

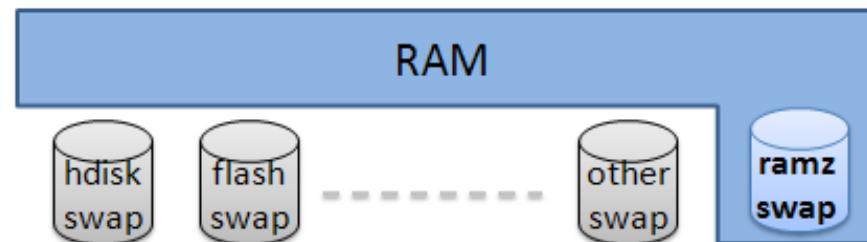
# xvMalloc (A TLSF variant for compcache)

Nitin Gupta



# compcache project

- This project creates RAM based block device (named ramzswap) which acts as swap disk. Pages swapped to this disk are compressed and stored in memory itself.



- Project home: <http://code.google.com/p/compcache>
- Mailing list: [linux-mm-cc@laptop.org](mailto:linux-mm-cc@laptop.org)



# Memory Allocator: xvMalloc

- xvMalloc: memory allocator designed specifically for compcache project
  - $O(1)$  Alloc/Free - Except for cases where we have to go to system page allocator to get additional memory.
  - Very low fragmentation - In all tests, xvMalloc memory usage within 12% of “Ideal” allocator.
  - Low metadata overhead
- Supports multiple pools: each pool can grow/shrink
- Based on TLSF (Two Level Segregate Fit): uses two level bitmap scheme similar to TLSF allocator:
  - [http://rtportal.upv.es/rtmalloc/files/MRBC\\_2008.pdf](http://rtportal.upv.es/rtmalloc/files/MRBC_2008.pdf)



## xvMalloc contd...

- Why new allocator
  - Allocators assume memory being managed is always mapped to VA space.
  - 32-bit systems have very small kernel VA region (Linux 32-bit: ~256MB only)
    - This limits amount of memory compcache can allocate
- Need allocation only in small range: say, [32b,  $\frac{3}{4}$ \*PAGE\_SIZE]
  - xvMalloc: equally spaced (just 8 bytes!) free lists for entire size range
  - TLSF: size difference between free lists increases rapidly as size increases (for objects > 1k, free lists are separated by 128 bytes!)
- Per object metadata (64-bit system):
  - TLSF: 16b
  - xvMalloc: 4b
- Minor: xvMalloc stores **exact** object size in object header - saving another 2 bytes/object. Provides xvGetObjectSize(obj). Decompressor needs this size.



# xvMalloc contd...

- Object Layout

- Used object header (4 bytes)



## Fields:

Size: object size (as given by user – not rounded-off size)

Prev: offset of previous used/free block (relative to beginning of page)

## Flags:

- F1: Block is used/free

- F2: Previous block is used/free

Pad: not required if ALIGN is 4b



# xvMalloc contd...

- Object Layout

- Free object header (additional 12 bytes)



- Min object size: 32b
- Objects < 32b rounded off to 32b, others rounded to ALIGN (usually 4b)
- Each object located with <pageNum, offset> pair



## xvMalloc contd...

- Allocator needs implementation of function that provides dereferencable ptr given this <pageNum, offset> pair:
  - `void *GetPtr(u32 pageNum, u16 offset)`
  - `void PutPtr(void *ptr)`
- Merges adjacent free blocks
  - Maps/Unmaps pages (using GetPtr/PutPtr) as required when checking adjacent blocks
  - On Linux as simple as:

```
static void *GetPtr(u32 pageNum, u16 offset)
{
    unsigned char *page_ptr;
    page_ptr = kmap_atomic(pfn_to_page(pageNum), KM_USER0);
    return (page_ptr + offset);
}
```

```
static void PutPtr(void *ptr)
{
    kunmap_atomic(ptr, KM_USER0);
}
```



## xvMalloc contd...

- More info:

<http://code.google.com/p/compcache/wiki/xvMalloc>

- Code:

<http://code.google.com/p/compcache/source/browse/#svn/trunk/sub-projects/allocators/xvmalloc-kmod>



END