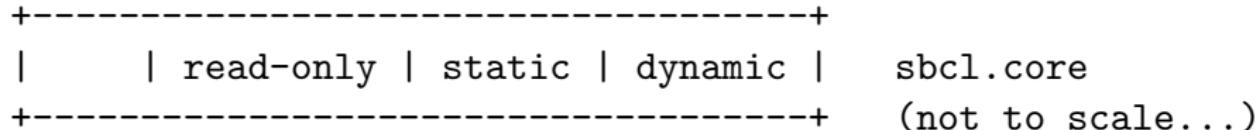


Incremental allocation for dynamic space

David Lichteblau

December 15, 2009

SBCL memory allocation



GC keeps a table of dynamic space pages:

```
page_index_t page_table_pages;  
struct page *page_table;
```

Dynamic space mmap()ed in one step, page table allocated in one step.

OS-level issues

```
(iter (repeat 20000) (cons-a-megabyte))
```

- ▶ If swapping possible, system unusable due to trashing.
- ▶ Without swapping, Linux VM starts killing things (when overcommit allowed).
- ▶ I want to run with overcommit disabled.
- ▶ I don't want to calculate a --dynamic-space-size in advance.

Lisp-level issues

```
(handler-bind
  ((out-of-memory (lambda (*) (cons-a-kilobyte))))
  ...)
```

- ▶ Hard to design out-of-memory handling that is provably safe in Lisp (the error handling will cons). Compare to Java where a cached exception can unwind the stack, without risk of handler-bind making things worse.
(Java throw is cl:throw, not cl:signal)
- ▶ With SBCL's copying GC, can't cope with running out of memory within the GC anyway.

Solutions

1. Start with a small dynamic space, grow it dynamically.
2. Set an (arbitrary but) "soft" memory limit, which can be increased in the debugger.

The incremental allocation patch

```
page_index_t
gc_find_freeish_pages(page_index_t *restart_page_ptr,
                      long nbytes,
                      int unboxed)
{
    ...
    if (first_page >= page_table_pages)
+ifdef LISP_FEATURE_INCREMENTAL_ALLOCATION
+    return gc_map_new_pages(restart_page_ptr, nbytes);
+else
+    gc_heap_exhausted_error_or_lose(0, nbytes);
+endif

    ...
    return last_page;
}
```

```

static page_index_t
gc_map_new_pages(page_index_t *restart_page_ptr, long nbytes)
{
    ...

/* first check the soft allocation limit */
if (soft_pages_limit && ...)
    ... signal SOFT-HEAP-EXHAUSTED-ERROR ...

actual_pos = gc_validate_monotonically(target_pos, nbytes);

/*
 *          target_pos      actual_pos
 *          |              |
 *          v              v
 *
 *          +-----+.....|.....|
 *          |11111022220333333333111022220|.....|
 *          +-----+.....|.....|
 *          old page table           ^      ^
 *                               |      |
 *                               +-----+-----+
 *                               |33333333|00000000|
 *                               +-----+-----+
 *                               hole     new pages
 */
realloc_page_table(new_page_table_pages, nbytes);

for (; i < new_page_table_pages; i++) {
    page_table[i].allocated = FREE_PAGE_FLAG;
    ...
}
...
}

```

What about the holes?

Ideally, there will be $\text{target_pos} = \text{actual_pos}$, so that dynamic space is contiguous.

In case there are holes, GC ignores those pages.

Core file saving uses `relocate_all` to remove the holes.

Demonstration