JFS Tuning and Performance

Mark Ray
Hewlett Packard
mark_ray@hp.com

HP WORLD 2002
Conference & Expo

JFS Tuning and Performance

- Understanding JFS
- Understanding your application
- Creating your file system
- Mount options
- File system tuneables
- System wide tuneables
- JFS ioctl() options

Understanding JFS

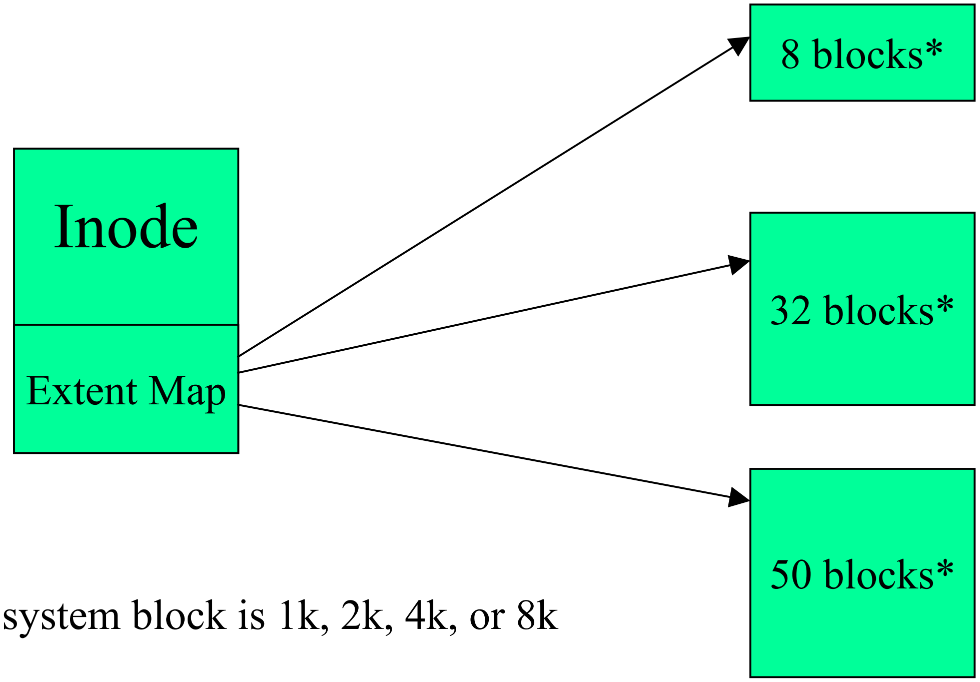
- JFS software versions vs. disk layout versions
- Variable sized extent based file system
- Extent allocation
- Transaction journaling
- Fragmentation
- Defragmenting your file systems

JFS Software Versions vs. Disk Layout Versions

OS	SW version	Disk layout version
10.01	JFS 2.0	2*
10.10	JFS 2.3	2*
10.20	JFS 3.0	2,3*
11.0	JFS 3.1	2,3*
	JFS 3.3	2,3*,4
11.11	JFS 3.3	2,3,4*

