About ...

- Joined WhatsApp in 2011
- New to Erlang
- Background in performance of C-based systems on FreeBSD and Linux
- Prior work at Yahoo!, SGI
Overview

- The “good problem to have”
- Performance Goals
- Tools and Techniques
- Results
- General Findings
- Specific Scalability Fixes
The Problem

- A good problem, but a problem nonetheless
- Growth, Earthquakes, and Soccer!

- Mexican earthquake

Msg rates for past four weeks

HT

FT

goals
The Problem

- Initial server loading: ~200k connections
- Discouraging prognosis for growth
- Cluster brittle in the face of failures/overloads
Performance Goals

- 1 Million connections per server ... !
- Resilience against disruptions under load
  - Software failures
  - Hardware failures (servers, network gear)
  - World events (sports, earthquakes, etc.)
Performance Goals

- Our standard configuration
  - Dual Westmere Hex-core (24 logical CPUs)
  - 100GB RAM, SSD
  - Dual NIC (user-facing, back-end/distribution)
  - FreeBSD 8.3
  - OTP R14B03
Tools and Techniques

- System activity monitoring (wsar)
- OS-level

- BEAM
Tools and Techniques

- Processor hardware perf counters (pmcstat)
- dtrace, kernel lock-counting, gprof
Tools and Techniques

- fprof (w/ and w/o cpu_timestamp)

<table>
<thead>
<tr>
<th>Function</th>
<th>Total</th>
<th>Time</th>
<th>%</th>
<th>Total</th>
<th>%</th>
<th>Total</th>
<th>%</th>
</tr>
</thead>
<tbody>
<tr>
<td>lists,do_flatten,2</td>
<td>234817</td>
<td>758.760</td>
<td>4.24%</td>
<td>4.24%</td>
<td>561.161</td>
<td></td>
<td></td>
</tr>
<tr>
<td>wap_j_i,decode_attr,3</td>
<td>59673</td>
<td>425.213</td>
<td>2.46%</td>
<td>6.64%</td>
<td>996.159</td>
<td></td>
<td></td>
</tr>
<tr>
<td>wap_j_i,encode_keyword_or_binary,1</td>
<td>84821</td>
<td>325.764</td>
<td>1.84%</td>
<td>8.48%</td>
<td>456.088</td>
<td></td>
<td></td>
</tr>
<tr>
<td>wap_j_i,encode_keyword_or_binary,1</td>
<td>73166</td>
<td>313.390</td>
<td>1.77%</td>
<td>10.25%</td>
<td>624.413</td>
<td></td>
<td></td>
</tr>
<tr>
<td>lists,map,2</td>
<td>72372</td>
<td>312.713</td>
<td>1.77%</td>
<td>12.43%</td>
<td>2369.478</td>
<td></td>
<td></td>
</tr>
<tr>
<td>envelope.parse_xml,1</td>
<td>38440</td>
<td>171.991</td>
<td>1.54%</td>
<td>13.55%</td>
<td>628.768</td>
<td></td>
<td></td>
</tr>
<tr>
<td>wib,hex,2</td>
<td>51382</td>
<td>155.198</td>
<td>1.44%</td>
<td>14.95%</td>
<td>276.617</td>
<td></td>
<td></td>
</tr>
<tr>
<td>charx.router,'-route_to_node/4-1$^\theta/^-0^-$',2</td>
<td>75977</td>
<td>132.190</td>
<td>1.31%</td>
<td>15.28%</td>
<td>162.912</td>
<td></td>
<td></td>
</tr>
<tr>
<td>envelope.from_xml,2</td>
<td>23798</td>
<td>136.361</td>
<td>1.30%</td>
<td>17.60%</td>
<td>1416.912</td>
<td></td>
<td></td>
</tr>
<tr>
<td>xml.get_attr,5,3</td>
<td>55258</td>
<td>201.735</td>
<td>1.14%</td>
<td>18.74%</td>
<td>278.513</td>
<td></td>
<td></td>
</tr>
<tr>
<td>lists,keysearch,3</td>
<td>96324</td>
<td>201.082</td>
<td>1.14%</td>
<td>19.88%</td>
<td>281.082</td>
<td></td>
<td></td>
</tr>
<tr>
<td>wap,decode_attr,1</td>
<td>33905</td>
<td>195.083</td>
<td>1.21%</td>
<td>21.66%</td>
<td>785.417</td>
<td></td>
<td></td>
</tr>
<tr>
<td>binary_stream,parse,2</td>
<td>26303</td>
<td>198.411</td>
<td>1.12%</td>
<td>22.12%</td>
<td>2413.095</td>
<td></td>
<td></td>
</tr>
<tr>
<td>xml.normalize_attr,4</td>
<td>38453</td>
<td>194.789</td>
<td>1.10%</td>
<td>23.22%</td>
<td>314.697</td>
<td></td>
<td></td>
</tr>
<tr>
<td>wap,decode,thing,1</td>
<td>48607</td>
<td>185.697</td>
<td>1.06%</td>
<td>24.27%</td>
<td>243.223</td>
<td></td>
<td></td>
</tr>
<tr>
<td>envelope.fill_element,1</td>
<td>25950</td>
<td>164.017</td>
<td>1.01%</td>
<td>25.31%</td>
<td>456.118</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

