That's Billion with a B:
Scaling to the next level at WhatsApp

Rick Reed
WhatsApp

Erlang Factory SF
March 7, 2014

About

- Me
  - Joined WhatsApp in 2011
  - Learned Erlang at WhatsApp
  - Scalability & multimedia
- Team
  - Small (~10 on Erlang)
  - Handle development and ops

Erlang

- Awesome choice for WhatsApp
  - Scalability
  - Non-stop operations

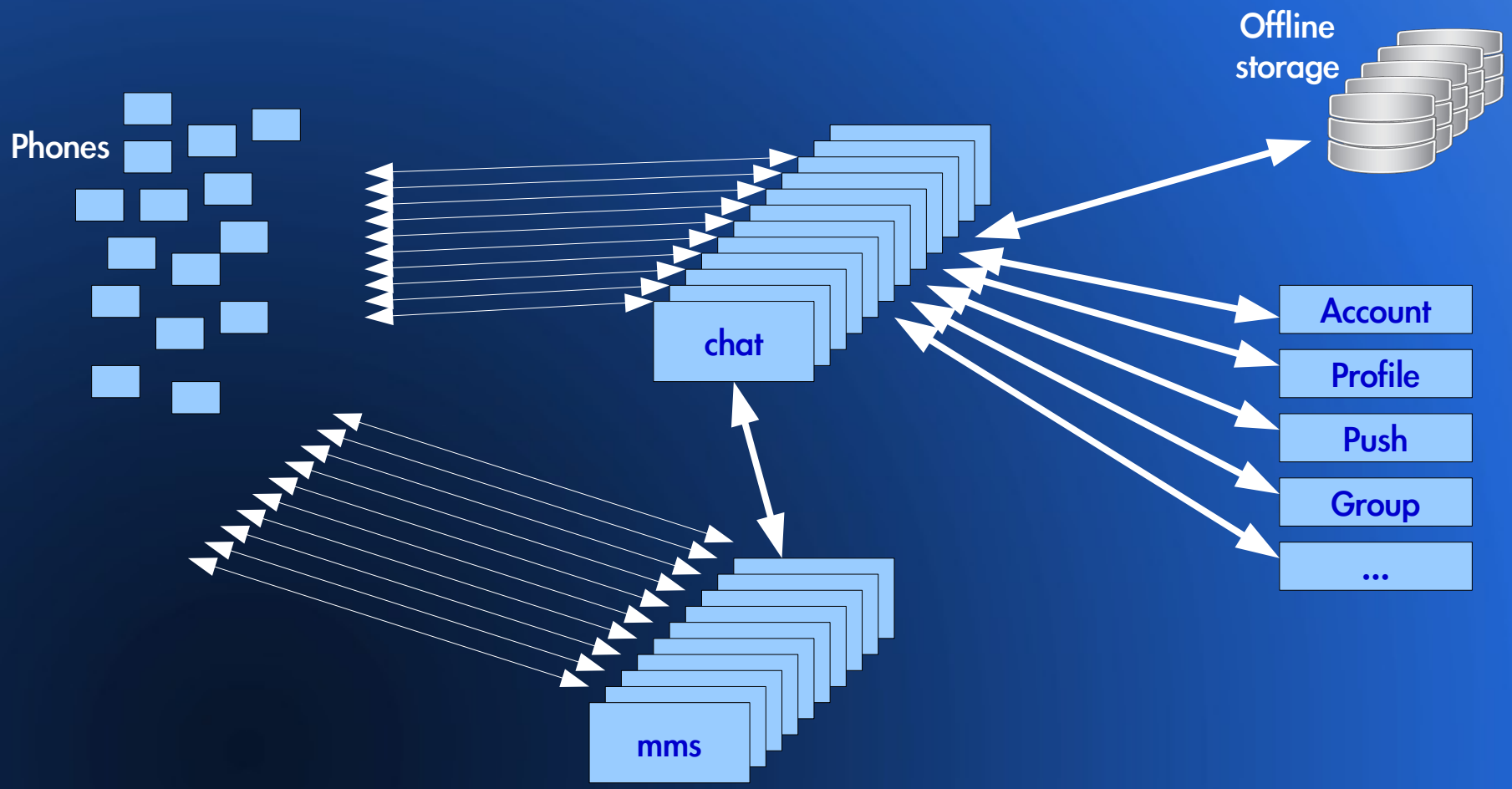
Numbers

- 465M monthly users
- 19B messages in & 40B out per day
- 600M pics, 200M voice, 100M videos
- 147M concurrent connections
- 230K peak logins/sec
- 342K peak msgs in/sec, 712K out