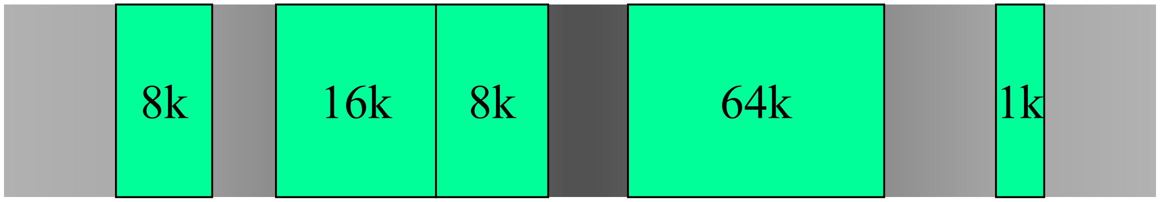
* Denotes default disk lay out version

Variable Sized Extent Based File System



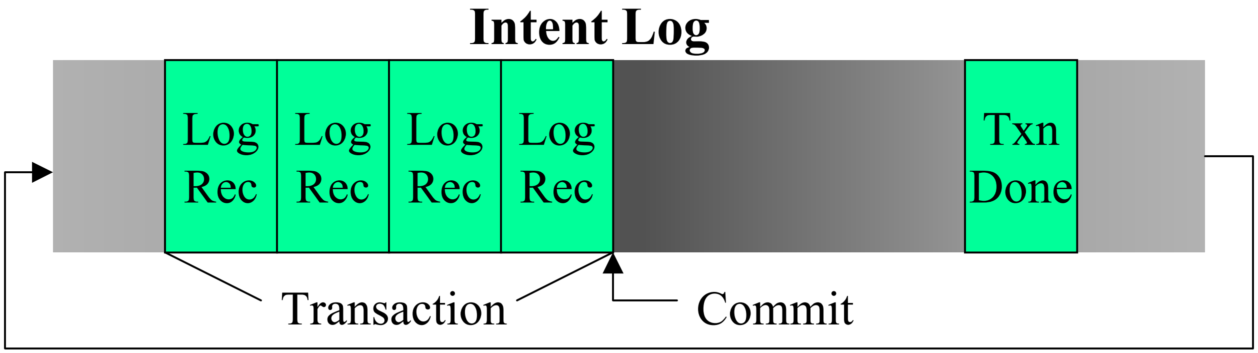
* Each file system block is 1k, 2k, 4k, or 8k

Extent Allocation



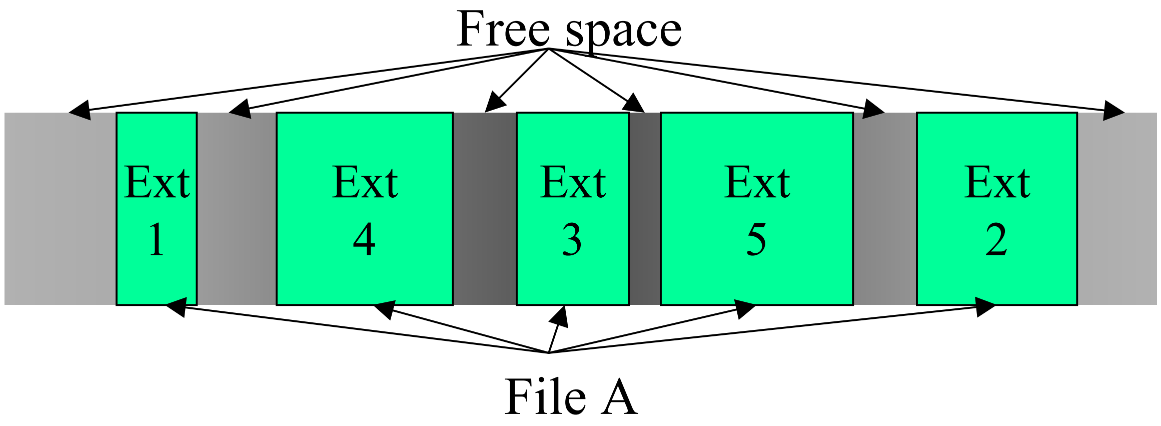
- Amount of writes is unknown until the file is closed
- Initial extent size is determined by size of the 1st write (8k minimum)
- Extend current extent when full if possible
- Extents get progressively larger
- Last extent is trimmed on last close

Transaction Journaling



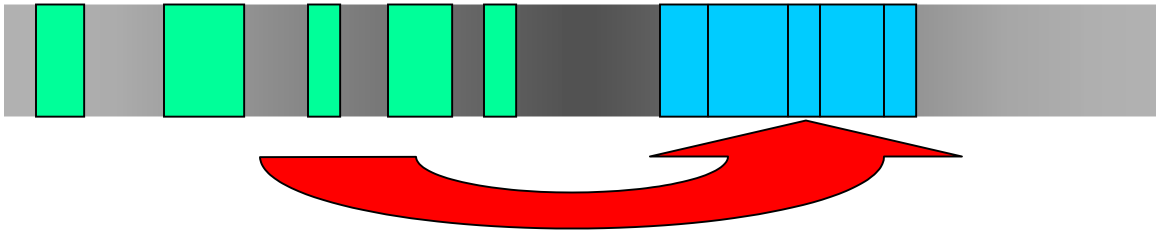
- Log structural changes to the file system
- Circular log called *Intent Log*
- Provides fast file system recovery after a system crash
- Small synchronous writes may also be logged

Fragmentation



- As files are created and removed, free space becomes fragmented
- When files are closed, the last extent is trimmed
- As files are extended, free space may come from non-adjacent areas

Defragment Your File Systems



- Use fsadm –e to defragment on a regular basis, fsadm –E to report on fragmentation
- Performing 1 8k I/O will be much faster than performing 8 1k I/Os
- File systems with small block sizes are more susceptible to fragmentation

Understanding Your Application

- How are your files accessed?
  - Reads vs. Writes
  - Sequential vs. Random
  - Size of I/O, files, directories
  - Volume and file system layout
  - Parallel vs. Single access
  - Data integrity vs. Performance

