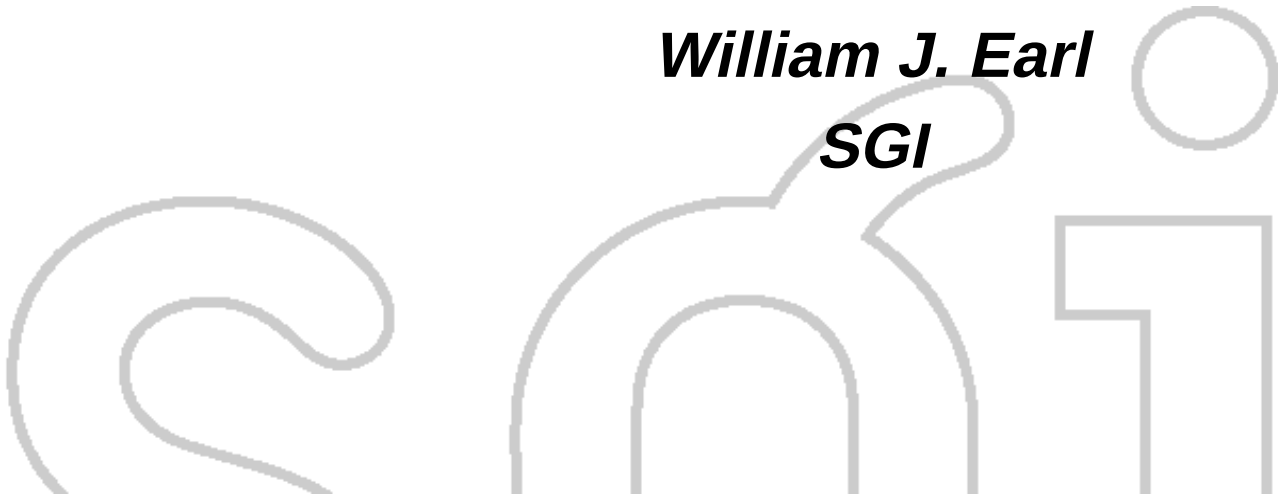
Buffer Management for XFS in Linux

William J. Earl

SGI



XFS Requirements for a Buffer Cache

- Delayed allocation of disk space for cached writes
  - supports high write performance
- Delayed allocation main memory reservation
  - avoids memory deadlocks when later allocating pages
- Single buffer object for a large logical buffer
  - supports very high data rates (7 GB/second for a single file)
  - buffers of 1 MB or more needed in many cases
- Ability to pin storage for a buffer in memory
  - supports write−ahead−log protocol for metadata updates
- Full integration of buffer cache with page cache
  - all buffer data pages are entered in the page cache
- Integration with Direct I/O
- Parallel I/O initiation

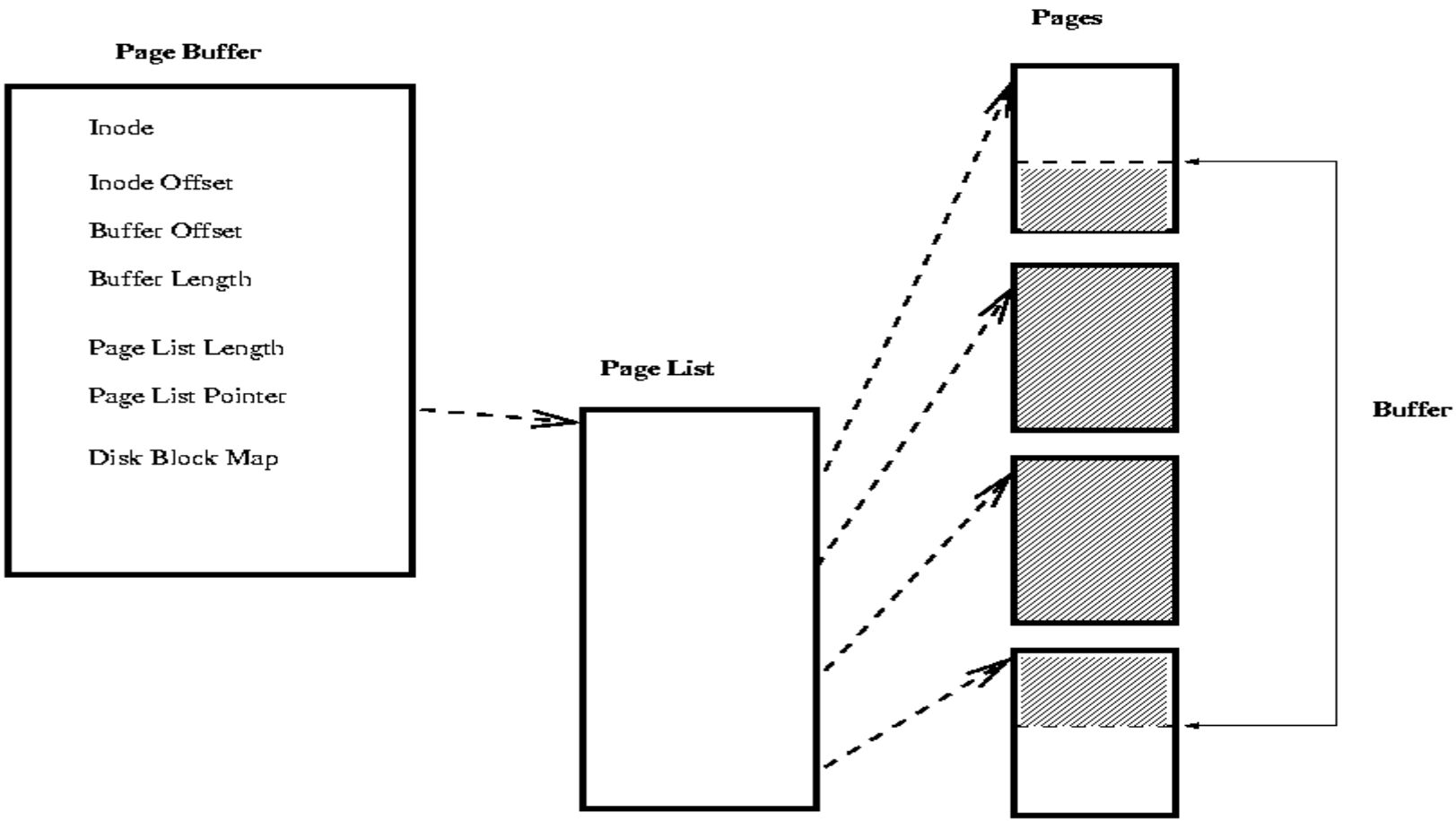
Delayed Allocation Main Memory Reservation

