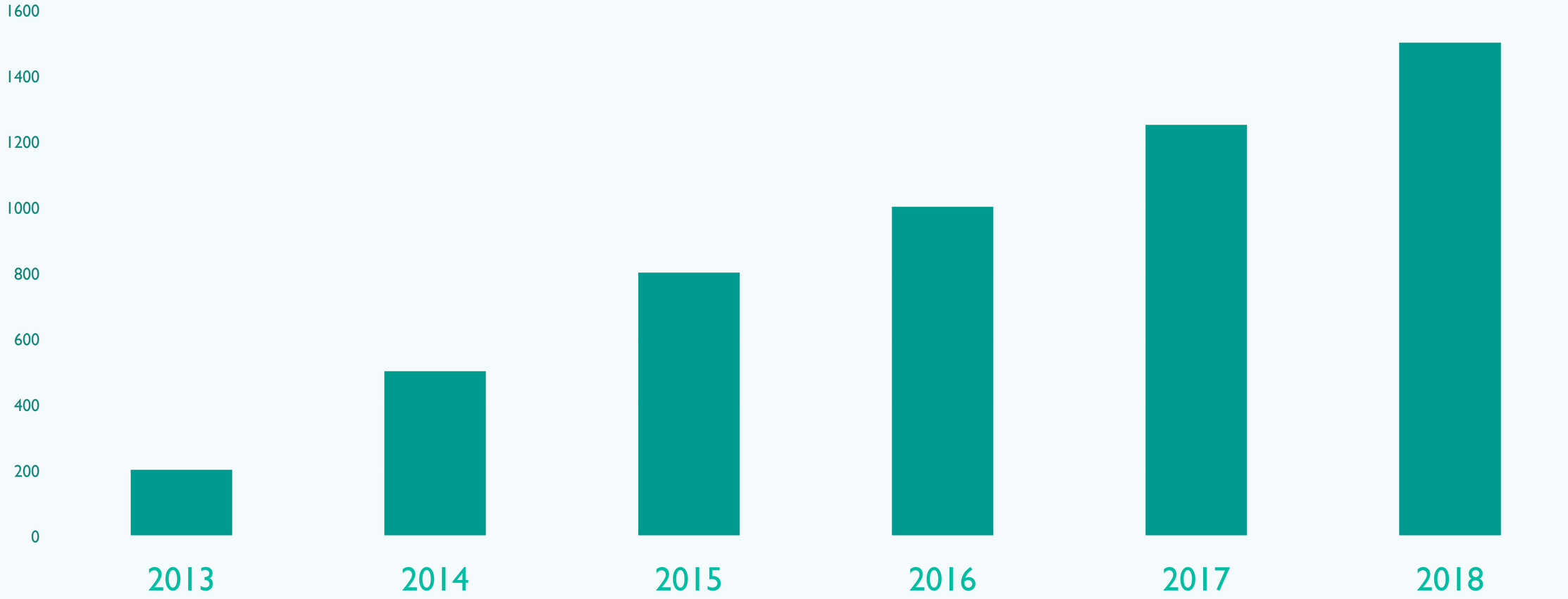
Scaling Erlang cluster to 10,000 nodes

Maxim Fedorov
Software Engineer @ WhatsApp

WhatsApp User Base Growth

From 200m to 1.5B and growing fast

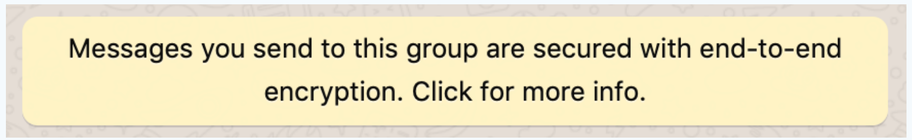


WhatsApp Features Development

Not just a simple messenger

- New platforms support
- Voice and video calls
- End-to-end encryption
- WhatsApp Business
- Live location
- … and a few more