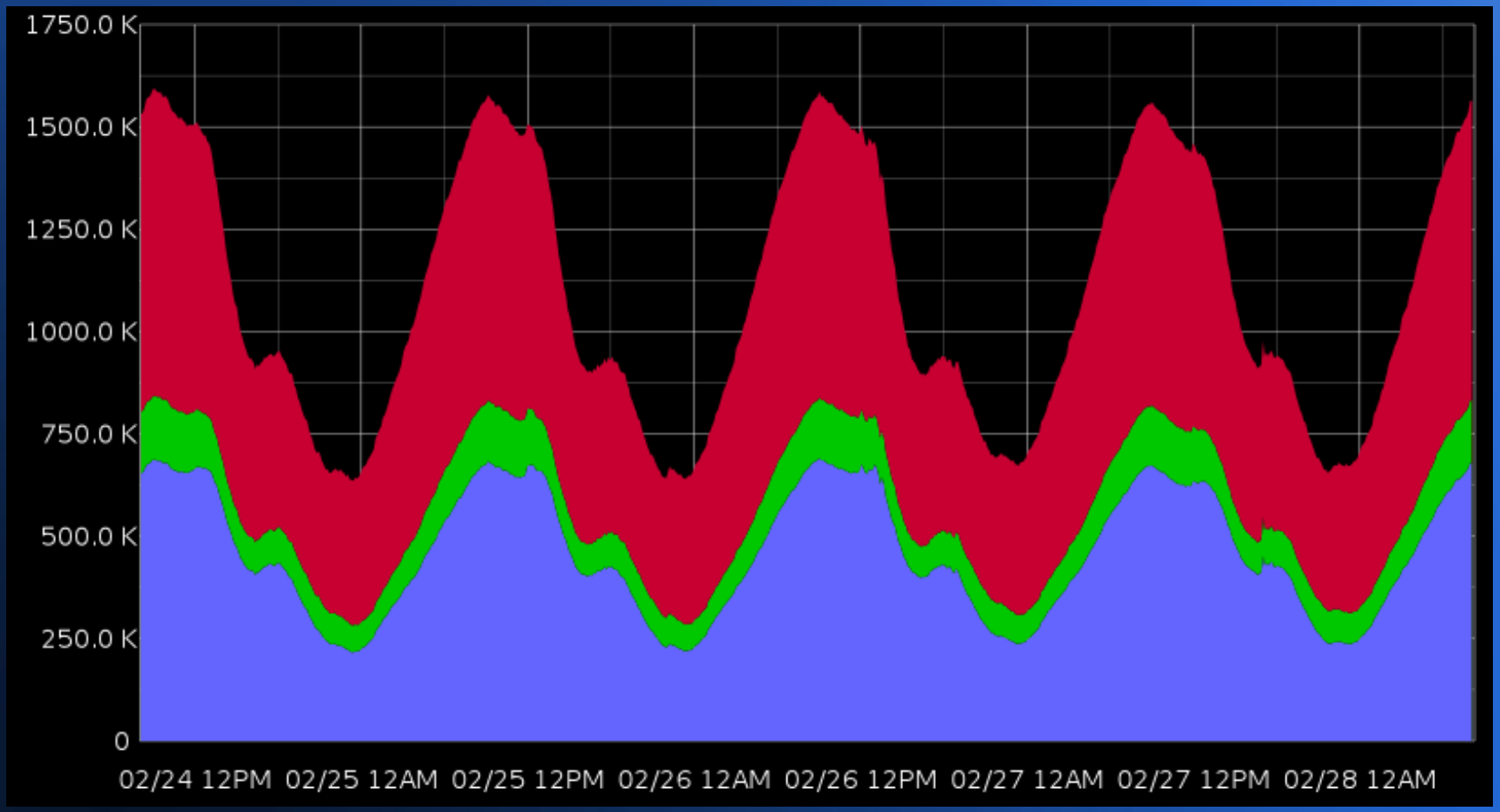
Multimedia Holiday Cheer

- 146Gb/s out (Christmas Eve)
- 360M videos downloaded (Christmas Eve)
- 2B pics downloaded (46k/s) (New Years Eve)
- 1 pic downloaded 32M times (New Years Eve)

System Overview



Output scale



Messages, Notifications, & Presence

Throughput scale

```
(4 of 16 partitions)
psh311          | ERL   msg------------------ dist---------------- wan-----------------------
time            | nodes qlen qmax  nzq   recv  msgin msgout    kbq  wanin wanout nodes    kbq
02/25 07:30:01    408    0    0    0 435661 221809 135867      0      0      0     0      0
02/25 08:00:00    408    0    0    0 446658 227349 140331      0      0      0     0      0
02/25 08:30:01    408    0    0    0 454529 231521 143486      0      0      0     0      0
02/25 09:00:01    408    0    0    0 455700 231930 143928      0      0      0     0      0
02/25 09:30:00    408    0    0    0 453770 231048 143467      0      0      0     0      0

          mnes----------------------------- io-in---out sched gc--- mem---
           tminq tmoutq   tmin  tmout nodes  kb/s  kb/s %util  /sec tot Mb
               0      0  11371  11371     4 32511 39267  44.0 47860 166808
               0      0  11418  11420     4 33502 39693  45.4 49255 166817
               0      0  11473  11472     4 34171 40460  46.3 50212 166830
               0      0  11469  11468     4 34306 40811  46.5 50374 166847
               0      0  11257  11254     4 34159 40763  46.3 50208 166870


(2 of 16 partitions)
prs101          | ERL   msg------------ dist--------- mnes---------------------- sched mem---
time            | nodes qlen qmax   recv  msgin msgout tminq tmoutq   tmin  tmout %util tot Mb
02/24 10:00:00    400    0    0 357383 174975 104489     0      0  76961  76999  27.7  15297
02/24 10:30:00    400    0    0 352178 172389 102970     0      0  75913  75893  27.3  15352
02/24 11:00:01    400    0    0 347643 170111 101688     0      0  74894  74916  27.0  15227
02/24 11:30:01    400    0    0 341300 167085  99822     0      0  73467  73478  26.6  15170
```

Db scale

(1 of 16 partitions)

Active Tables	Local Copy Type	Records	Bytes
mmd_obj2(128)	disc_copies	165,861,476	32,157,681,888
mmd_reclaim	disc_copies	5,898,714	861,434,424
mmd_ref3(128)	disc_copies	932,819,505	168,494,166,624
mmd_upload2(128)	disc_copies	1,874,045	262,430,920
mmd_xcode3(128)	disc_copies	7,786,188	2,430,697,040
schema	disc_copies	514	568,664
Total		1,114,240,442	204,206,979,560

Hardware Platform

- ~ 550 servers + standby gear
  - ~150 chat servers (~1M phones each)
  - ~250 mms servers
  - 2x2690v2 Ivy Bridge 10-core (40 threads total)
  - 64-512 GB RAM
  - SSD (except video)
  - Dual-link GigE x 2 (public & private)
- > 11,000 cores

Software Platform

- FreeBSD 9.2
- Erlang R16B01 (+patches)

Improving scalability

- Decouple
- Parallelize
- Decouple
- Optimize/Patch
- Decouple
- Monitor/Measure
- Decouple

Decouple

- Attempt to isolate trouble/bottlenecks
  - Downstream services (esp. non-essential)
  - Neighboring partitions
- Asynchronicity to minimize impact of latency on throughput

Decouple

- Avoid mnesia txn coupling: async_dirty
- Use calls only when returning data, else cast
- Make calls w/ timeouts only: no monitors
- Non-blocking casts (nosuspend) sometimes
- Large distribution buffers

Parallelize

- Work distribution: start with gen_server
  - Spread work to multiple workers: gen_factory
  - Spread dispatch to multiple procs: gen_industry
  - Worker select via key (for db) or FIFO (for i/o)
- Partitioned services
  - Usu. 2-32 partitions
  - pg2 addressing
  - Primary/secondary (usu. in pairs)

Parallelize

- mnesia
  - Mostly async_dirty
    - Isolate records to 1 node/1 process via hashing
    - Each frag read/written on only 1 node
  - Multiple mnesia_tm: parallel replication streams
  - Multiple mnesia dirs: parallel i/o during dumps
  - Multiple mnesia “islands” (usu. 2 nodes/isle)
    - Better schema ops completion
    - Better load-time coordination

Decouple

- Avoid head-of-line blocking
  - Separate read & write queues
  - Separate inter-node queues
    - Avoid blocking when single node has problem
    - Node-to-node message forwarding
    - mnesia async_dirty replication
  - “Queuer” FIFO worker dispatch

Optimize

- Offline storage I/O bottleneck
  - I/O bottleneck writing to mailboxes
  - Most messages picked up very quickly
  - Add write-back cache with variable sync delay
  - Can absorb overloads via sync delay

```
pop/s msgs/p nonz% cach% xcac% synca  maxa  rd/s push/s   wr/s
12694    5.9  24.7  78.3  98.7    21 51182    41  17035  10564
```

Optimize

- Offline storage (recent improvements)
  - Fixed head-of-line blocking in async file i/o
    - (BEAM patch to enable round-robin async i/o)
  - More efficient handling of large mailboxes
    - Keep large mailboxes from polluting cache

Optimize

- Overgrown SSL session cache
  - Slow connection setup
  - Lowered cache timeout

Optimize

- Slow access to mnesia table with lots of frags
  - Account table has 512 frags
  - Sparse mapping over islands/partitions
  - After adding hosts, throughput went down!
  - Unusually slow record access
  - On a hunch, looked at ets:info(stats)
  - Hash chains >2K (target is 7). Oops.

