HP WORLD 2004
Solutions and Technology Conference & Expo

JFS Tuning and Performance

Mark Ray
HP-UX Global Solutions Engineering
Hewlett-Packard

JFS Tuning and Performance

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

Understanding JFS

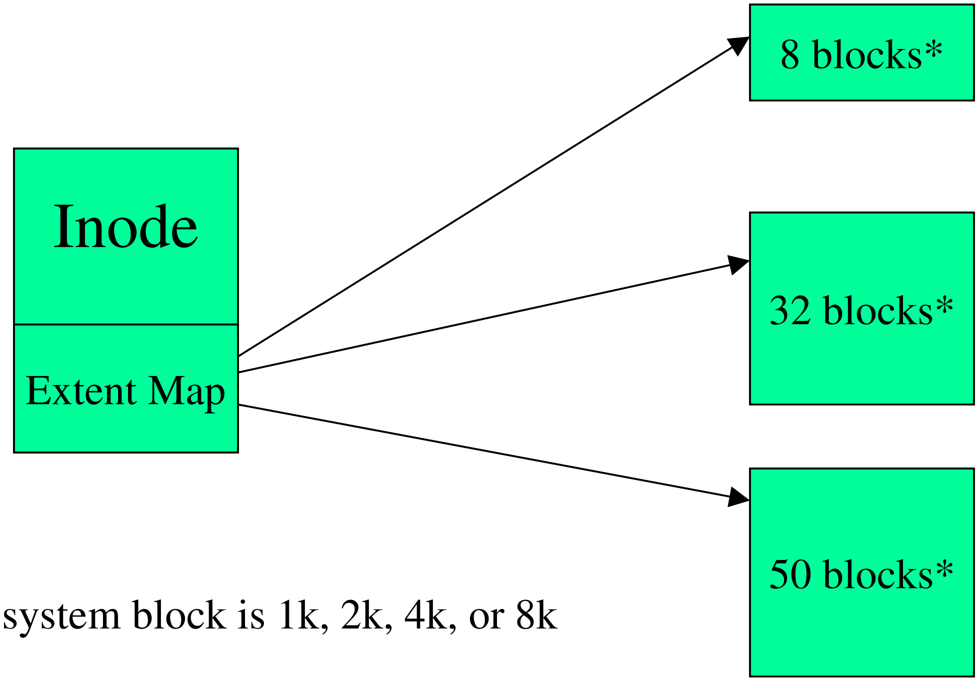
- JFS software versions vs. disk layout versions
- Variable sized extent based file system
- Extent allocation
- Fragmentation
- Defragmenting your file systems
- Transaction journaling

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 JFS 3.3	2,3* 2,3*,4
11.11	JFS 3.3 JFS 3.5	2,3,4* 4*
11.22	JFS 3.3	2,3,4*
11.23	JFS 3.5	4,5*