KaiOS



Paradigm Shift

Few powerful servers to many tightly packed blades

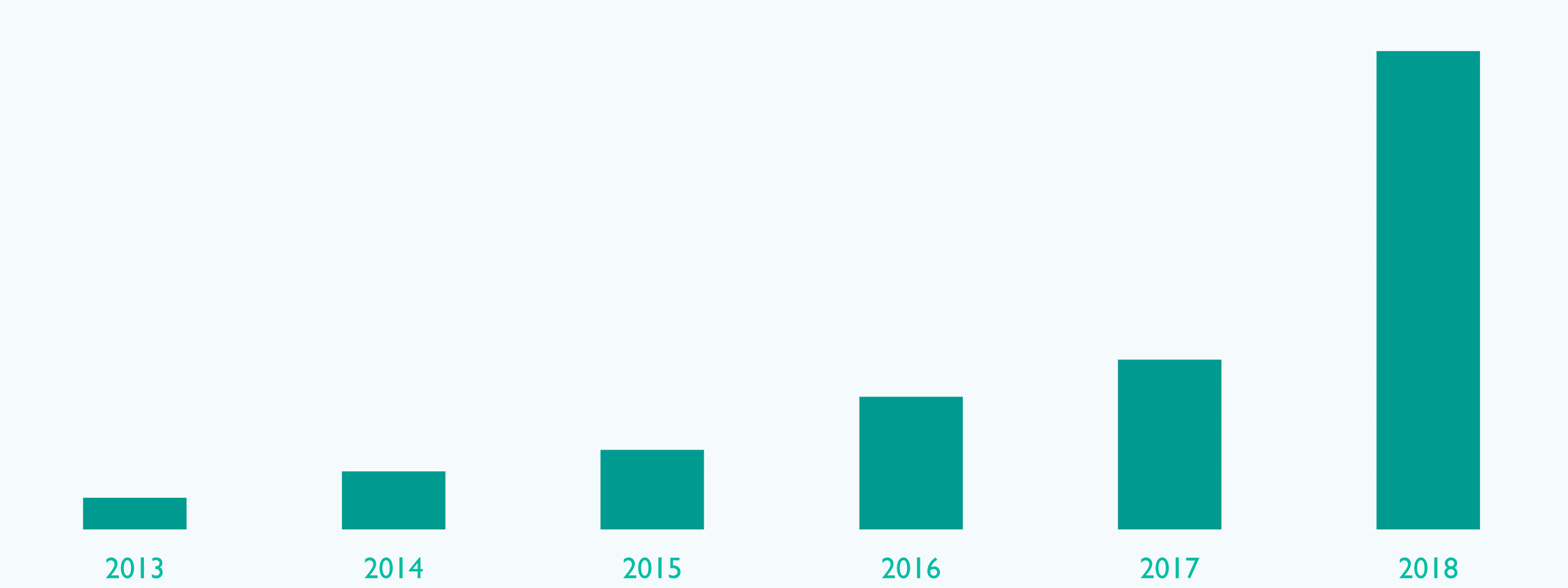
- Dual Socket Xeon E5-2690 2.6-3.5 GHz
  128 - 512 G RAM

- Xeon-D 1540 2.0 – 2.6 GHz 32 G RAM
- Dual Skylake-X, 256 G RAM

Foundation Replacement

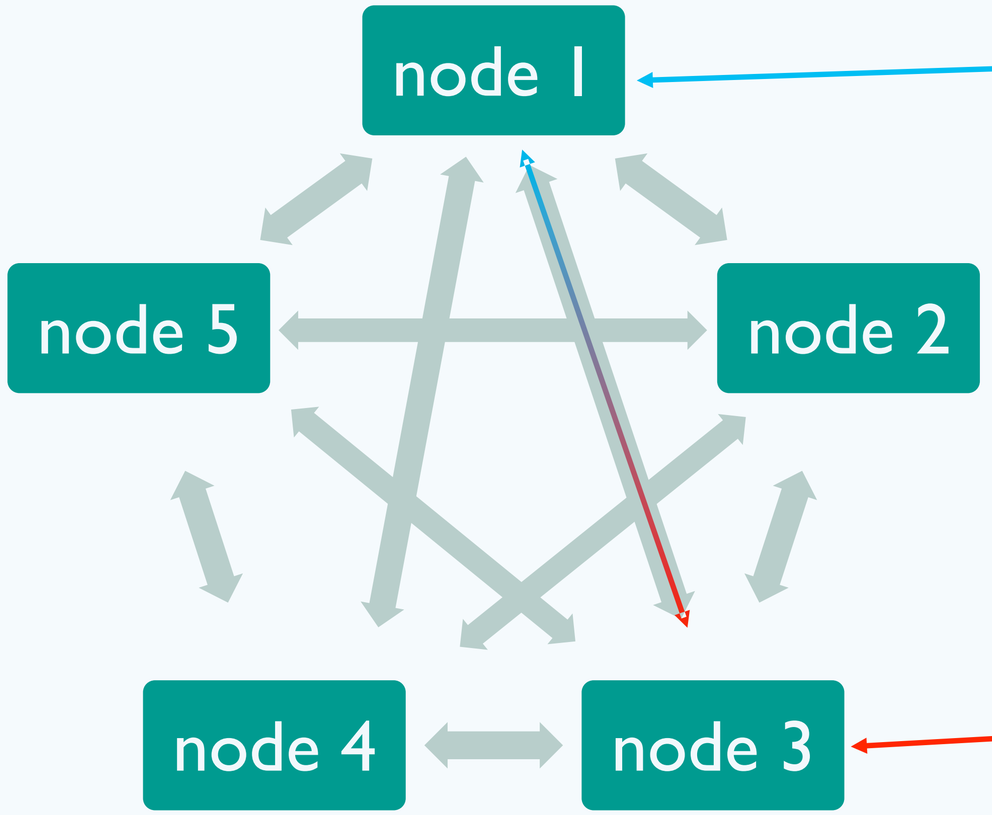
WhatsApp Cluster Size

From just a few to over 10,000



Erlang Cluster

Erlang Cluster: Fully Connected Mesh



- 1500 nodes in a single distribution cluster
- TCP connection maintenance overhead is hardly noticeable after a few tweaks
- Fast startup

Discovery
is a Problem!