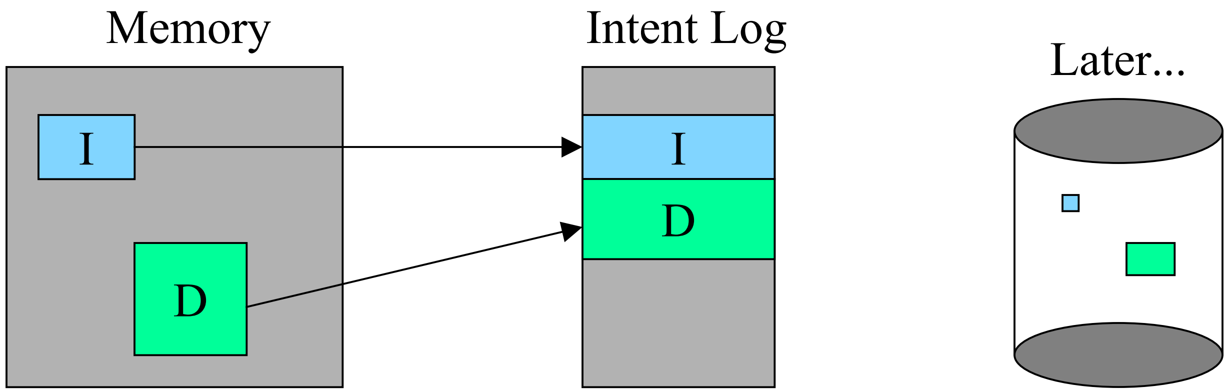
Creating Your File System (newfs, mkfs)

- Block size (bsize)
  - Use large block size for performance
  - Use small block size to reduce wasted space
- Intent Log size (logsize)
  - Increase Intent Log size for heavy log activity
- Disk layout version (version)
  - Later disk layout versions contain improvements that can affect performance

Mount Options

- Clearing data blocks during extent allocation (*blkclear*)
- Logging small synchronous writes in the intent log (*datainlog, nodatainlog*)
- Buffer cache options (*mincache*)
- Converting O_SYNC operations (*convosync*)
- Intent Log options

Mount Options
datainlog, nodatainlog



- Logs small synchronous writes in the Intent Log (datainlog)
- *Datainlog* simulates synchronous writes
- Available with HP OnLineJFS product

Mount Options mincache

- Buffer cache options (*mincache*)
  - Mincache=closesync
  - Mincache=dsync*
  - Mincache=direct*/unbuffered*
  - Mincache=tmpcache*

*Available only with HP OnLineJFS product

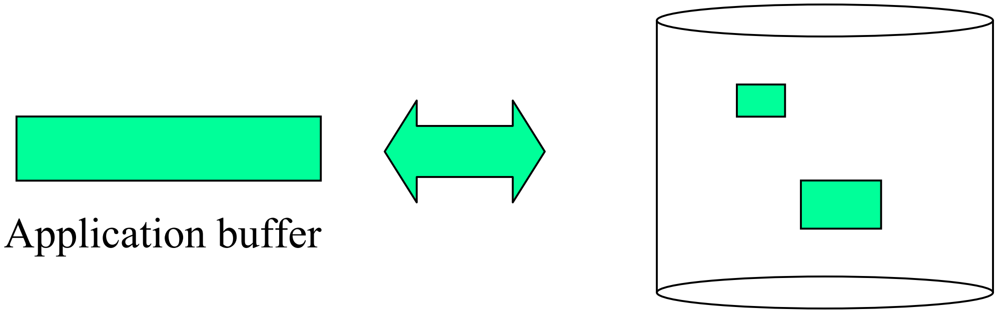
Mount Options
convosync

- Converting O_SYNC operations (*convosync*)
  - convosync=closesync
  - convosync=direct/unbuffered
  - convosync=dsync
  - convosync=delay
- Available only with HP OnLineJFS product

Mount Options
Intent Log

- Log level
  - nolog - with JFS 3.3, same as tmplog
  - tmplog - most transactions delayed
  - delaylog - some transactions delayed
  - log (default) - transactions must be flushed before operation can be performed