* Denotes default disk lay out version

Variable Sized Extent Based File System



Inode

Extent Map

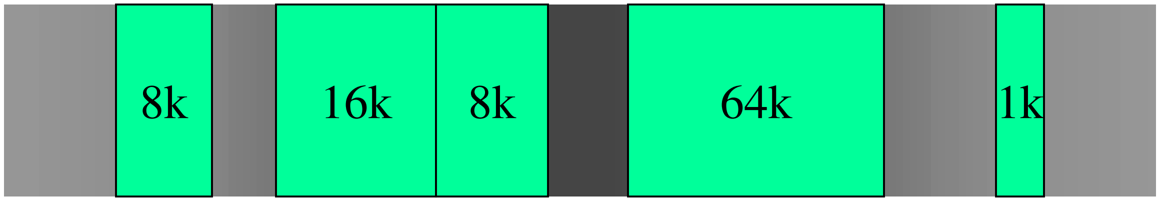
8 blocks*

32 blocks*

50 blocks*

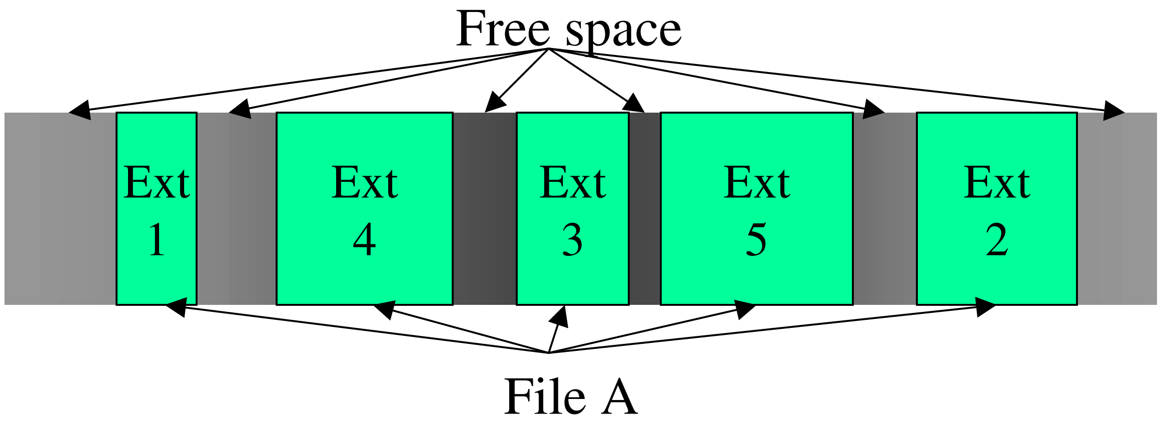
* Each file system block is 1k, 2k, 4k, or 8k

Extent Allocation



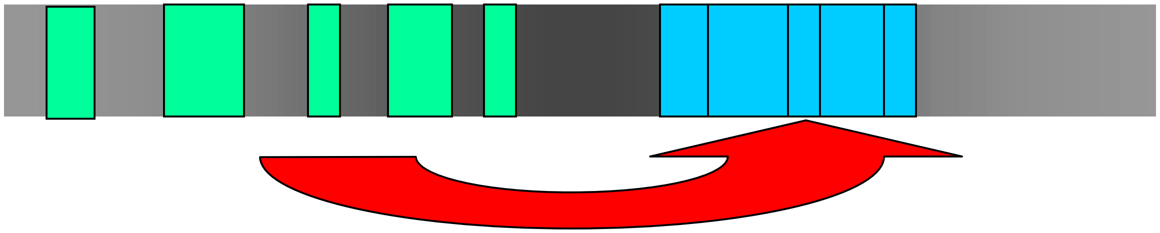
- Amount of writes is unknown until the file is closed
- Initial extent size is determined by size of the 1st write (8k minimum)
- Extend current extent when full if possible
- Extents get progressively larger
- Last extent is trimmed on last close

Fragmentation



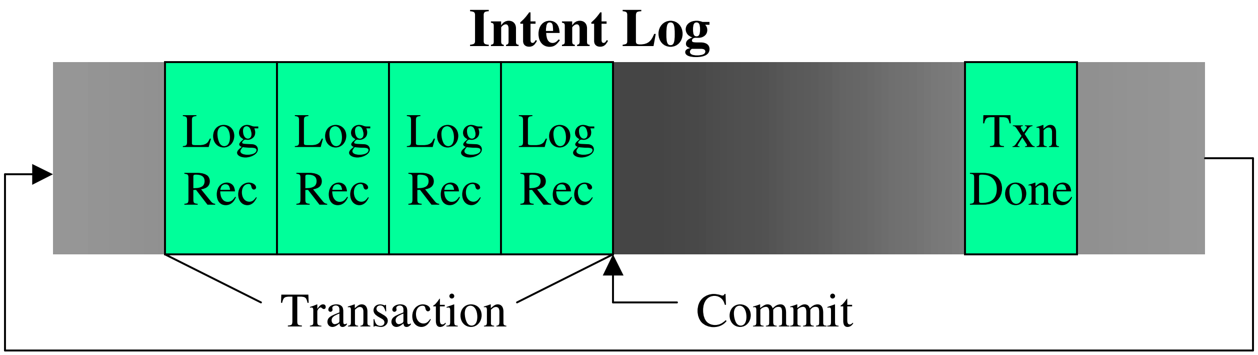
- As files are created and removed, free space becomes fragmented
- When files are closed, the last extent is trimmed
- As files are extended, free space may come from non-adjacent areas

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

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

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)

- 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
- Block size (bsize)
  - Use small block size to reduce wasted space
  - Larger block sizes use less space for metadata. Less impact from free space fragmentation.