Distributed Process Registry

A good problem to solve

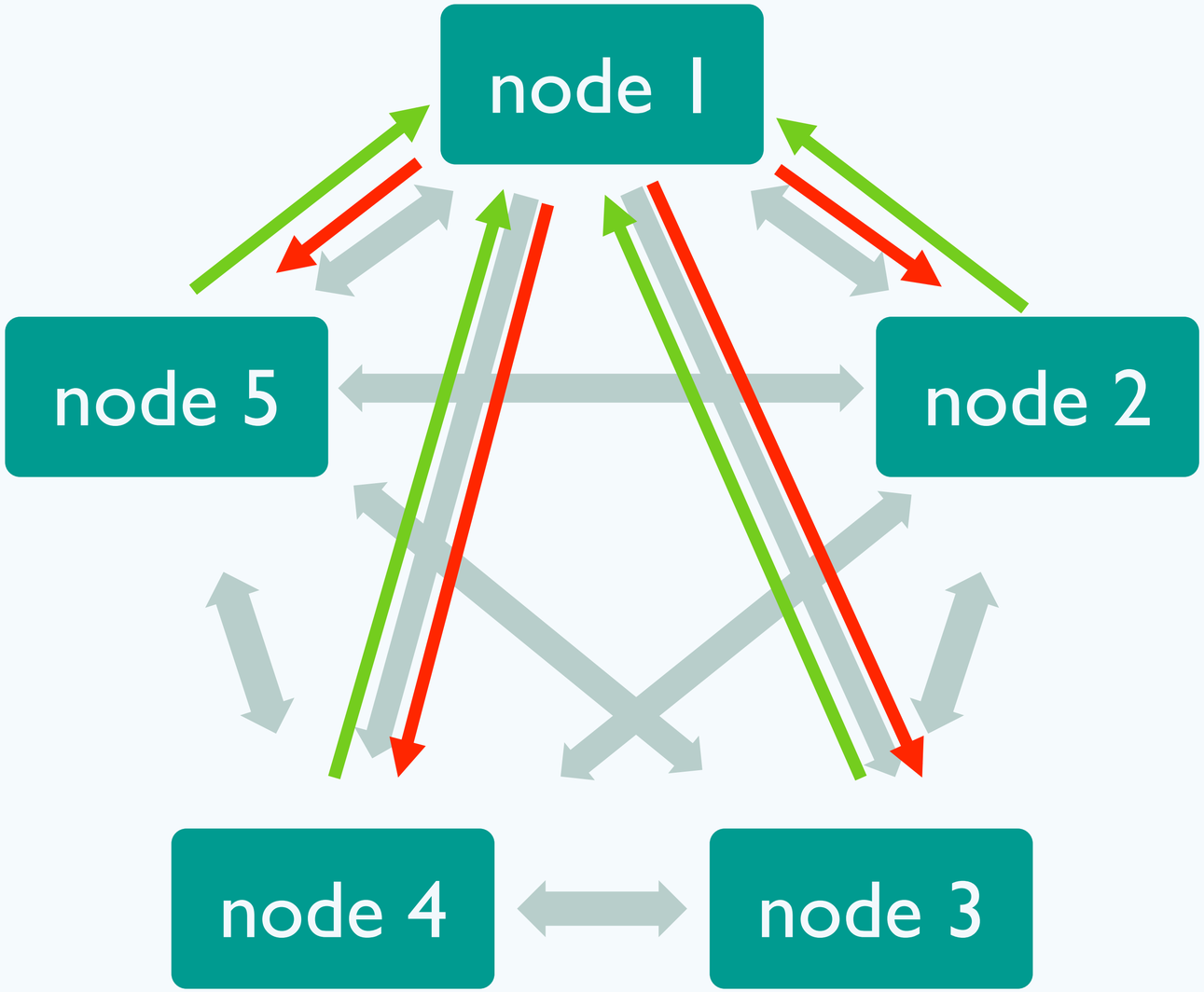
Coordinated

- *global*
- *pg2*
- gproc
- s_groups

Eventual

- Riak PG
- cpg
- Syn
- Swarm
- Lasp PG

Coordinated Approach (global)