Direct I/O

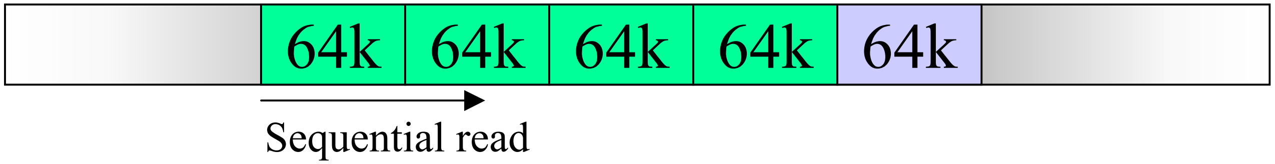


- Direct I/O bypasses buffer cache
- Only available with HP OnLineJFS product
- Good for large I/O and data accessed once
- Data integrity
- Enabled with mount options or VX_SETCACHE ioctl or through Discovered Direct I/O
- All direct I/O is synchronous

Dynamic File System Tunables

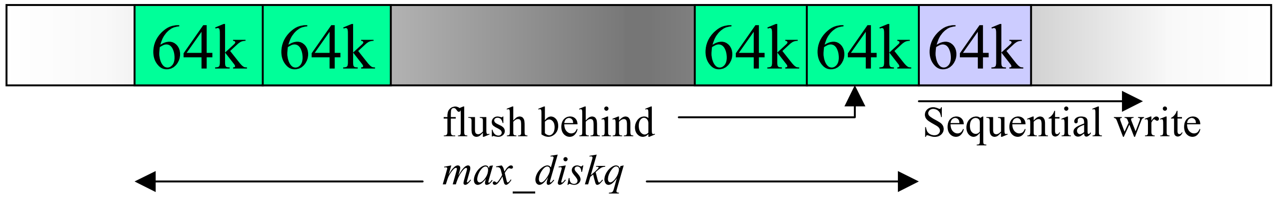
- Read ahead (*read_pref_io* and *read_nstream*)
- Flush behind (*write_pref_io* and *write_nstream*)
- I/O throttling (*max_diskq*)
- Buffer sizes (*max_buf_data_size*)
- Discovered Direct I/O (*discovered_direct_iosz*)
- Extent allocation policies (*initial_extent_size* and *max_seqio_extent_size*)

Read Ahead



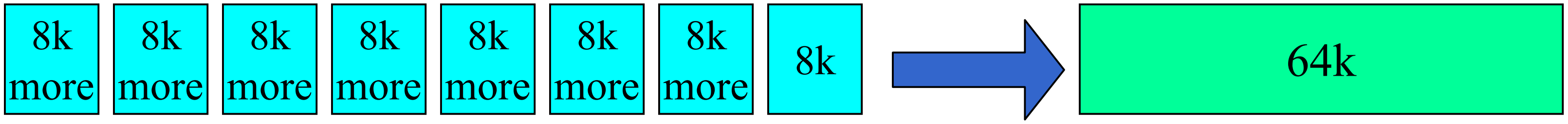
- JFS detects sequential pattern, prefetches data into buffer cache
- Read ahead size is calculated using *read_pref_io* and *read_nstream*
- Maintains 4 ranges of read ahead size
- Sequential read ahead affected by other processes or threads

Flush Behind and I/O Throttling



- Flush behind amount is calculated using *write_pref_io * write_nstream*
- Amount of data being flushed cannot exceed *max_diskq* (default 1MB)
- Processes block until amount of outstanding flushes drops below *max_diskq*

Buffer Sizes



- JFS uses a default maximum buffer size of 8k
- Maximum buffer size can be changed to 65536 by tuning *max_buf_data_size*
- For large reads and read ahead, JFS “chains” buffers together
- Change *max_buf_data_size* to 64k for large reads and writes

Discovered Direct I/O

- If read and write size is greater than or equal to *discovered_direct_iosz, then* direct I/O will be used
- Only available with HP OnLineJFS product
- Has same advantages and disadvantages as direct I/O

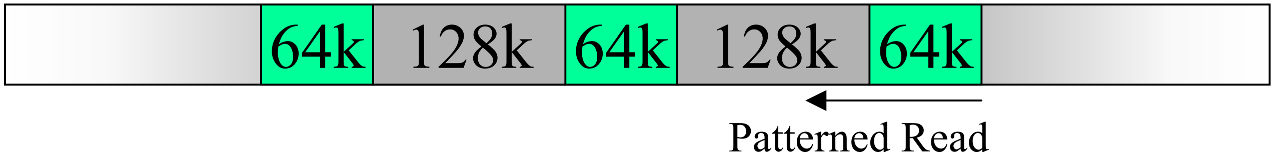
Extent Allocation Policies

- First extent is usually the smallest
- *initial_extent_size* can be used to change the size of the initial extent (default 8 blocks)
- *max_seqio_extent_size* can be used to change maximum size of an extent (default 2048 blocks)