- BEAM lock-counting (invaluable!!!)
Tools and Techniques

- Synthetic workload
  - Good for subsystems with simple interfaces
  - Limited value for user-facing systems
Tools and Techniques

- Tee'd workload
  - Where side-effects can be contained
  - Extremely useful for tuning
Tools and Techniques

- Diverted workload
  - Add additional production load to server
  - DNS via extra IP aliases
    - TTL issues
  - IPFW forwarding
    - Ran into a few kernel panics at high conn counts
### Results

- **Initial bottlenecks appeared around 425k**
- **First round of fixes got us to 1M conns**
- **Fruit was hanging pretty low**

```
<table>
<thead>
<tr>
<th>CPU</th>
<th>TCP</th>
<th>send</th>
<th>recv</th>
<th>listen</th>
<th>pcb</th>
<th>conn</th>
<th>VM</th>
</tr>
</thead>
<tbody>
<tr>
<td>72.4 47.4</td>
<td>8213 8004</td>
<td>3.34</td>
<td>50816 4941</td>
<td>10086672 1414</td>
<td>75.1 62.6 0.0 0.0 12.5 24.3 998760</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>time</th>
<th>Util</th>
<th>User</th>
<th>Nice</th>
<th>Sys</th>
<th>Xintr</th>
<th>Xidle</th>
<th>spkt/s</th>
<th>skb/s</th>
<th>% rpkt/s</th>
<th>skb/s</th>
<th>ov/lw</th>
<th>count</th>
<th>/sec</th>
<th>Util</th>
<th>%Act</th>
<th>%Inac</th>
<th>%Xach</th>
<th>%Wire</th>
<th>%Free</th>
<th>files</th>
</tr>
</thead>
<tbody>
<tr>
<td>09/22 13:44:07</td>
<td>73.2 47.8</td>
<td>0.0</td>
<td>20.1</td>
<td>5.3</td>
<td>26.7</td>
<td>81292</td>
<td>8032</td>
<td>5.31</td>
<td>49977</td>
<td>4915</td>
<td>10088956</td>
<td>1493</td>
<td>75.4</td>
<td>62.9</td>
<td>0.0</td>
<td>0.0</td>
<td>12.5</td>
<td>24.3</td>
<td>998760</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>CPU</th>
<th>TCP</th>
<th>send</th>
<th>recv</th>
<th>listen</th>
<th>pcb</th>
<th>conn</th>
<th>VM</th>
</tr>
</thead>
<tbody>
<tr>
<td>72.8 47.5</td>
<td>8236 8086</td>
<td>1.33</td>
<td>50831</td>
<td>4918</td>
<td>1001250 1412</td>
<td>75.9 63.4</td>
<td>0.0</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>time</th>
<th>Util</th>
<th>User</th>
<th>Nice</th>
<th>Sys</th>
<th>Xintr</th>
<th>Xidle</th>
<th>spkt/s</th>
<th>skb/s</th>
<th>% rpkt/s</th>
<th>skb/s</th>
<th>ov/lw</th>
<th>count</th>
<th>/sec</th>
<th>Util</th>
<th>%Act</th>
<th>%Inac</th>
<th>%Xach</th>
<th>%Wire</th>
<th>%Free</th>
<th>files</th>
</tr>
</thead>
<tbody>
<tr>
<td>09/22 13:45:07</td>
<td>72.8 47.5</td>
<td>0.0</td>
<td>20.0</td>
<td>5.3</td>
<td>37.2</td>
<td>82368</td>
<td>8086</td>
<td>1.33</td>
<td>50831</td>
<td>4918</td>
<td>1001250</td>
<td>1412</td>
<td>75.9</td>
<td>63.4</td>
<td>0.0</td>
<td>0.0</td>
<td>12.5</td>
<td>24.3</td>
<td>9988870</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>time</th>
<th>Util</th>
<th>User</th>
<th>Nice</th>
<th>Sys</th>
<th>Xintr</th>
<th>Xidle</th>
<th>spkt/s</th>
<th>skb/s</th>
<th>% rpkt/s</th>
<th>skb/s</th>
<th>ov/lw</th>
<th>count</th>
<th>/sec</th>
<th>Util</th>
<th>%Act</th>
<th>%Inac</th>
<th>%Xach</th>
<th>%Wire</th>
<th>%Free</th>
<th>files</th>
</tr>
</thead>
<tbody>
<tr>
<td>09/22 13:46:08</td>
<td>73.7 48.1</td>
<td>0.0</td>
<td>20.2</td>
<td>5.3</td>
<td>26.3</td>
<td>81290</td>
<td>7946</td>
<td>3.40</td>
<td>49433</td>
<td>4853</td>
<td>10094420</td>
<td>1415</td>
<td>76.0</td>
<td>63.5</td>
<td>0.0</td>
<td>0.0</td>
<td>12.5</td>
<td>24.0</td>
<td>10002723</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>time</th>