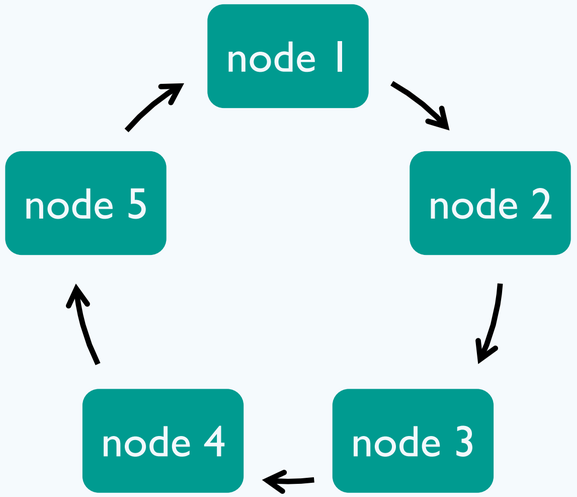


Registration

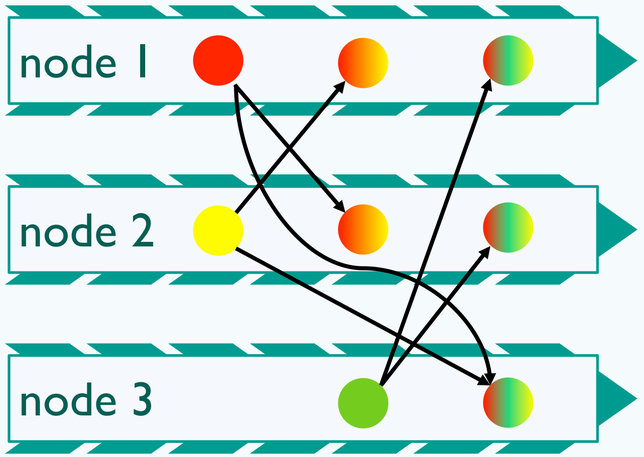
- Lock boss node
- Lock the rest
- Register process
- Unlock all except boss
- Unlock boss node

