarken	Hedmark
🟫	Jarlsberg	Vestfold
🟩	Kristians	Oppland
🟥	Lister og Mandal	Vest-Agder
🟪	Nordre Bergenshus	Sogn og Fjordane
🟩	Nordre Trondhjem	Nord-Trøndelag
🟧	Nedenes	Aust-Agder
🟥	Nordland	Nordland
🟦	Romsdal	Møre og Romsdal
🟩	Søndre Bergenshus	Hordaland
🟪	Søndre Trondhjem	Sør-Trøndelag
⬛	Smaalenenes	Østfold
🟫	Stavanger	Rogaland
🟨	Tromsø	Troms

Municipal Counties

🟥	Kristiania	Oslo
🟪	Bergen	Bergen

Voting order

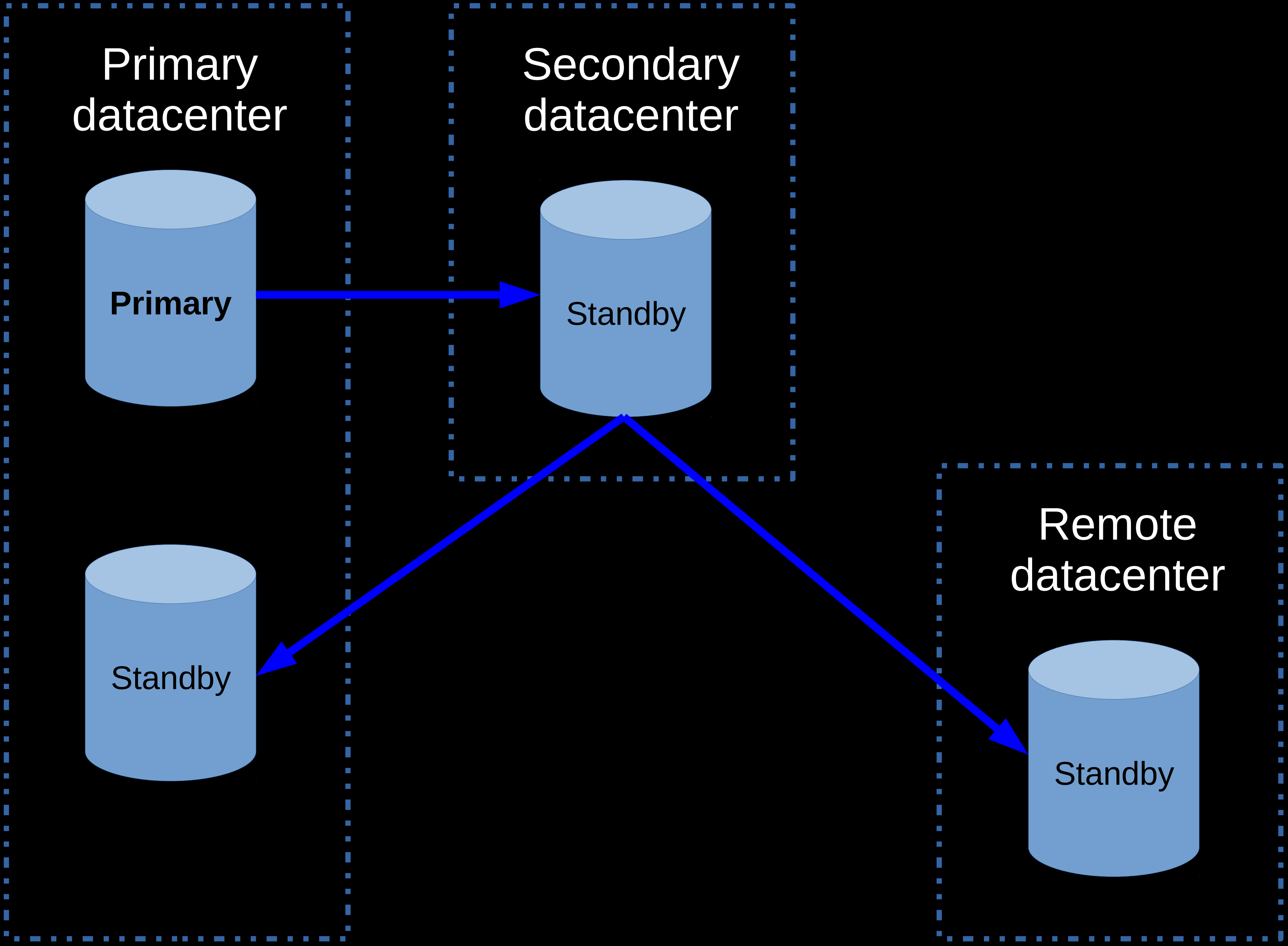
- Some pre-voting
- Majority done on election day
  - Opens 8AM, closes 9PM
  - Paper ballots
  - Counted locally
  - Scanned centrally
  - Incremental results posted from PM

Election Administration System

- Live
  - Who can vote?
  - Who did vote?
- Batch
  - Scanned results
- Output
  - Who is winning?