Optimize

- mnesia frags (cont.)
  - Small percentage of hash buckets being used
  - ets uses *average* chain length to trigger split

```
 #define MAX_HASH 0xEFFFFFFFUL
 #define INVALID_HASH 0xFFFFFFFFUL
+#define HASH_INITVAL 33554467UL

 /* optimised version of make_hash (normal case? atomic key) */
 #define MAKE_HASH(term) \
     ((is_atom(term) ? (atom_tab(atom_val(term))->slot.bucket.hvalue) : \
-      make_hash2(term)) % MAX_HASH)
+      make_hash2_init(term, HASH_INITVAL)) % MAX_HASH)
```

Patch

- FreeBSD 9.2
  - No more patches
  - Config for large network & RAM

Patch

- Our original BEAM/OTP config/patches
  - Allocator config (for best superpage fit)
  - Real-time OS scheduler priority
  - Optimized timeofday delivery
  - Increased bif timer hash width
  - Improved check_io allocation scalability
  - Optimized prim_inet / inet accepts
  - Larger dist receive buffer

Patch

- Our original config/patches (cont.)
  - Add pg2 denormalized group member lists
  - Limit runq task stealing
  - Add send w/ prepend
  - Add port reuse for prim_file:write_file
  - Add gc throttling w/ large message queues

Patch

- New patches (since EFSF 2012 talk)
  - Add multiple timer wheels
  - Workaround mnesia_tm selective receive
  - Add multiple mnesia_tm async_dirty senders
  - Add mark/set for prim_file commands
  - Load mnesia tables from nearby node

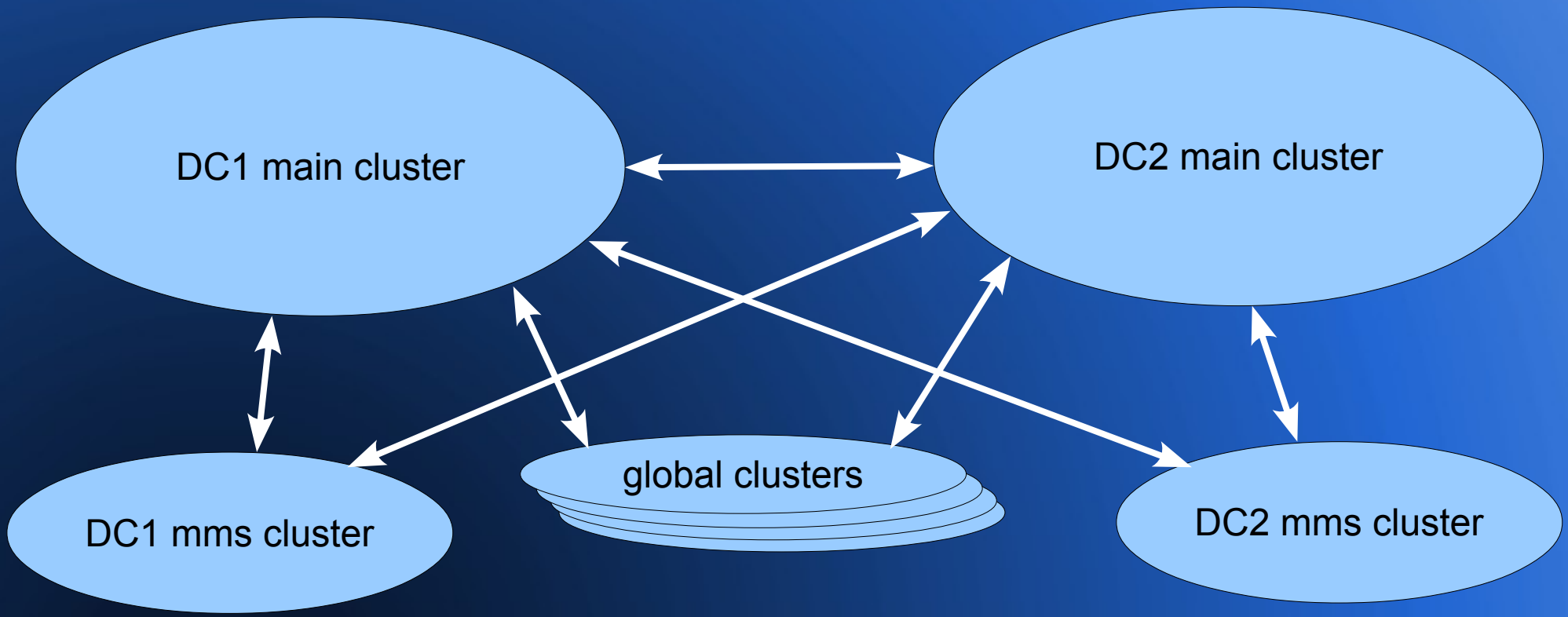
Patch

- New patches (since EFSF 2012 talk) (cont.)
  - Add round-robin scheduling for async file i/o
  - Seed ets hash to break coincidence w/ phash2
  - Optimize ets main/name tables for scale
  - Don't queue mnesia dump if already dumping