Distributed Hash Tables

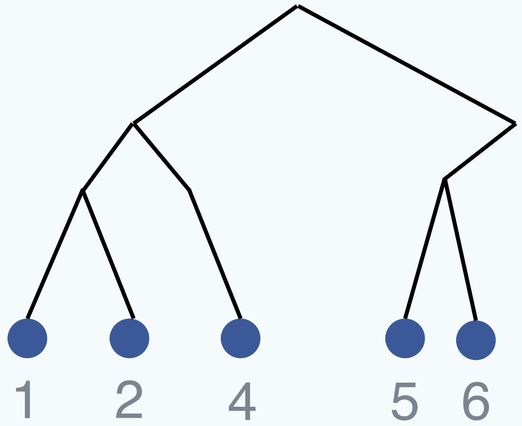
Consistent Hashing



CRDT



Kademlia



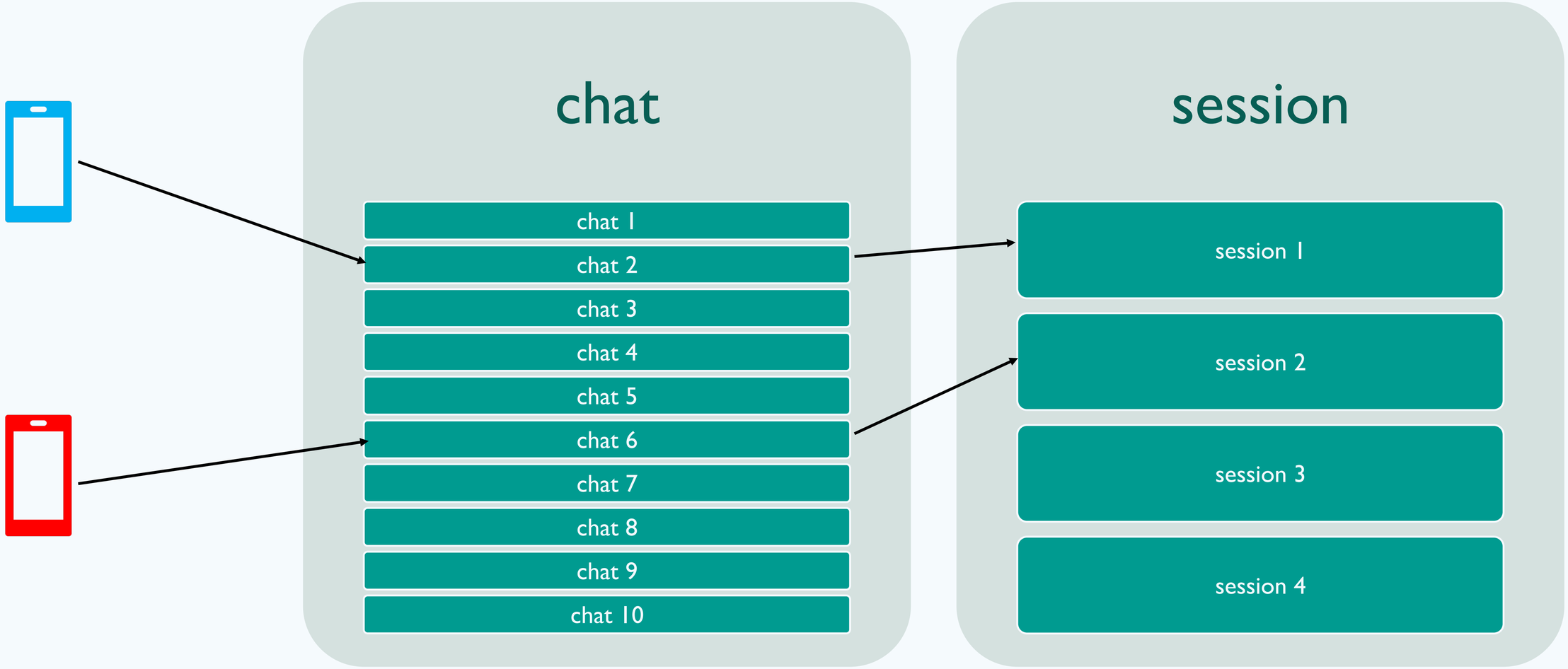
Keep It Simple

More than one process registry

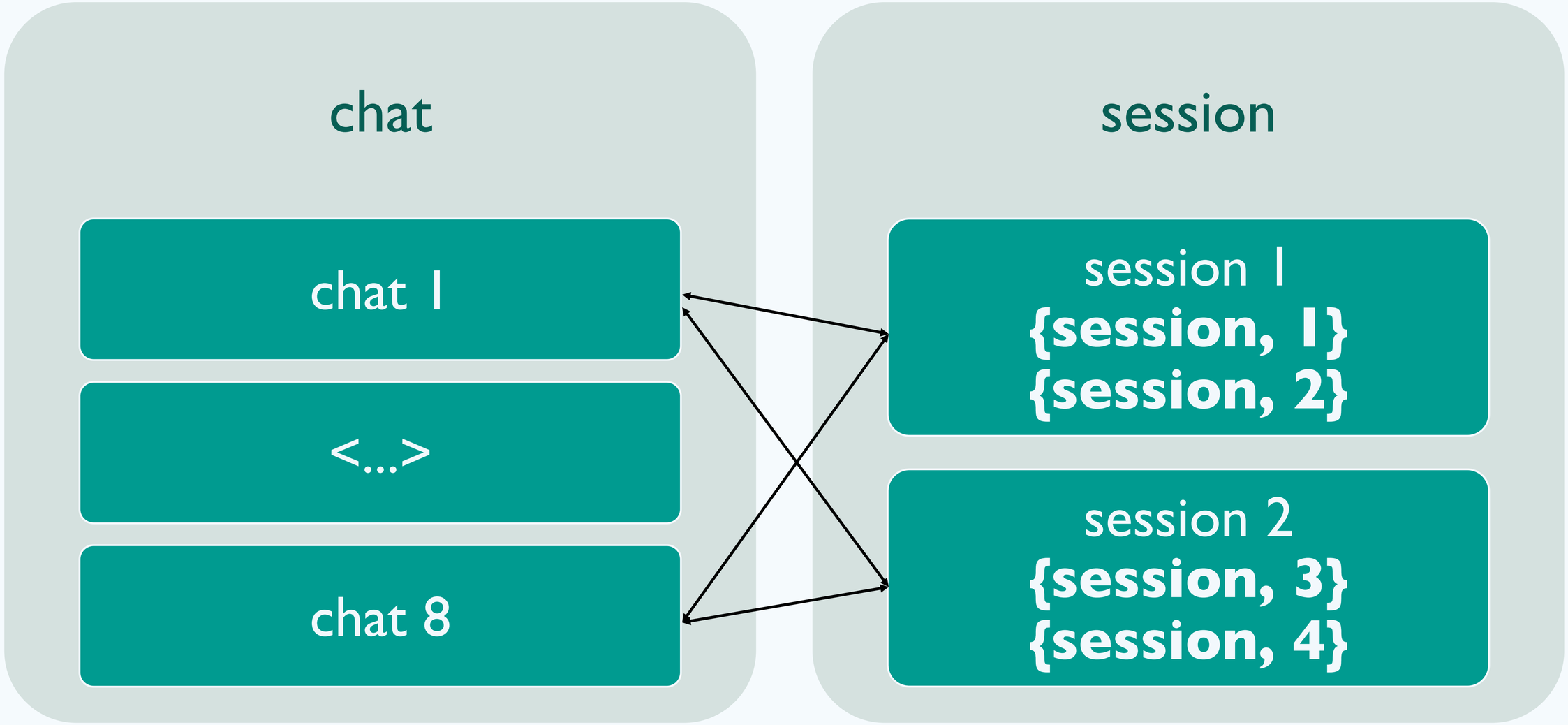
- Centralised store for high rate registrations - session manager
- Globally replicated state for rare changes - pg2