Election Administration System

- Locally developed application
  - Originally inherited legacy...
- WildFly clusters for different works
  - Almost entirely Hibernate
- Single PostgreSQL backend cluster
  - 9.3 on RHEL
  - Bare metal hardware, SSD


Primary datacenter
Primary
Standby
Secondary datacenter
Standby
Remote datacenter
Standby

Everybody worried about perf

- Some experiences with previous solutions
- No full-scale performance tests
  - Difficult to build proper tests

In general worked very well

- Mostly 15-20% load
  - 48 core box, 32Gb RAM
- Very fast response times
- Bottlenecks were elsewhere
  - (and there were a number)

Two noteworthy events

Unintentional serializing

- Scanning interface used “homemade sequences”
- Trigger that updated individual row in table
- Not caught in testing
  - Not enough concurrency tested
  - Actual scanning application also fairly slow

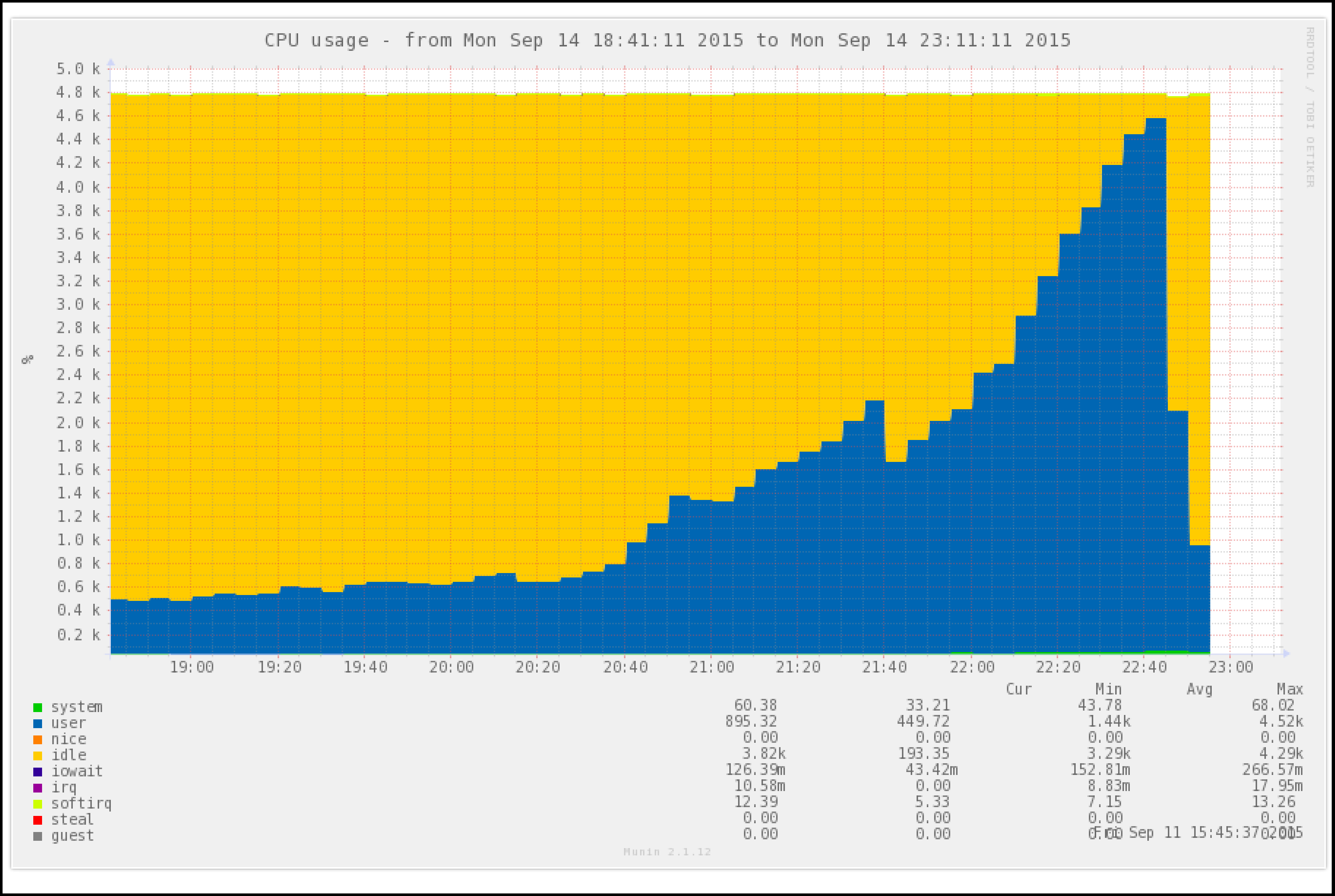
Unintentional serializing

- Tracked down with *pg_locks*
- Replaced with SEQUENCE

Missing indexes

- One very central table
- Used very central late in the process
  - Few 1000s queries / second
  - Simple JOINs
- Performed very well
  - Until it grew

