Agenda

- The Problem
- Concepts
- Statistics
- Case 1: Large binaries
- Case 2: Fragmentation
- New Features
The Problem

- Normal OS default allocator:
  - is relatively slow for many small allocations
  - uses same allocation strategy for all data, increased fragmentation
  - no cross platform fine-grained statistics
- Try it on your system, +Mea min
  - Disables erts allocators and uses malloc directly for everything
- With multi-core memory management is even more important (and even more difficult)
Concepts

- Carriers and Blocks
- Single– vs multi–block carriers
- Multiblock allocators
- Thread specific allocators
Carriers and Blocks

- **Block** – A piece of memory requested by the VM
- **Carrier** – A piece of memory that contains one or more blocks

```erlang
ets:insert(Tid,0,“HELLO”).
ets:insert(Tid,1,[0,1,2,...,63]).
ets:insert(Tid,2,[0,1,2,...,128]).
ets:delete(Tid,1).
```
Single- vs Multi-block Carriers

- Large blocks are placed in a singleblock carrier (sbc)
  - What is a large block? depends...
    - Control with +M<sbct (singleblock carrier threshold)
    - default is 512 kb
- Normally you want most of your data in multiblock carriers (mbc)
  - If you increase sbct you probably want to increase smbc<cs and lmbcs by an equal %
  - Size of carrier is controlled with +M<smbcs, +M<lmbcs and +M<mbcgs
Allocator types

- Different strategies possible for different types of data
  - eheap, binary, driver, ets
  - temporary, short lived, standard lived, long lived
  - fix size
Allocator types

• temporary
  – C function scope
  – temp gc rootset
  – dist msg decode

• standard
  – links
  – monitors
    • fixed
      – process control block
      – port control block

• short
  – ets match specs
  – short timers
  – fd select list

• long
  – code
  – atoms
Multiblock allocator strategies

- **Block oriented**
  - best fit
  - address order best fit
  - address order first fit
  - good fit
  - a fit

- **Carrier oriented**
  - address order first fit
  - carrier best fit
  - address order first fit
  - carrier address order best fit
Best fit example
Best fit example
Carrier Allocators

- mseg alloc
  - uses /dev/zero and mmap,munmap,mremap
  - caches freed carriers
- sys alloc
  - maps to malloc,free (or posix_memalign/free)
  - carriers are a multiple of +Muycs to help avoid fragmentation
  - used for main carrier allocation +M<S>mmbcs
Memory architecture

Thread 1 ...

sl_alloc
eheap_alloc
binary_alloc

ets_alloc
mseg_alloc

erts_mmap

Scheduler 1 ...

sl Alloc
eheap_alloc
binary_alloc

ets_alloc
mseg_alloc

Scheduler 2 ...

sl Alloc
eheap_alloc
binary_alloc

ets_alloc
mseg_alloc

Scheduler N ...

sl Alloc
eheap_alloc
binary_alloc

ets_alloc
mseg_alloc

Pool
Statistics: `erlang:system_info(allocator)`

- Allocator types
  - `sys_alloc` `mseg_alloc`
  - `eheap_alloc` `ets_alloc` `binary_alloc` `driver_alloc`
  - `temp_alloc` `sl_alloc` `std_alloc` `ll_alloc` `fix_alloc`
- Settings
  - sbc threshold, mbc allocation strategy, etc
- Features
  - aligned alloc, lock physical, etc
Statistics: \texttt{erlang:system_info(\{allocator,Type\})}

\[
\{\text{instance}, 0, \\
  \{\text{versions}, \ldots \}, \\
  \{\text{options}, \ldots \}, \\
  \{\text{mbcs}, \ldots \}, \\
  \{\text{sbc}, \ldots \}, \\
  \{\text{calls}, \ldots \}\}, \\
\{\text{instance}, 1, \ldots \}, \\
\{\text{instance}, S+1, \ldots \}\}
\]
Statistics: mbcs / sbcs

```
[{blocks, 1066675, 1068988, 1811013},
 {blocks_size, 860267920, 862367120, 3546346384},
 {carriers, 455, 455, 455},
 {carriers_size, 3763863552, 3763863552, 3763863552}],
```
## Statistics: example mbc

<table>
<thead>
<tr>
<th></th>
<th>Current</th>
<th>Max (Last)</th>
<th>Max</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>blocks</strong></td>
<td>1066675</td>
<td>1068988</td>
<td>1811013</td>
</tr>
<tr>
<td><strong>blocks_size</strong></td>
<td>860267920</td>
<td>860267920</td>
<td>3546346384</td>
</tr>
<tr>
<td><strong>carriers</strong></td>
<td>455</td>
<td>455</td>
<td>455</td>
</tr>
<tr>
<td><strong>carriers_size</strong></td>
<td>3763863552</td>
<td>3763863552</td>
<td>3763863552</td>
</tr>
</tbody>
</table>
## Statistics: example mbc

<table>
<thead>
<tr>
<th></th>
<th>Current</th>
<th>Max (Last)</th>
<th>Max</th>
</tr>
</thead>
<tbody>
<tr>
<td>blocks</td>
<td>1066675</td>
<td>1068988</td>
<td>1811013</td>
</tr>
<tr>
<td>blocks_size</td>
<td>820 MB</td>
<td>820 MB</td>
<td>3382 MB</td>
</tr>
<tr>
<td>carriers</td>
<td>455</td>
<td>455</td>
<td>455</td>
</tr>
<tr>
<td>carriers_size</td>
<td>3590 MB</td>
<td>3590 MB</td>
<td>3590 MB</td>
</tr>
</tbody>
</table>
## Statistics: example sbc