Session Manager

Central storage of phone-to-node mapping

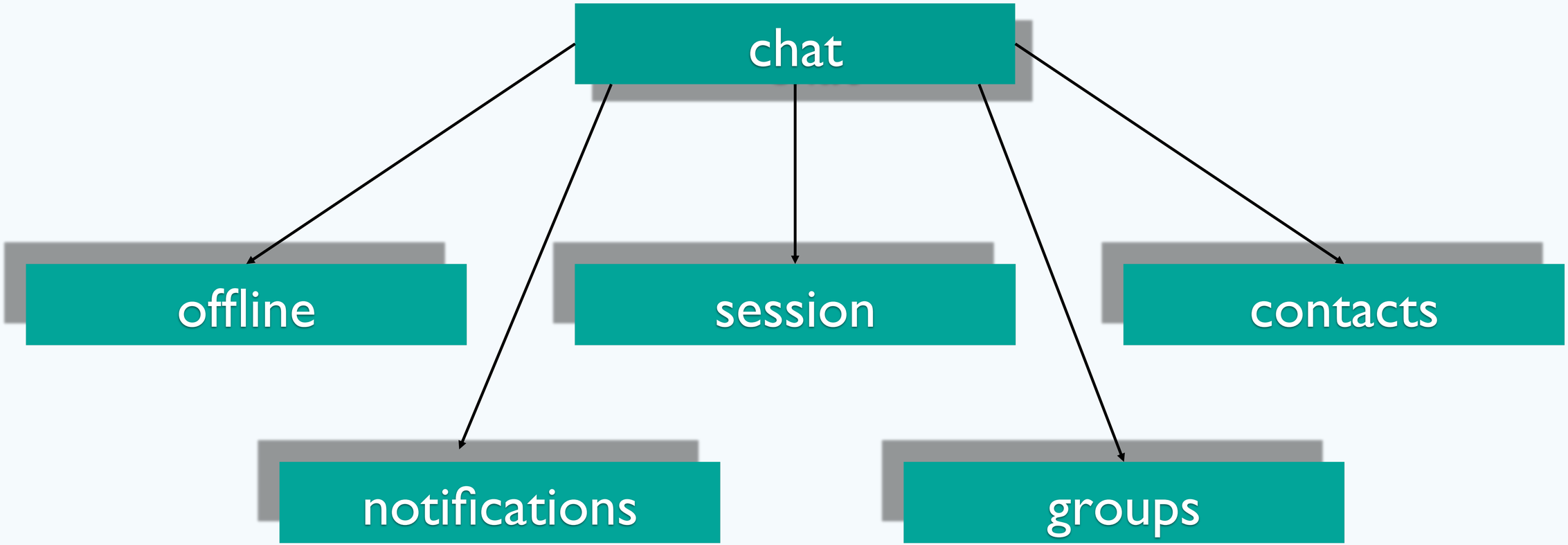


pg2 for Service Discovery



WhatsApp Meta-cluster

Meta-clustering



Limits Are Still There

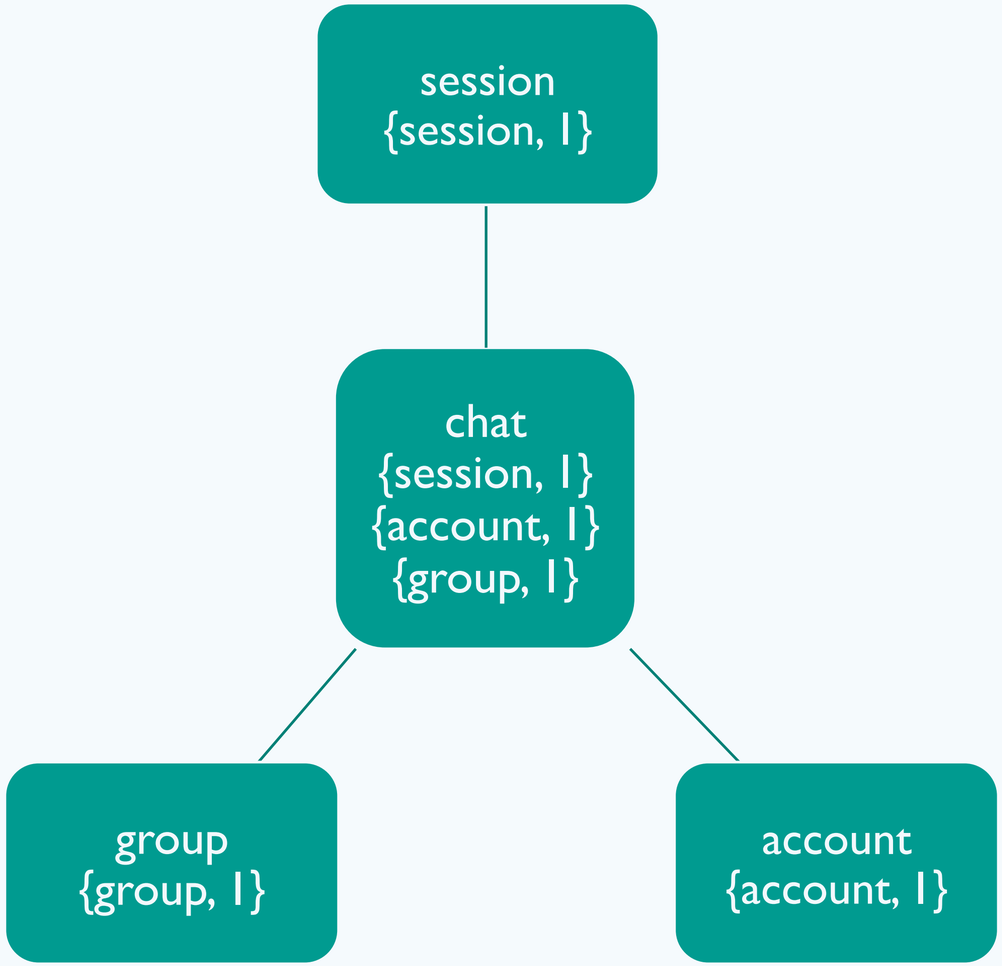
pg2 scaling is limited, but limits can be pushed further away