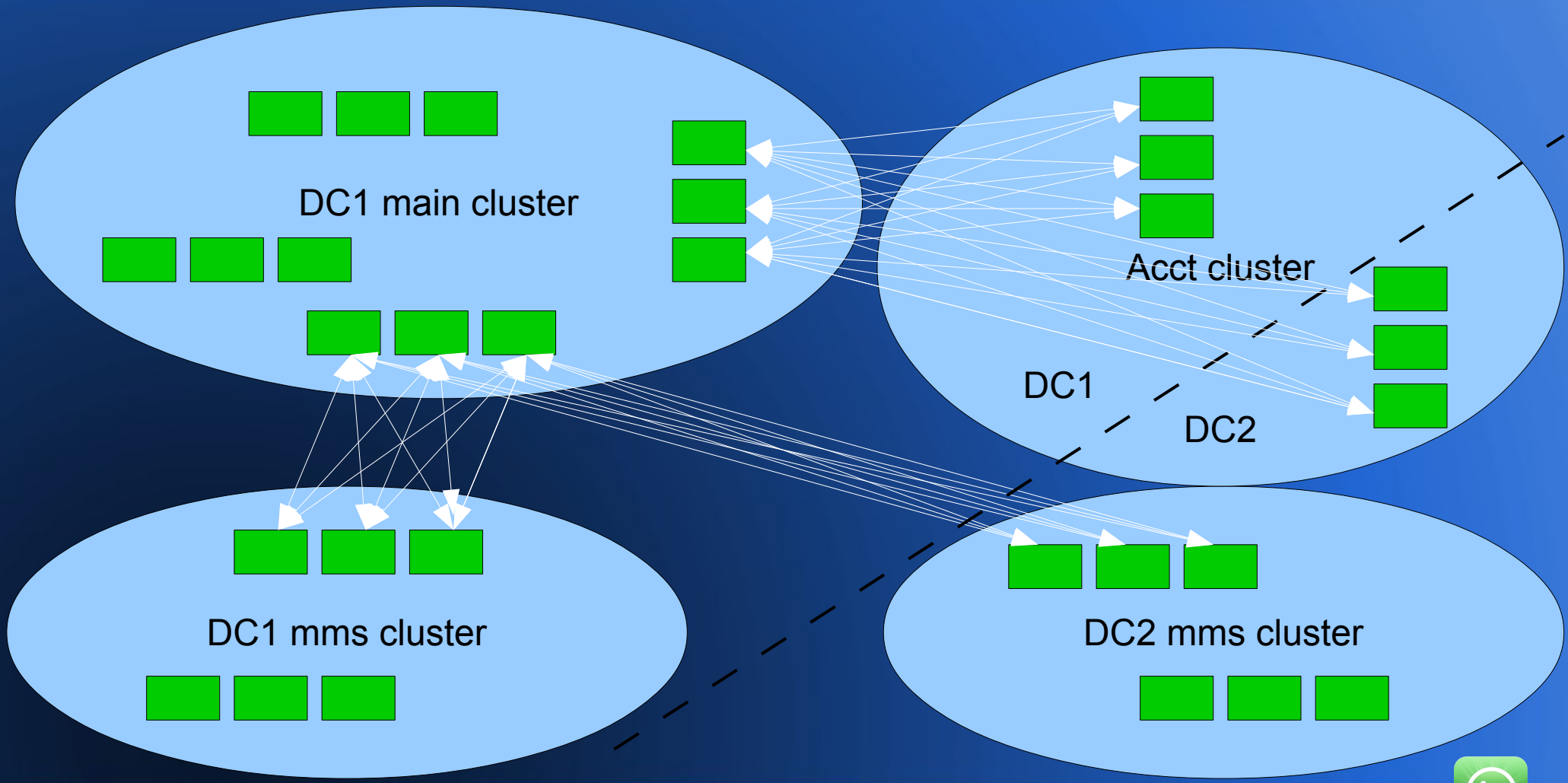
Decouple

- Meta-clustering
  - Limit size of any single cluster
  - Allow a cluster to span long distances
  - wandist: dist-like transport over gen_tcp
    - Mesh-connected functional groups of servers
  - Transparent routing layer just above pg2
    - Local pg2 members published to far-end
    - All messages are single-hop

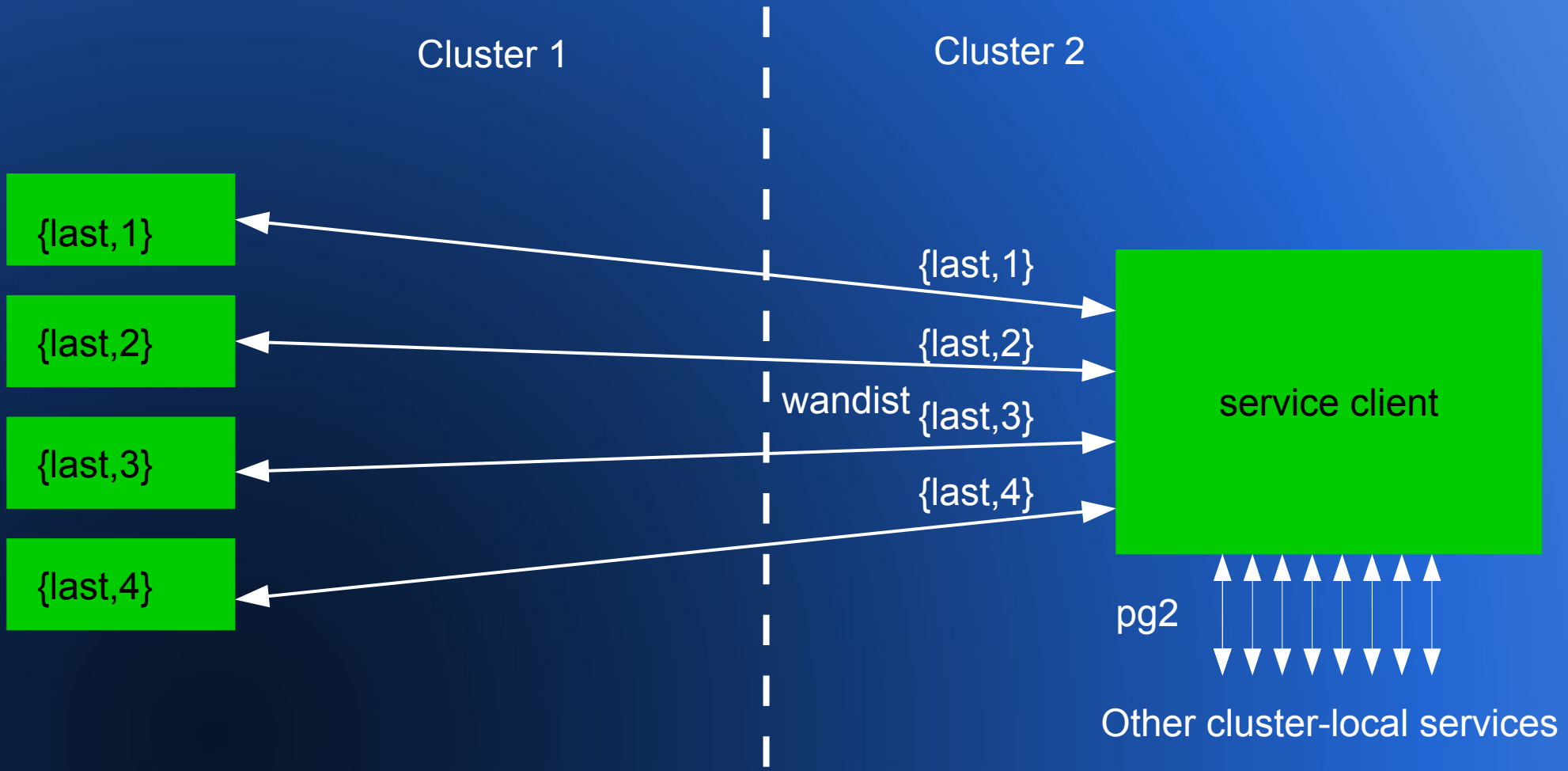
Meta-clustering



Topology


DC1 main cluster
Acct cluster
DC1
DC2
DC1 mms cluster
DC2 mms cluster

Routing


Cluster 1
Cluster 2
{last,1}
{last,2}
{last,3}
{last,4}
wandist
{last,1}
{last,2}
{last,3}
{last,4}
service client
pg2
Other cluster-local services

Clearing the minefield

- Generally able to detect/defuse scalability mines before they explode
- Events which test the system
  - World events (esp. soccer)
  - Server failures (usu. RAM)
  - Network failures
  - Bad software pushes

Clearing the minefield

- Not always successful: 2/22 outage
  - Began with back-end router glitch
  - Mass node disconnect/reconnect
  - Resulted in a novel unstable state
  - Unsuccessful in stabilizing cluster (esp. pg2)
  - Full stop & restart (first time in years)
  - Also uncovered an overly-coupled subsystem
  - Rolling out pg2 patch

Challenges

- Db scaling, esp. MMS
  - Load time (~1M objects/sec)
  - Load failures (unrecoverable backlog)
    - Bottlenecked on disk write throughput (>700MB/s)
    - Patched a selective-receive issue, but more to go
- Real-time cluster status & control at scale
  - A bunch of csshX windows no longer enough
- Power-of-2 partitioning

Questions?