<table>
<thead>
<tr>
<th></th>
<th>Current</th>
<th>Max (Last)</th>
<th>Max</th>
</tr>
</thead>
<tbody>
<tr>
<td>blocks</td>
<td>6</td>
<td>6</td>
<td>21</td>
</tr>
<tr>
<td>blocks_size</td>
<td>6 MB</td>
<td>6 MB</td>
<td>20 MB</td>
</tr>
<tr>
<td>carriers</td>
<td>6</td>
<td>6</td>
<td>21</td>
</tr>
<tr>
<td>carriers_size</td>
<td>7.5 MB</td>
<td>7.5 MB</td>
<td>25 MB</td>
</tr>
</tbody>
</table>
Statistics: calls

<table>
<thead>
<tr>
<th></th>
<th>alloc</th>
<th>free</th>
<th>realloc</th>
</tr>
</thead>
<tbody>
<tr>
<td>binary</td>
<td>28379577160</td>
<td>28378510479</td>
<td>985494638</td>
</tr>
<tr>
<td>mseg</td>
<td>24186</td>
<td>23725</td>
<td>6839</td>
</tr>
<tr>
<td>sys</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>
## Statistics: calls

<table>
<thead>
<tr>
<th></th>
<th>alloc</th>
<th>free</th>
<th>realloc</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>binary</strong></td>
<td>28380 MC</td>
<td>28379 MC</td>
<td>985 MC</td>
</tr>
<tr>
<td><strong>mseg</strong></td>
<td>24186</td>
<td>23725</td>
<td>6839</td>
</tr>
<tr>
<td><strong>sys</strong></td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>
## Statistics: mseg

<table>
<thead>
<tr>
<th>Statistic</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>cached segments</td>
<td>2</td>
</tr>
<tr>
<td>cache hits</td>
<td>424</td>
</tr>
<tr>
<td>segments</td>
<td>12</td>
</tr>
<tr>
<td>segments_size</td>
<td>12136448</td>
</tr>
<tr>
<td>segments_watermark</td>
<td>4</td>
</tr>
<tr>
<td>mseg alloc</td>
<td>464</td>
</tr>
<tr>
<td>mseg dealloc</td>
<td>452</td>
</tr>
<tr>
<td>mseg create</td>
<td>40</td>
</tr>
<tr>
<td>mseg destroy</td>
<td>32</td>
</tr>
</tbody>
</table>
Case studies

- Case 1: Large binaries
- Case 2: Fragmentation
Case 1: Large binaries

- Symptoms
  - used strace to find that many more malloc than mmap were made
## Case 1: Large binaries

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>calls binary_alloc</td>
<td>321 MC</td>
</tr>
<tr>
<td>calls mseg_alloc</td>
<td>0.4 MC</td>
</tr>
<tr>
<td>calls sys_alloc</td>
<td>1.4 MC</td>
</tr>
<tr>
<td>mbcs carrier_size</td>
<td>2.4 GB</td>
</tr>
<tr>
<td>sbcs carrier_size</td>
<td>11 GB</td>
</tr>
<tr>
<td>avg sbc block size</td>
<td>1.68 MB</td>
</tr>
</tbody>
</table>
Case 1: Large binaries

- +MBsbct 2147483648
  - Put binaries that are > 2 MB to mbcs
- +MBImbcs 20480 +MBsmbcs 1024
  - Increase average mbc size to fit the new larger blocks that will be put there
Case 2: Fragmentation

- Symptoms
  - erlang:memory(total) = about 7GB
  - top showed process at about 15 GB
  - Crash dump was written to: erl_crash.dump. ets_alloc: Cannot allocate XYZ bytes of memory. Abnormal termination
## Case 2: Fragmentation

<table>
<thead>
<tr>
<th></th>
<th>Current</th>
<th>Max</th>
</tr>
</thead>
<tbody>
<tr>
<td>blocks</td>
<td>2161022</td>
<td>4346598</td>
</tr>
<tr>
<td>blocks_size</td>
<td>1647 MB</td>
<td>6823 MB</td>
</tr>
<tr>
<td>carriers</td>
<td>934</td>
<td>936</td>
</tr>
<tr>
<td>carriers_size</td>
<td>7262 MB</td>
<td>7271 MB</td>
</tr>
<tr>
<td>avg block sz</td>
<td>799 Bytes/Block</td>
<td>1645 Bytes/Block</td>
</tr>
<tr>
<td>avg carrier sz</td>
<td>7.8 MB/Carrier</td>
<td>7.7 MB/Carrier</td>
</tr>
<tr>
<td>block sz / carrier sz</td>
<td>22,7%</td>
<td>93,8%</td>
</tr>
</tbody>
</table>
Case 2: Fragmented binaries

- +MBas aobf
  - Strive to allocate binaries in address order when there are ties
- +MBImbcs 512
  - Decreasing largest mbc size will make more carriers and hopefully be able to free them
New features

- Migration of carriers of same type
  - Added in R16B01, default in 17.0
  - Requires carrier oriented allocation strategy
- Super carrier
  - Added in R16B03
Questions?