- Denormalise pg2_table for fast access to local and remote group members
- Apply ‘boss node’ algorithm to pg2
- Add monitoring for local processes
- … Introduce hidden (non-transitive) pg2 membership
- Pushed from 32 partitions to hundreds

wandist: Extending Erlang Distribution

Connecting disjoint Erlang clusters

- SSL support
- SOCKS proxy support
- Delivery confirmation
- Standby connections
- Maintain non-transitive pg2 lists
- Compatibility (R16 <-> R21)



Challenges

- I/O scaling – going from kqueue to epoll
  Upgrade to Erlang R21
- Routing performance – pg2 concurrent updates
  Reduce contention
- Long-range communications – increased latency
  Test with injected latency
  Absorb latency with increased concurrency
- SSL performance – handshake bottleneck
  Reduce contention

Bits & Bolts

Diagnostic Tools

- Built-in inspection: process_info, statistics, system_info
- MSACC – microstate accounting (with extra acc on)
- Lock-counting BEAM
- gdb (with etp-commands)
- BPF/BCC
- fprof, valgrind
- Erlang OTP source code!

Microstate Accounting

Thread	alloc	aux	bifbusy_wait	check_io	emulator	ets	gc	gc_full	nif	other	port	send	sleep	timers
Stats per type:														
async	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	100.00%	0.00%
aux	0.03%	0.21%	0.00%	0.00%	0.06%	0.00%	0.00%	0.00%	0.00%	0.01%	0.00%	0.00%	99.69%	0.00%
dirty_cpu_sche	0.02%	0.00%	0.00%	0.01%	0.00%	0.00%	0.00%	0.00%	0.16%	0.00%	0.00%	0.00%	99.80%	0.00%
dirty_io_sched	0.00%	0.00%	0.00%	0.01%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	99.99%	0.00%
poll	0.00%	0.00%	0.00%	0.00%	1.72%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	98.28%	0.00%
scheduler	2.73%	1.24%	3.61%	0.17%	0.17%	14.55%	4.31%	0.12%	2.51%	2.34%	2.23%	2.56%	0.95%	62.45%	0.07%

When things go wrong

Stats per type:														
async	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	100.00%	0.00%
aux	0.09%	0.08%	0.00%	0.00%	0.13%	0.00%	0.00%	0.00%	0.00%	0.00%	0.01%	0.00%	0.00%	99.70%	0.00%
dirty_cpu_sche	0.00%	0.00%	0.00%	0.01%	0.00%	0.00%	0.00%	0.00%	0.03%	0.00%	0.00%	0.00%	0.00%	99.96%	0.00%
dirty_io_sched	0.01%	0.00%	0.00%	0.26%	0.00%	0.00%	0.00%	0.00%	0.00%	0.07%	0.00%	0.00%	0.00%	99.66%	0.00%
poll	0.02%	0.00%	0.00%	0.00%	0.08%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	99.90%	0.00%
scheduler	1.39%	0.06%	0.91%	0.00%	0.07%	3.71%	0.32%	0.08%	1.20%	91.78%	0.06%	0.32%	0.09%	0.00%	0.01%
scheduler	0.76%	0.64%	0.99%	0.19%	0.08%	2.96%	51.75%	0.03%	0.47%	0.14%	1.41%	0.67%	0.35%	39.56%	0.00%
dirty_cpu_sche	0.06%	0.00%	0.00%	0.07%	0.00%	0.17%	0.00%	0.00%	81.18%	0.00%	0.00%	0.00%	0.00%	18.52%	0.00%

Lock Counting

```
(chatd@cdev0015.frc)8> lcnt:conflicts().
                  lock   id  #tries  #collisions  collisions [%]  time [us]  duration [%]
                 -----  --- ------- ------------ --------------- ---------- -------------
           crypto_stat   41  322739        46131         14.2936  196183895       860.2979
               db_tab 2684 1313726        16574          1.2616     327473         1.4360
        alcu_allocator   10  395238         1099          0.2781      60941         0.2672
             run_queue   34 4204447        47825          1.1375      51013         0.2237
              pix_lock 1024    1763           18          1.0210      48944         0.2146
          drv_ev_state 2048  230137         1383          0.6009      12459         0.0546
             proc_main 6758  898915         7493          0.8336       7341         0.0322
             proc_msgq 6758 1077268         1058          0.0982       2132         0.0093
         process_table    1   87129          138          0.1584       1819         0.0080
```

```
lock: crypto_stat
id:   []
type: rw_mutex
    location  #tries  #collisions  collisions [%]  time [us]  duration [%] histogram [log2(us)]
   --------- ------- ------------ --------------- ---------- ------------ --------------------
 undefined:0   23251        18582         79.9191   90225533      685.5644 |      ..........x...XXx..     |
```

Lock Counting & Source Code

```
            for (i=nlocks-1; i>=0; --i) {
-               lock_vec[i] = enif_rwlock_create("crypto_stat");
+               snprintf(lock_name, sizeof(lock_name), "crypto_%s", CRYPTO_get_lock_name(i));
+               lock_vec[i] = enif_rwlock_create(lock_name);
                if (lock_vec[i]==NULL) return NULL;
```