- rr@ whatsapp.com
- @td_rr
- GitHub: reedr/otp

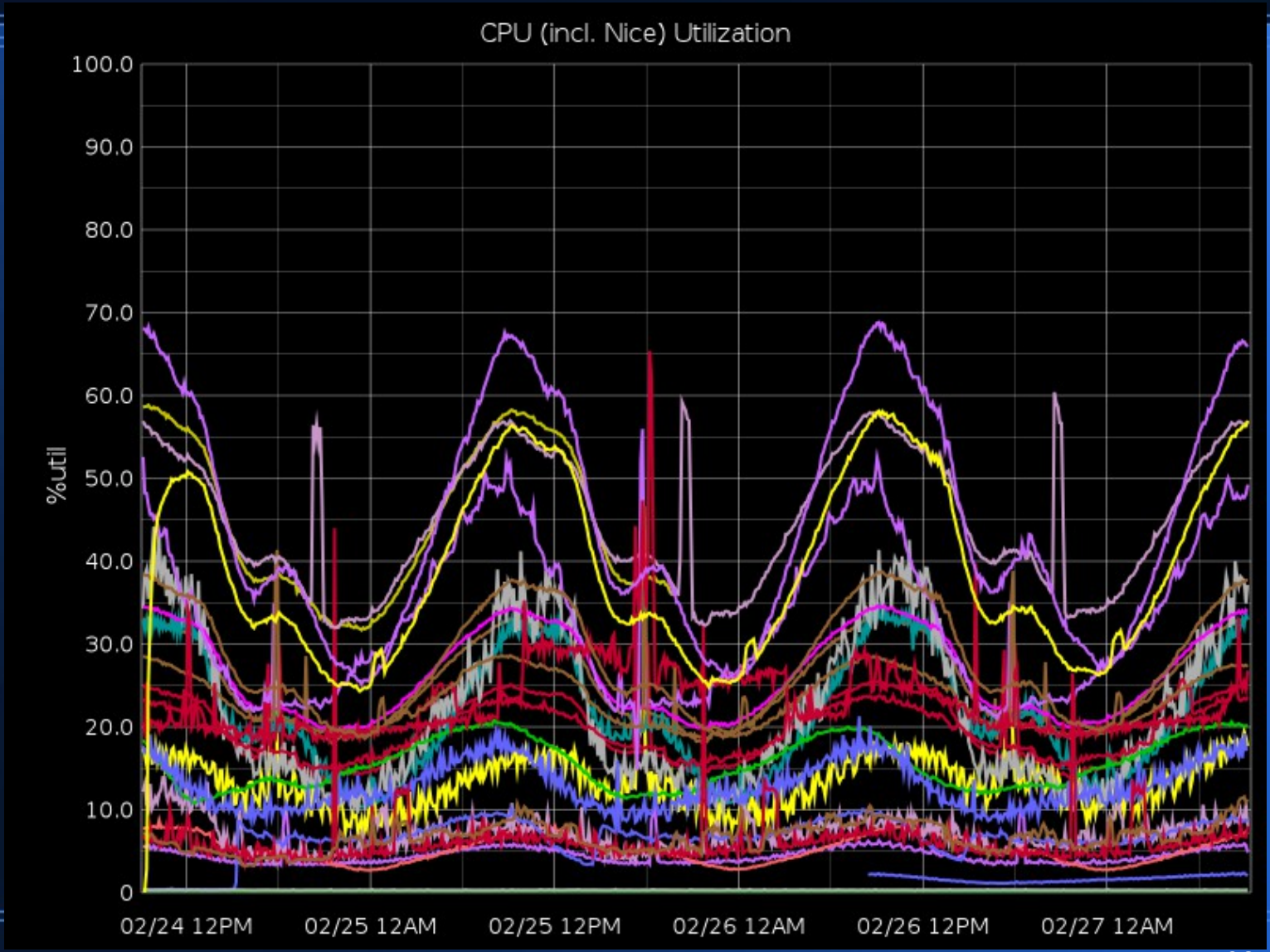
Monitor/Measure

- Per-node system metrics gathering
  - 1-second and 1-minute polling
  - Pushed to Graphite for plotting
- Per-node alerting script
  - OS limits (CPU, mem, network, disk)
  - BEAM (running, msgq backlog, sleepy scheds)
- App-level metrics
  - Pushed to Graphite

