Compressing Dynamic Data Structures

Haifeng He (hehf@cs.arizona.edu)

Problem

- Embedded system software is becoming increasingly complex.
- Physical memory is limited.
- Reduce memory consumption by compressing dynamic data structures

Program memory footprint

<table>
<thead>
<tr>
<th>static</th>
<th>dynamic</th>
</tr>
</thead>
<tbody>
<tr>
<td>Code</td>
<td>Stack</td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Dynamic data (heap)

Global data

Opportunity: Most scalar variables do not require a machine word to store values.**

Data structures compression

Value characteristics

- Narrow width: [0~0x100] 9 bits
- Common prefix: [0x8c0000~0x8c0fff] 12 bits (offset) + prefix
- Small set: {1,5,7,0x7ffffff} 2 bits (index) + Value table

Compression and Decompression

Primitives: given a compression scheme $S$
- $test\_compress(S, v)$, if $v$ can be compressed with $S$
- $compress(S, v)$, does the actual compression
- $decompress(S, v)$, decompresses $v$ with $S$

What if an incompressible value is encountered?
Expand compressed data object with additional memory space.

Source code transformation

Original | Transformed
struct $S$ {int d; char *p;}; | struct $S'$ {char cdata[size];};
$s\rightarrow d = v$; | $compress\_store(s\rightarrow cdata, fid, v)$
v = $s\rightarrow d$; | $v = compress\_load(s\rightarrow cdata, fid)$

$sid$ is a unique number assigned for each compressed structure field.

Example

Profiling

Compressing

Experimental results

Slab allocators in Linux kernel 2.6.19

<table>
<thead>
<tr>
<th>Struct Name</th>
<th>OrigSize</th>
<th>CompSize</th>
<th>Reduce</th>
</tr>
</thead>
<tbody>
<tr>
<td>ext2_inode_info</td>
<td>372</td>
<td>276</td>
<td>25.8%</td>
</tr>
<tr>
<td>inode</td>
<td>264</td>
<td>212</td>
<td>19.7%</td>
</tr>
<tr>
<td>dentry</td>
<td>124</td>
<td>80</td>
<td>35.5%</td>
</tr>
</tbody>
</table>

Why these structures? These structures consumes about 80% of the total size of memory spaces in the slab allocators.

Dynamic impact

Reduces the total memory consumption of the slab allocators by 16% when running MediaBench suite with minimal increases in execution time (1.6%).

Related work

- Zhang and Gupta: Data compression transformations for dynamically allocated data structures – hardware assisted approach.
- Cooprider and Regehr: Offline compression for on-chip RAM – compressing static allocated objects.

* This work was supported in part by NSF Grants CNS-0410918 and CNS-0615347.

** Brooks and Martonosi showed that over half of the integer operations in SPECint95 can be represented with 16 bits or less.