Missing indexes



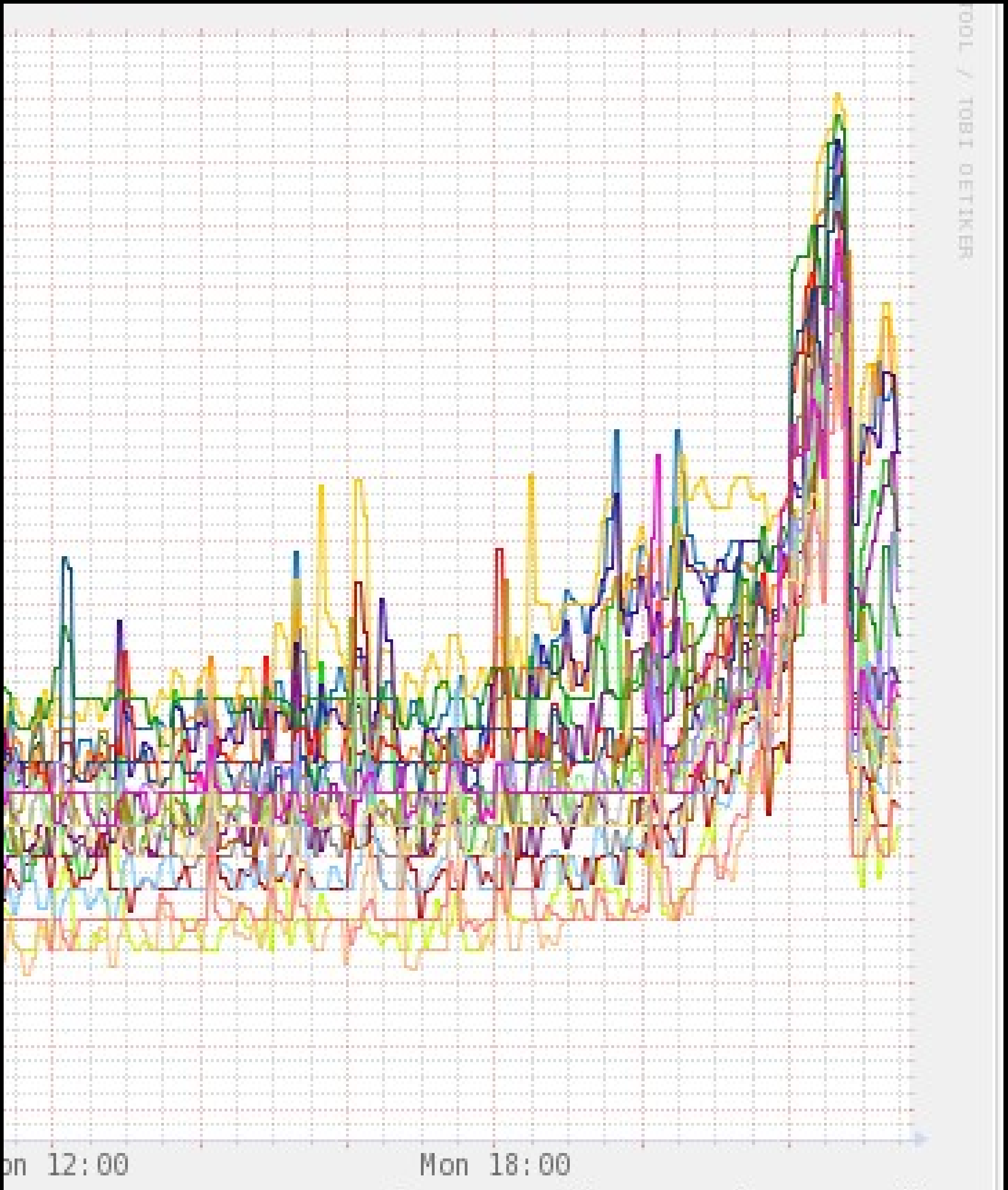
	Cur	Min	Avg	Max
🟩 system	60.38	33.21	43.78	68.02
🟦 user	895.32	449.72	1.44k	4.52k
🟧 nice	0.00	0.00	0.00	0.00
🟨 idle	3.82k	193.35	3.29k	4.29k
🟪 iowait	126.39m	43.42m	152.81m	266.57m
🟪 irq	10.58m	0.00	8.83m	17.95m
🟩 softirq	12.39	5.33	7.15	13.26
🟥 steal	0.00	0.00	0.00	0.00
⬛ guest	0.00	0.00	0.00	0.00

Fri Sep 11 15:45:37 2015

Missing indexes

- Noticed by general system load growing
- Tracked down with *pg_stat_statements*
- Fixed with *CREATE INDEX CONCURRENTLY*

Missing indexes



Conclusion

- Democracy through PostgreSQL!