Meaningful lock name

```
(chatd@cdev0015.frc)3> lcnt:conflicts().
            lock    id   #tries  #collisions  collisions [%]  time [us]  duration [%]
            -----   ---  -------  -----------  --------------  ---------  ------------
      crypto_rsa     1   280851       190678         67.8929  899670045      617.4626
          db_tab  9282 13009375       132871          1.0213    5075031        3.4831
       run_queue    34 51182832       771052          1.5065     987565        0.6778
```

Lock Counting

```
      lock      id   #tries  #collisions  collisions [%]  time [us]  duration [%]
     -----     ---  -------  -----------  --------------  ----------  ------------
    db_tab    3630 70515587        78828          0.1118   572328685     3321.0877
```

Name is already there!

```
.whatsapp.net)11> lcnt:inspect(db_tab, [{print, [id, colls, ratio, duration]}]).
             id  #collisions  collisions [%]  duration [%]
            --- ------------ --------------- -------------
    wa_wandist         69671          7.5119     3313.6525
     session_14            79          0.1992        2.5627
```

BCC (BPF Compiler Collection)

```
/usr/local/bcc/bin/trace.py -tp 277465 'sys_read (arg3 > 20000) "read %d bytes", arg3'
```

Trace large reads

```
TIME     PID     TID     COMM            FUNC             -
2.061669 277465  277485  5_scheduler     sys_read         read 65536 bytes
2.061934 277465  277485  5_scheduler     sys_read         read 65536 bytes
2.062264 277465  277495  15_scheduler    sys_read         read 65536 bytes
2.062621 277465  277495  15_scheduler    sys_read         read 65536 bytes
2.062835 277465  277495  15_scheduler    sys_read         read 65536 bytes
2.063099 277465  277485  5_scheduler     sys_read         read 65536 bytes
2.063357 277465  277496  16_scheduler    sys_read         read 65536 bytes
2.063615 277465  277496  16_scheduler    sys_read         read 65536 bytes
```

gdb + etp-commands

Not necessarily post-mortem

```
9644          -                                   & !(state & ERTS_PSFLG_SUSPENDED));
      9644    +                                   & !(state & (ERTS_PSFLG_SUSPENDED|ERTS_PSFLGS_DIRTY_WORK)));
```

```
etp-1                          etp-boxed-immediate-1      etp-cp-func-info-1         etp-fds
etp-address-to-beam-opcode     etp-carrier-blocks         etp-ct-atom-1              etp-float-1
etp-alloc-instances            etp-char-1                 etp-ct-name-1              etp-heapdump
etp-alloc-stats                etp-chart                  etp-ct-printable-1         etp-heapdump-1
etp-array-1                    etp-chart-entry-1          etp-ct-variable-1          etp-heapdump-old
etp-atom-1                     etp-chart-print            etp-dictdump               etp-help
etp-aux-work-flags             etp-chart-start            etp-disasm                 etp-id2port
etp-bignum-1                   etp-check-beam-ranges      etp-disasm-1               etp-id2port-1
etp-bitmap-array-1             etp-compile                etp-ets-obj                etp-immediate-1
etp-block                      etp-compile-debug          etp-ets-tabledump          etp-init
etp-block-size-1               etp-compile-info           etp-ets-tables             etp-lc-dump
etp-block2mbc                  etp-config-h-info          etp-extpid-1               etp-list-1
etp-block2mbc-1                etp-cp                     etp-extport-1              etp-list-2
etp-boxed-1                    etp-cp-1                   etp-extref-1               etp-list-printable-1
```

Erlang OTP is getting better

Some original WhatsApp patches are no longer in use

- GC Throttling -> Off-Heap message queue
- prim_file patches -> built-in NIF-based file I/O
- TLS 1.2 support (cipher suite selection)
- HW-accelerated crypto
- public_key: PKCS8 support, certificate verification, SNI
- Hashing clashes (ETS to mnesia)
- Bugfixes!

But Not There Yet

Recently added and reworked patches

- SSL/TLS handshake acceleration, PEM cache validation
- inet_db: race condition during .hosts file reload
- prim_inet: race/suboptimal accept() behaviour
- prepend send (stuck worker detection, TTL)
- flush process message queue
- process message/signal queue stats

But Not There Yet

Recently added and reworked patches

- system monitoring (signal queues, rpc tracing)
- httpc_client TLS upgrade timeout
- wider lock tables (check IO, ETS meta)
- worker pools, dispatcher pools
- convenience patches (noisy logging suppression, default eunit timeouts, listen backlog queues sizes, pretty printing, shell history, supervisor ‘ETS-TRANSFER’)

Embrace Open Source community

- Upstream our patches
- Be Open!

Questions?

Maxim Fedorov
dane@ whatsapp.com
GitHub: max-au

Paradigm Shift

- Per-node monitoring -> cluster health
- Years of uptime -> simple & reliable restart
- Trigger-based alerts -> level-based
- Local configuration -> global
- Aggregates and centralised logging