<th>Util</th>
<th>User</th>
<th>Nice</th>
<th>Sys</th>
<th>Xintr</th>
<th>Xidle</th>
<th>spkt/s</th>
<th>skb/s</th>
<th>% rpkt/s</th>
<th>skb/s</th>
<th>ov/lw</th>
<th>count</th>
<th>/sec</th>
<th>Util</th>
<th>%Act</th>
<th>%Inac</th>
<th>%Xach</th>
<th>%Wire</th>
<th>%Free</th>
<th>files</th>
</tr>
</thead>
<tbody>
<tr>
<td>09/22 13:47:07</td>
<td>72.7 47.3</td>
<td>0.0</td>
<td>20.0</td>
<td>5.4</td>
<td>27.3</td>
<td>83318</td>
<td>8115</td>
<td>1.47</td>
<td>50744</td>
<td>4948</td>
<td>1007680</td>
<td>1451</td>
<td>76.3</td>
<td>63.7</td>
<td>0.0</td>
<td>0.0</td>
<td>12.5</td>
<td>23.7</td>
<td>10004766</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>time</th>
<th>Util</th>
<th>User</th>
<th>Nice</th>
<th>Sys</th>
<th>Xintr</th>
<th>Xidle</th>
<th>spkt/s</th>
<th>skb/s</th>
<th>% rpkt/s</th>
<th>skb/s</th>
<th>ov/lw</th>
<th>count</th>
<th>/sec</th>
<th>Util</th>
<th>%Act</th>
<th>%Inac</th>
<th>%Xach</th>
<th>%Wire</th>
<th>%Free</th>
<th>files</th>
</tr>
</thead>
<tbody>
<tr>
<td>09/22 13:48:08</td>
<td>72.6 47.3</td>
<td>0.0</td>
<td>19.9</td>
<td>5.3</td>
<td>27.4</td>
<td>81392</td>
<td>7929</td>
<td>3.40</td>
<td>49421</td>
<td>4831</td>
<td>1009899</td>
<td>1418</td>
<td>76.3</td>
<td>63.8</td>
<td>0.0</td>
<td>0.0</td>
<td>12.5</td>
<td>25.3</td>
<td>10005356</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>time</th>
<th>Util</th>
<th>User</th>
<th>Nice</th>
<th>Sys</th>
<th>Xintr</th>
<th>Xidle</th>
<th>spkt/s</th>
<th>skb/s</th>
<th>% rpkt/s</th>
<th>skb/s</th>
<th>ov/lw</th>
<th>count</th>
<th>/sec</th>
<th>Util</th>
<th>%Act</th>
<th>%Inac</th>
<th>%Xach</th>
<th>%Wire</th>
<th>%Free</th>
<th>files</th>
</tr>
</thead>
<tbody>
<tr>
<td>09/22 13:49:08</td>
<td>73.4 47.9</td>
<td>0.0</td>
<td>20.1</td>
<td>5.4</td>
<td>26.6</td>
<td>82539</td>
<td>8068</td>
<td>3.39</td>
<td>50493</td>
<td>4927</td>
<td>1101326</td>
<td>1428</td>
<td>76.4</td>
<td>63.8</td>
<td>0.0</td>
<td>0.0</td>
<td>12.6</td>
<td>23.6</td>
<td>10007721</td>
<td></td>
</tr>
</tbody>
</table>

---

<table>
<thead>
<tr>
<th>info</th>
<th>output</th>
<th>sched</th>
<th>gc</th>
<th>mem</th>
<th>nproc</th>
</tr>
</thead>
<tbody>
<tr>
<td>mng</td>
<td>mngout</td>
<td>load</td>
<td>runq</td>
<td>Util</td>
<td>100%</td>
</tr>
<tr>
<td>253438</td>
<td>57872</td>
<td>3406</td>
<td>1830</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>25263</td>
<td>57597</td>
<td>3472</td>
<td>1820</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>25447</td>
<td>57909</td>
<td>3403</td>
<td>1823</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>25070</td>
<td>57155</td>
<td>3434</td>
<td>1792</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>25009</td>
<td>58732</td>
<td>3492</td>
<td>1812</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>25090</td>
<td>57264</td>
<td>3485</td>
<td>1779</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>25473</td>
<td>58031</td>
<td>3482</td>
<td>1814</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>
Results

- Continued attacking similar bottlenecks
- Achieved 2M conns about a month later
- Put further optimizations on back burner
Results

- Began optimizing app code after New Years
- Unintentional record attempt in Feb
- Peaked at 2.8M conns before we intervened

571k pkts/sec, >200k dist msgs/sec
Results

- Still trying to obtain elusive 3M conns