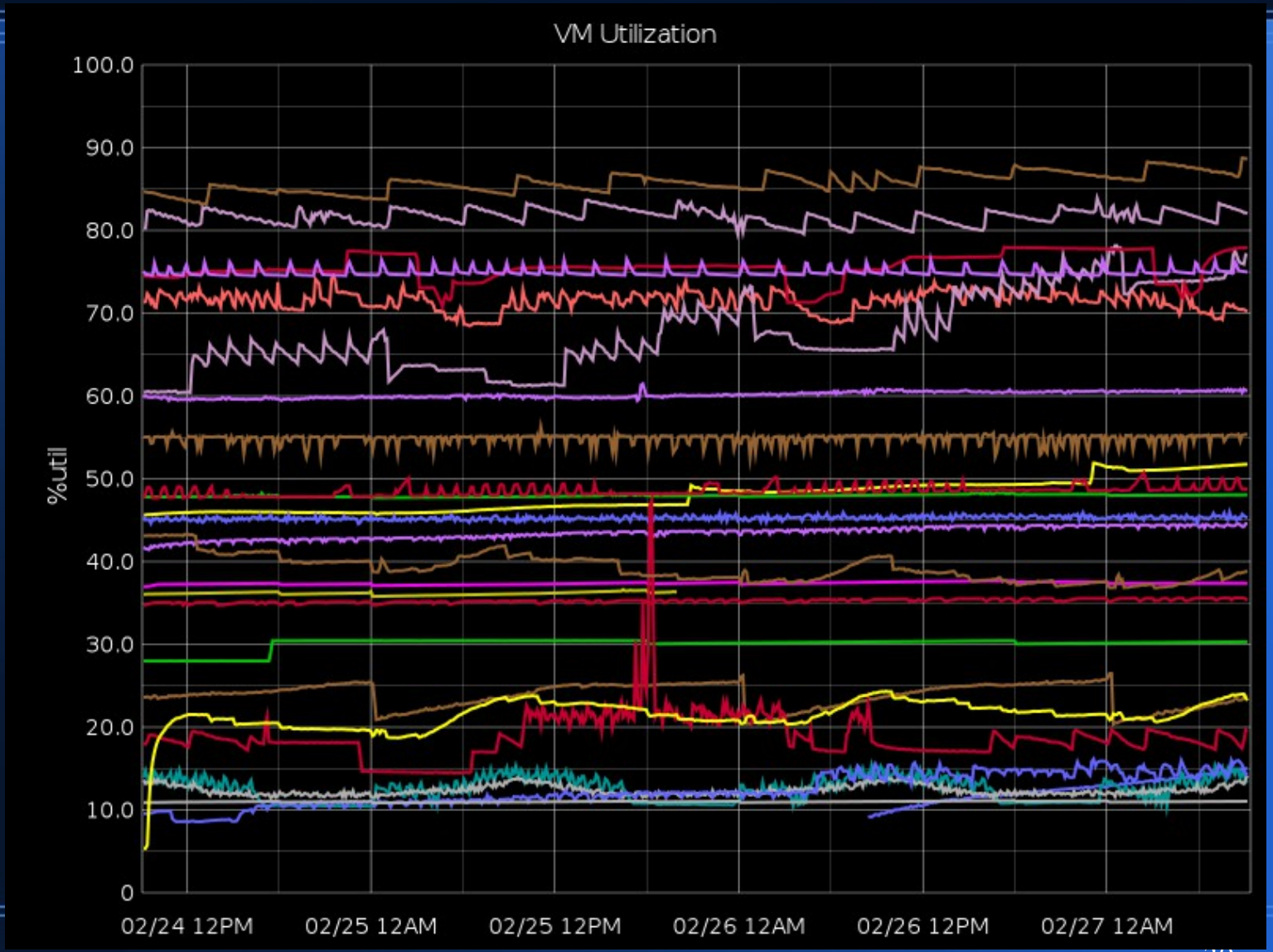
Monitor/Measure

- Capacity plan against system limits
  - CPU util%, Mem util%, Disk full%, Disk busy%
- Watch for process message queue backlog
  - Generally strive to remove all back pressure
  - Bottlenecks show as backlog
  - Alert on backlog > threshold (usu. 500k)