System Wide Tunables

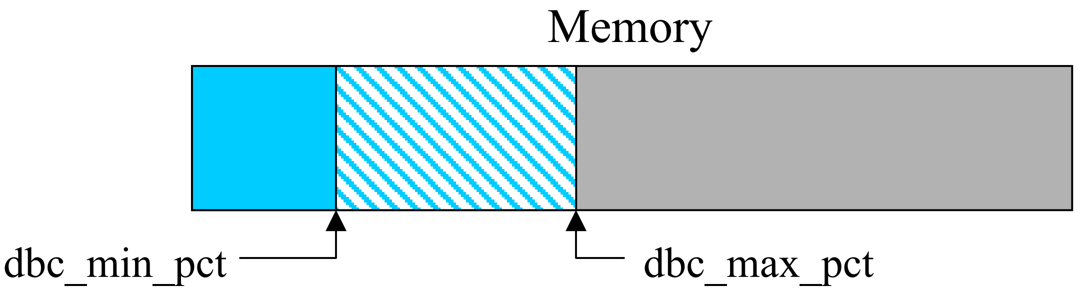
- Fancy Read Ahead (*vx_fancyra_enable*)
- Buffer cache (*nbuf, bufpages, dbc_min_pct, dbc_max_pct*)
- JFS Inode Cache (*vx_ninode*)
- Directory Name Lookup Cache (*ncsize, vx_ncsize*)

Fancy Read Ahead



- Detects non-sequential patterns
- Capable of handling multiple patterns from one or more threads
- Enabled using system wide tuneable *vx_fancyra_enable*

Buffer Cache



- Buffered I/O can be done asynchronously
- *Dbc_max_pct* specifies maximum percent of memory that can be used by dynamic buffer cache
- Dynamic buffer cache grows quickly, shrinks slowly
- Use *nbuf* / *bufpages* to specify static buffer cache
- Buffers must be flushed or invalidated when file system is synced or unmounted

JFS Inode Cache

- Memory cache of most recently accessed inodes
- Size of cache is dynamic
- Default maximum size based on amount of memory
- Maximum size can be tuned using vx_ninode
- Must have 1 inode cache entry in memory for every opened file

Directory Name Lookup Cache

- DNLC is a cache of most recently used directory and file names
- DNLC is searched first before searching actual directories
- Caches directory names of 39 characters or less
- DNLC sized by *ncsize* and *vx_ncsize*

Keep Your Directories Small

- Keep directories small (<10,000 entries)
- Directories are typically fragmented files
- Simultaneous searches can lead to directory contention
- Avoid ll(1) or stat() of files in large directories
- Large directories can be defragmented*

* Version 4 disk layout

JFS ioctl() Options Cache Advisories

- VX_SETCACHE ioctl()
  - VX_RANDOM - Treat I/O as random
  - VX_SEQ - Perform maximum read ahead
  - VX_DIRECT - Bypass buffer cache
  - VX_NOREUSE - Invalidate buffer after use
  - VX_DSYNC - Data synchronous I/O
  - VX_UNBUFFERED - Bypass buffer cache
- Available with HP OnLineJFS product

JFS ioctl() Options Allocation Policies

- VX_SETEXT ioctl() sets a fixed extent size and optionally reserves space for the file
  - VX_NOEXTEND - do not extend past current reservation
  - VX_TRIM - trim file after last close
  - VX_CONTIGUOUS - reserved space must be contiguous
  - VX_ALIGN - extents must be aligned on an extent sized boundary
  - VX_NORESERVE - reservation will not survive crash
  - VX_CHGSIZE - reserve space and update inode
- Available with HP OnLineJFS product

Patches

- Several patches have been created for JFS 3.3 on 11.0 and 11.11 to address various performance related problems
- PHKL_27212 (11.0); PHKL_27121 (11.11)
  - Sequential I/O if read size < 64k
  - Multiple readers with Fancy Read Ahead
  - Sequential I/O with Fancy Read Ahead
  - Random reads
  - Backward and forward

Summary

newfs mkfs	Mount options	File system tuneables	System wide tuneables	Per-file attributes
bsize	mincache	read_pref_io	vx_fancyra_enable	VX_SETCACHE
logsize	convosync	read_nstream	vx_ninode	VX_SETEXT
version	datainlog	write_pref_io	vx_ncsize	
	nodatainlog	write_nstream	nc_size	
	log	max_diskq	nbuf	
	delaylog	max_buf_data_size	bufpages	
	tmplog	discovered_direct_iosz	dbc_min_pct	
	nolog	initial_extent size	dbc_max_pct	
		max_seqio_extent_size		