- St. Patrick's Day wasn't as lucky as hoped
General Findings

- Erlang has awesome SMP scalability
  - >85% cpu utilization across 24 logical cpus
  - FreeBSD shines as well
General Findings

CPU% vs. # Conns
General Findings

- Contention, contention, contention
  - From 200k to 2M were all contention fixes
  - Some issues are internal to BEAM
    - Some addressable with app changes
    - Most required BEAM patches
  - Some required app changes
    - Especially: partitioning workload correctly
    - Some common Erlang idioms come at a price
Specific Scalability Fixes

- FreeBSD
  - Backported TSC-based kernel timecounter
    - gettimeofday(2) calls much less expensive
  - Backported igb network driver
    - Had issues with MSI-X queue stalls
  - sysctl tuning
    - Obvious limits (e.g., kern.ipc.maxsockets)
    - net.inet.tcp.tcphashsize=524288
Specific Scalability Fixes

- **BEAM metrics**
  - Scheduler (%util, csw, waits, sleeps, …)
  - statistics(message_queues)
    - Msgs queued, #non-empty queues, longest queue
  - process_info(message_queue_stats)
    - Enq/deq/send count & rates (1s, 10s, 100s)
  - statistics(message_counts)
    - Aggregation of message_queue_stats
  - Enable fprof cpu_timestamp for FreeBSD
Specific Scalability Fixes

- **BEAM metrics (cont.)**
  - Make lock-counting work for larger async thread counts (e.g., +A 1024)
  - Add suspend, location, and port_locks options to erts_debug:lock_counters
  - Enable/disable process/port lock counting at runtime
  - Fix missing accounting for outbound dist bytes
Specific Scalability Fixes

- **BEAM tuning**
  - +swt low
    - Avoid scheduler perma-sleep
  - +Mummc/mmmmbc/mmsbc 99999
    - Prefer mseg over malloc
  - +Mut 24
    - Want allocator instance per scheduler
Specific Scalability Fixes

- BEAM tuning
  - +Mulmbcs 32767 +Mumbcgs 1
  - +Musmbcs 2047
    - Want large 2M-aligned mseg allocations to maximize superpage promotions
  - Run with real-time scheduling priority
  - +ssct 1 (via patch; scheduler spin count)
Specific Scalability Fixes

- BEAM contention
  - `timeofday` lock (esp., `timeofday` delivery)
  - Reduced slot traversals on timer wheel
  - Widened bif timer hash table
    - Ended up moving bif timers to receive timeouts
  - Improved `check_io` allocation scalability
  - Added `prim_file:write_file/3 & /4` (port reuse)
  - Disable `mseg` max check
Specific Scalability Fixes

- BEAM contention (cont.)
  - Reduce setopts calls in prim_inet:accept and in inet:tcp_controlling_process
Specific Scalability Fixes

- OTP throughput
  - Add gc throttling when message queue is long
  - Increase default dist receive buffer from 4k to 256k (and make configurable)
  - Patch mnesia_tm to dispatch async_dirty txns to separate per-table procs for concurrency
  - Add pg2 denormalized group member lists to improve lookup throughput
  - Increase max configurable mseg cache size
Specific Scalability Fixes

- **Erlang usage**
  - Prefer os:timestamp to erlang:now
  - Implement cross-node gen_server calls without using monitors (reduces dist traffic and proc link lock contention)
  - Partition ets and mnesia tables and localize access to smaller number of processes
  - Small mnesia clusters
Specific Scalability Fixes

Operability fixes

- Added [prepend] option to erlang:send
- Added process_flag(flush_message_queue)
Questions? Comments?

- rr@whatsapp.com