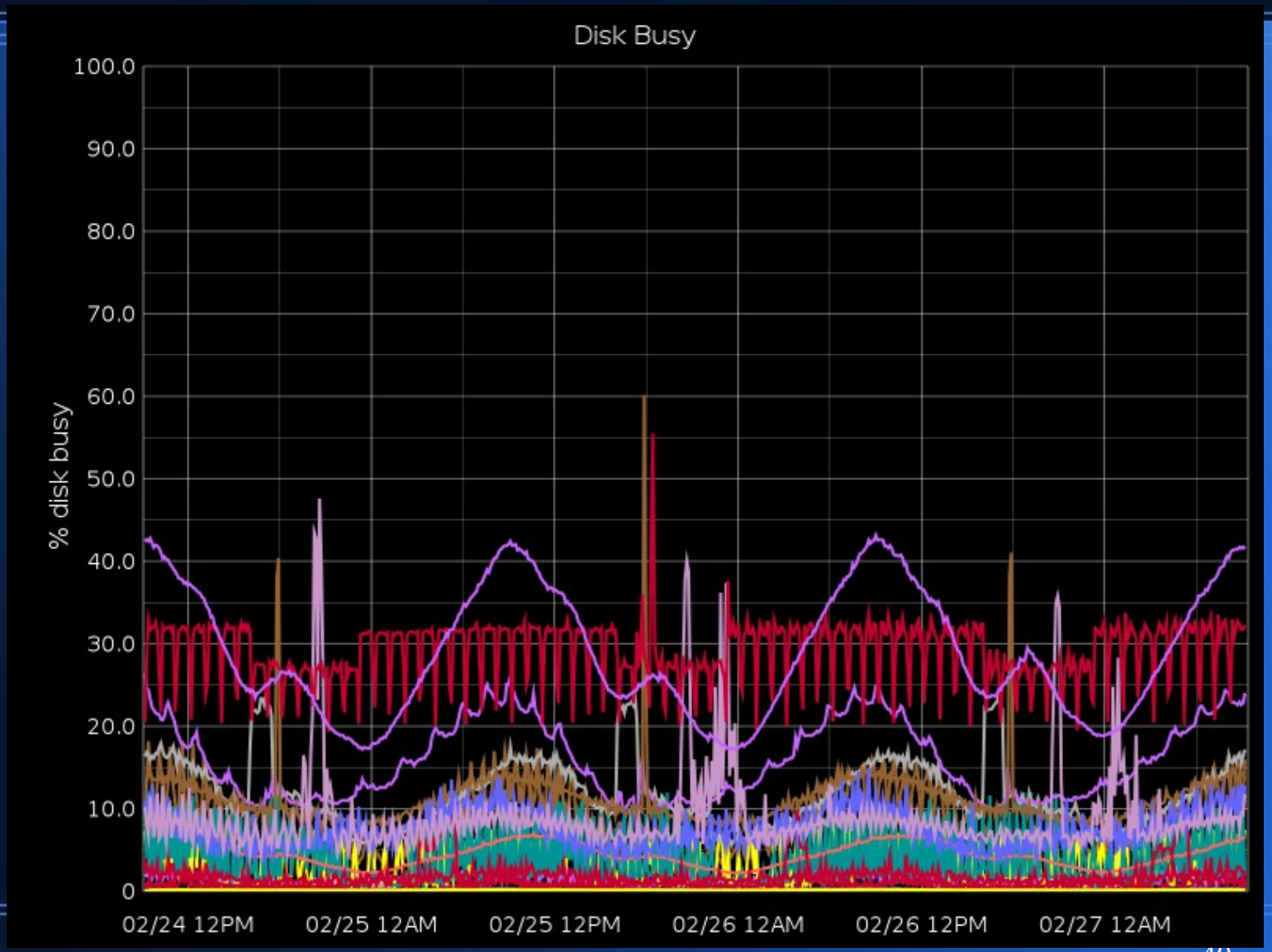
Monitor/Measure



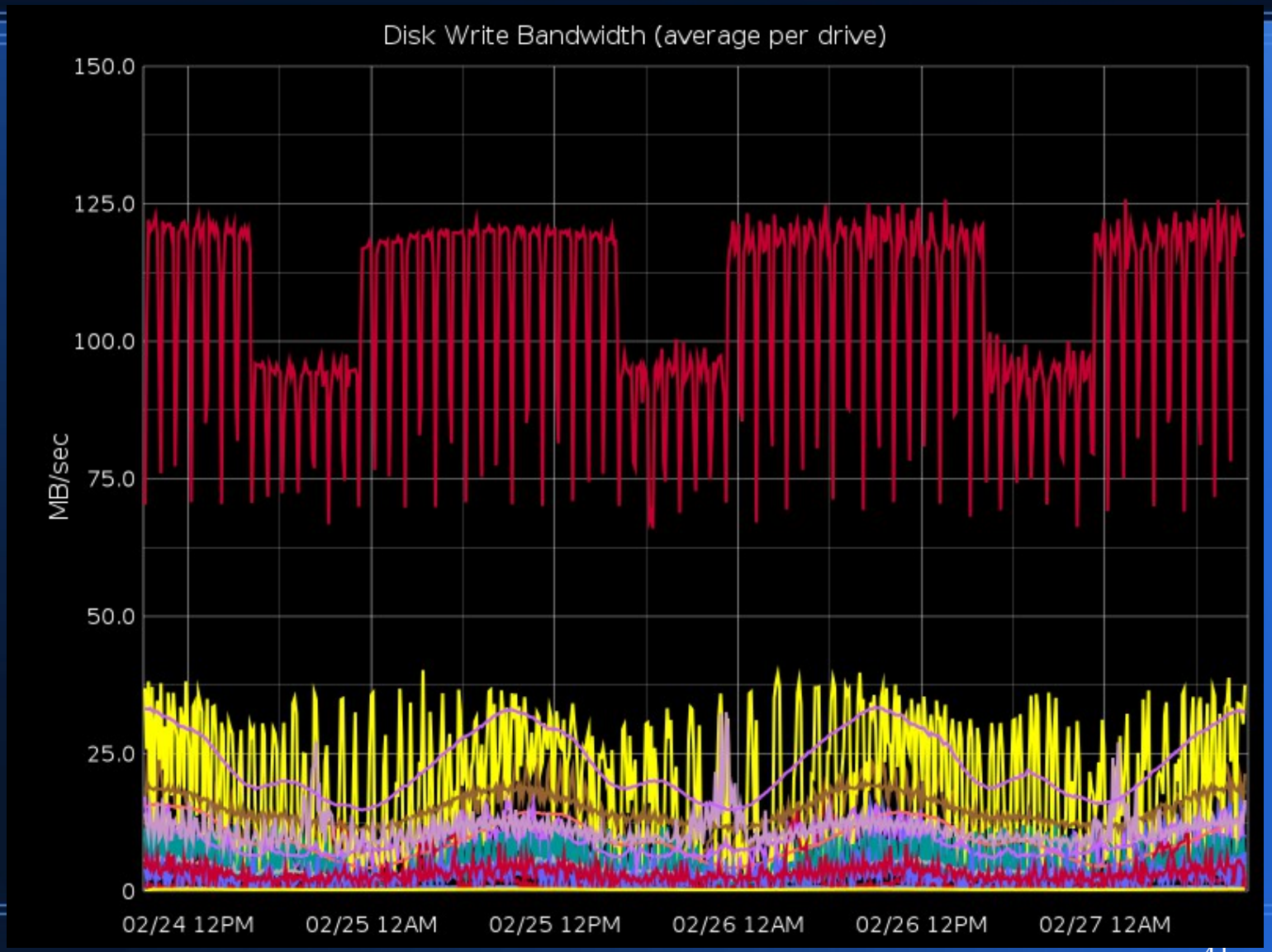
Monitor/Measure



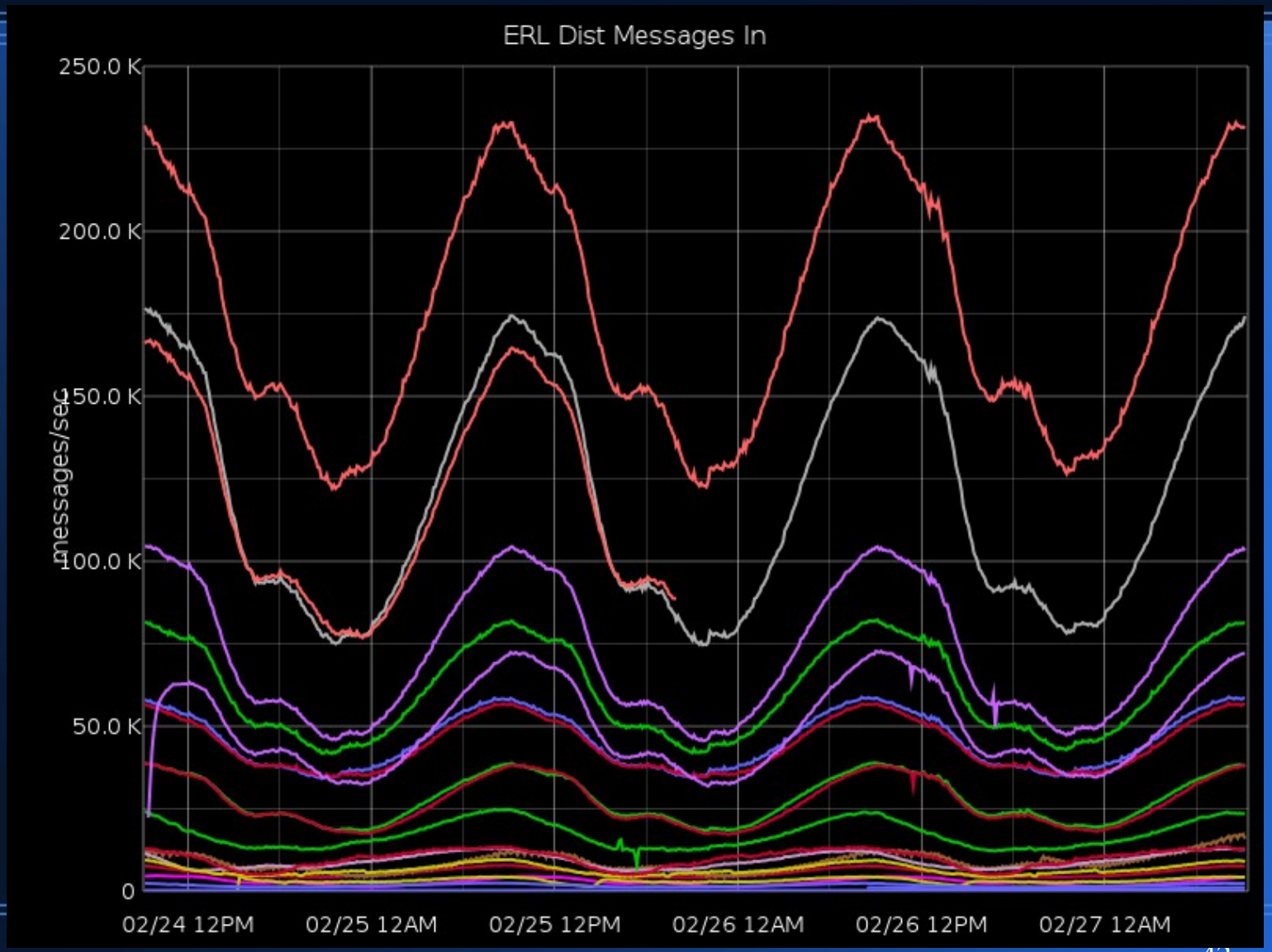
Monitor/Measure