- Actual allocation may require buffer space
- Freeing buffer space may require allocation of space on disk for delayed writes
- A reservation system must assure a minimum amount of buffer space to avoid deadlock during allocation
- Delayed allocations are counting against a maximum amount of main memory (typically 80% of available main memory), until actual allocation of disk space is completed
- Page flushing should flush enough delayed allocation pages to keep some memory available for reservation, even if there is free memory
- Reservation system allows threads to wait for space

Single Buffer Object for a Large Logical Buffer

- XFS supports very high data rates
  - 7 GB/second measured from a single file
- Large buffers used for high data rates
  - a buffer commonly represents a disk extent
- Using a buffer_head per block for a large (4 MB or larger) buffer requires a lot of space (and cache misses) and CPU overhead
  - *7 GB/second with 4 KB blocks is 1.75 million buffer_head create, use and destroy operations per second or about one every 570 ns*
- Aggregate (multiple−block) buffer object just keeps physical page number or mem_map_t pointers for much reduced space and time overhead
  - *prototype interface in SGI Raw I/O patch*

page_buf_t and components



Ability to Pin Memory for a Buffer

- **XFS uses a write−ahead log protocol for meta−data writes**
  - ***write log entry before updating meta−data***
  - ***on recovery just apply after images from the log (in case some of the meta−data writes were not completed)***
- **“Pin a page” means “keep page flushing from writing out a page”**
  - ***such pages must count against the memory reservation (just as do delayed allocation pages)***
- **XFS pins a metadata page before updating it, logs the updates, and then unpins the page when the relevant log entries have been written to disk**

Partial Aggregate Buffers

- Pages within an extent may be deleted from memory so a buffer for an extent may not find all pages present
- If the buffer is needed for writing, empty (invalid) pages may be used to fill the holes
- If the buffer is needed for reading just part of the extent, missing pages need not be read if all pages to be read are present
- When missing pages are required, cache module will read in the missing pages

Efficient Assembly of Buffers

- **Overhead for finding all valid pages within an extent must be low**
- **Pages for a given inode should be available cheaply in sorted order**
  - ***could change page cache to use an AVL tree off the inode to lookup pages derived from the inode rather than the hash table***
- **Large pages are required**
  - ***fewer pages to manage***
  - ***page migration required for reliable reassembly of large pages***
  - ***pages which are not migratable must be clustered to avoid permanent fragmentation of large pages***
  - ***buffers and other uses must not hold long−term locks on pages which prevent migration***

Metadata Buffers

- Place pages in cache associated with file system and log device inodes
  - regular file pages are associated with file inodes
- Common I/O path for metadata and regular data except that the metadata disk map Is one−to−one with the logical offset

Direct I/O

- **Small files which a frequently referenced are best kept in cache**
- **Huge files such as image and streaming media files and scientific data files are best not cached since blocks will always be replaced before being reused**
- **Direct I/O is raw I/O for files**
  - ***I/O directly to or from user buffers***
  - ***no data copying***
  - ***no invalidation of cached blocks***
- **Buffer cache must cooperate with direct I/O so that any pages which are cached and are modified are read from memory and writes update cached pages**

Direct I/O VM Issues

- **Direct I/O and Raw I/O avoid copying by addressing user pages directly**
  - ***application promises not to change the buffer during a write***
- **Physical Pages are locked in place for the duration of the I/O**
  - ***page reference count increased during the I/O***
  - ***user mapping of page may be release without causing errors***
- **SGI Raw I/O patch has an initial implementation (to be improved for the XFS port)**
- **Direct I/O would help with Samba and Web serving performance if it were supported by the network interfaces**
  - ***writes would block until packets were transmitted***
- **Compare to the IOLite model**

Parallel I/O Initiation

- Buffer cache must not interfere with parallel initiation of multiple independent I/O requests
- Locks should cover a minimum scope and be held briefly
- Page and buffer lookups must avoid excessive lock contention

Mapping XFS Buffering onto Linux

- XFS buffer cache module on top of stand Linux page cache
- Linux 2.3 has moved to using the page cache for file data
  - *Linux 2.2 page cache can support the layered XFS buffer cache module*
- Separate buffer_head object required for each block
  - *optional extension to drivers to support aggregate (multiple−block) buffer object for both XFS and Raw I/O*
- Limit complexity of layered buffer cache (compared to IRIX)
  - *buffer objects are temporary*
  - *all persistent data stored in mem_map_t*
  - *minimize long−term locks on buffers*
- Avoid deadlock issues as when writing to a file from a buffer mmapped onto the same file