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)
Output scale

Messages, Notifications, & Presence
### Throughput scale

(4 of 16 partitions)

<table>
<thead>
<tr>
<th>psh311</th>
<th>ERL</th>
<th>msg------------------</th>
<th>dist------------------</th>
<th>wan------------------</th>
</tr>
</thead>
<tbody>
<tr>
<td>time</td>
<td>nodes</td>
<td>qlen</td>
<td>qmax</td>
<td>nzq</td>
</tr>
<tr>
<td>02/25 07:30:01</td>
<td>408</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>02/25 08:00:00</td>
<td>408</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>02/25 08:30:01</td>
<td>408</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>02/25 09:00:01</td>
<td>408</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>02/25 09:30:00</td>
<td>408</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

(2 of 16 partitions)

<table>
<thead>
<tr>
<th>prs101</th>
<th>ERL</th>
<th>msg-----------------</th>
<th>dist-------------</th>
<th>mnes-----------------</th>
<th>sched</th>
<th>mem---</th>
</tr>
</thead>
<tbody>
<tr>
<td>time</td>
<td>nodes</td>
<td>qlen</td>
<td>qmax</td>
<td>recv</td>
<td>msgin</td>
<td>msgout</td>
</tr>
<tr>
<td>02/24 10:00:00</td>
<td>400</td>
<td>0</td>
<td>0</td>
<td>357383</td>
<td>174975</td>
<td>104489</td>
</tr>
<tr>
<td>02/24 10:30:00</td>
<td>400</td>
<td>0</td>
<td>0</td>
<td>352178</td>
<td>172389</td>
<td>102970</td>
</tr>
<tr>
<td>02/24 11:00:01</td>
<td>400</td>
<td>0</td>
<td>0</td>
<td>347643</td>
<td>170111</td>
<td>101688</td>
</tr>
<tr>
<td>02/24 11:30:01</td>
<td>400</td>
<td>0</td>
<td>0</td>
<td>341300</td>
<td>167085</td>
<td>99822</td>
</tr>
</tbody>
</table>
(1 of 16 partitions)

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

- Small percentage of hash buckets being used
- ets uses *average* chain length to trigger split

```c
#define MAX_HASH 0xEFFFFFFFFFUL
#define INVALID_HASH 0xFFFFFFFFFUL
#define HASH_INITVAL 33554467UL

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

- DC1 main cluster
- DC2 main cluster
- DC1 mms cluster
- DC2 mms cluster
- Global clusters
Topology

DC1 main cluster

DC1 mms cluster

Acct cluster

DC1

DC2

DC2 mms cluster
Routing

Cluster 1

{last,1}
{last,2}
{last,3}
{last,4}

Cluster 2

{last,1}
{last,2}
{last,3}
{last,4}

service client

wandist

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

CPU (incl. Nice) Utilization

%util

02/24 12PM 02/25 12AM 02/25 12PM 02/26 12AM 02/26 12PM 02/27 12AM
Monitor/Measure
Monitor/Measure

Disk Busy

% disk busy

02/24 12PM 02/25 12AM 02/25 12PM 02/26 12AM 02/26 12PM 02/27 12AM
Monitor/Measure

Disk Write Bandwidth (average per drive)
Monitor/Measure
Monitor/Measure
Input scaling

Logins