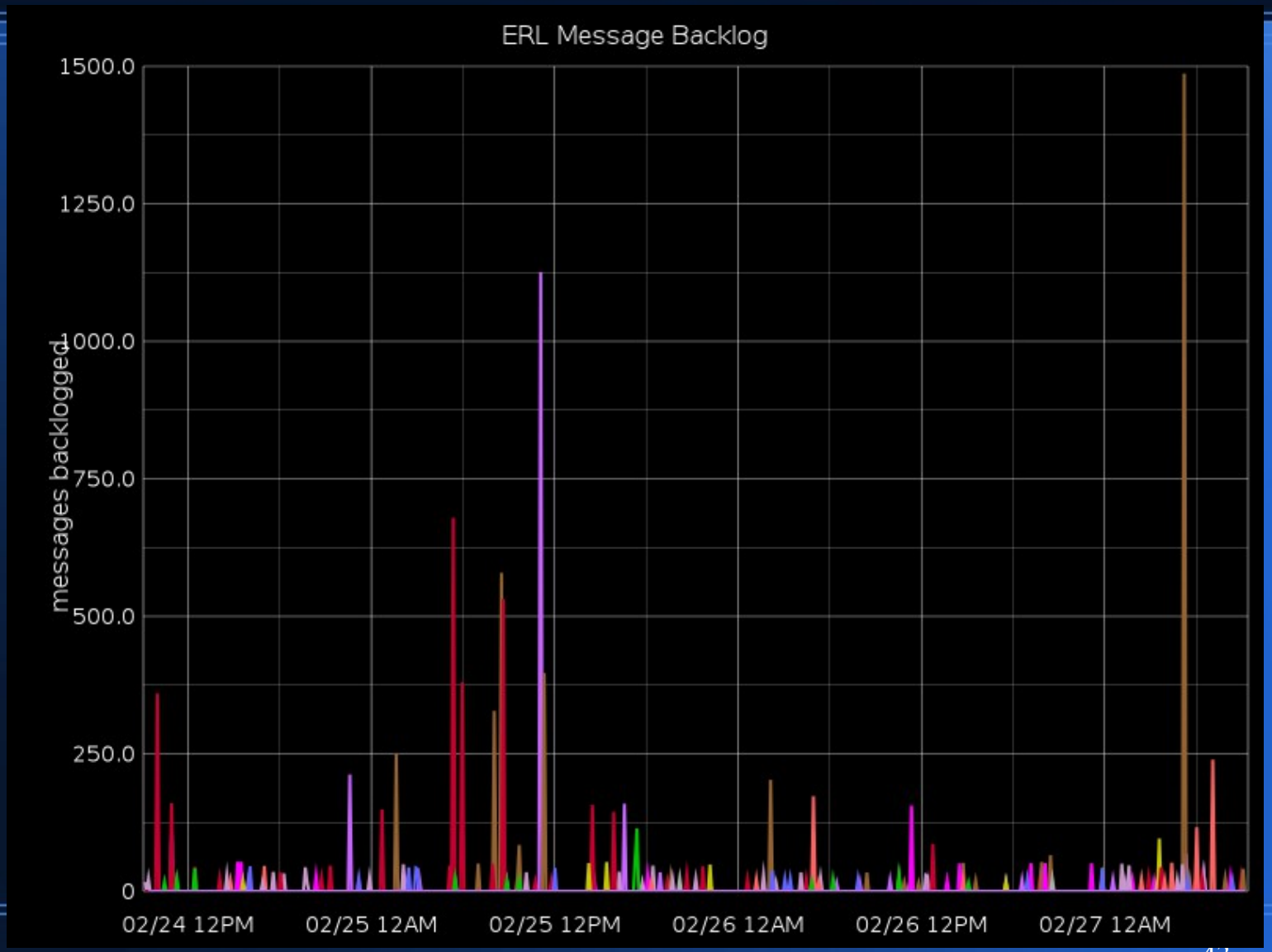
Monitor/Measure



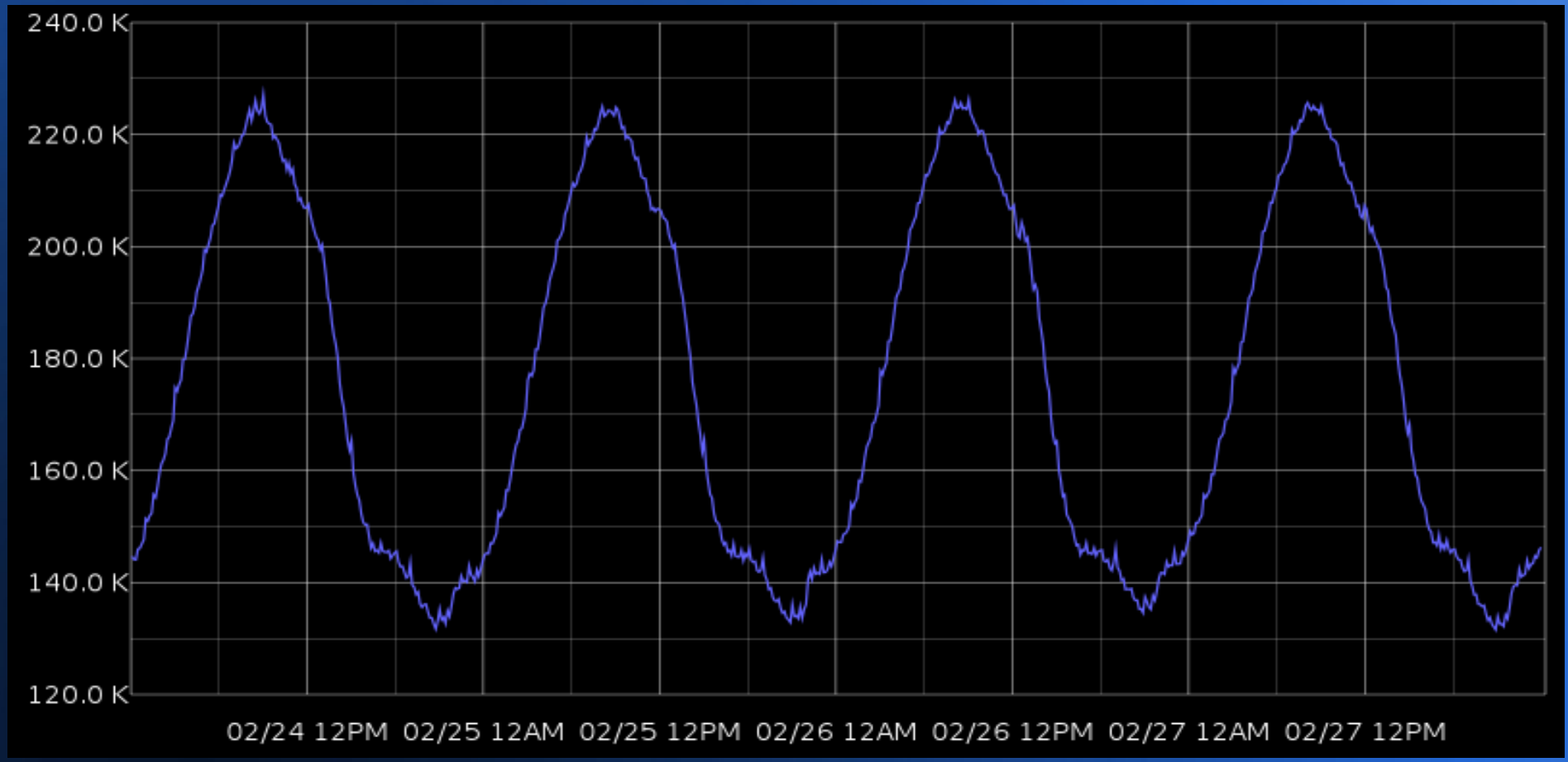
Monitor/Measure



Monitor/Measure



Input scaling



Logins