Creating Your File System
Default block sizes

FS Size	JFS 3.1	JFS 3.3	JFS 3.5
< 8GB	1K	1K	1K
<16GB	1K	2K	1K
<32GB	1K	4K	1K
<=4TB	1K	8K	1K*
<=8TB	-	-	2K*
<=16 TB	-	-	4K*
<=32TB	-	-	8K*

*VxFS Disk Layout Version 5 (11.23) and HP OnlineJFS licensed is needed for file systems >2TB
Currently, HP currently supports file systems <= 12 TB

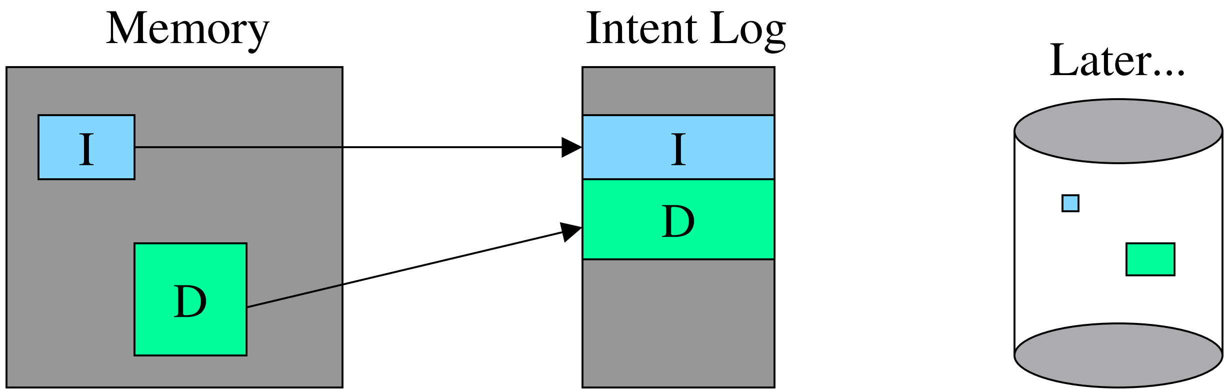
Mount Options

- Clearing data blocks during extent allocation (*blkclear*)
- Logging small synchronous writes in the intent log (*datainlog*)
- Buffer cache options (*mincache*)
- Converting O_SYNC operations (*convosync*)
- Intent Log options (*log, delaylog, tmplog, tranflush, logiosize*)
- Inode access times (*noatime*)

Mount Options - blkclear

- write() syscall
  - clear extent first by writing out zeros.
  - then write out data
- prevents stale data from showing up in a file after a system crash.
- Sacrifices performance for security
- writes to “holes” in a sparse file must always clear the extent first, then write the data.

Mount Options - datainlog



- 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

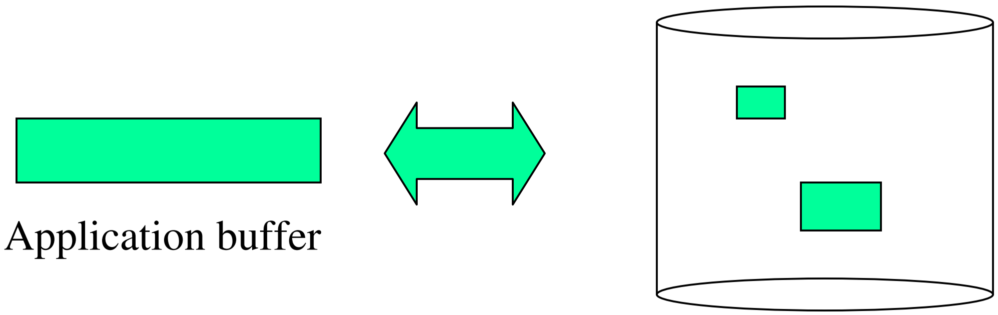
Mount Options - Intent Log

- 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
- logiosize – size of I/O buffered used by the Intent Logging.
- tranflush – flushes all metadata to disk before returning from a system call.

Mount Options - Intent Log

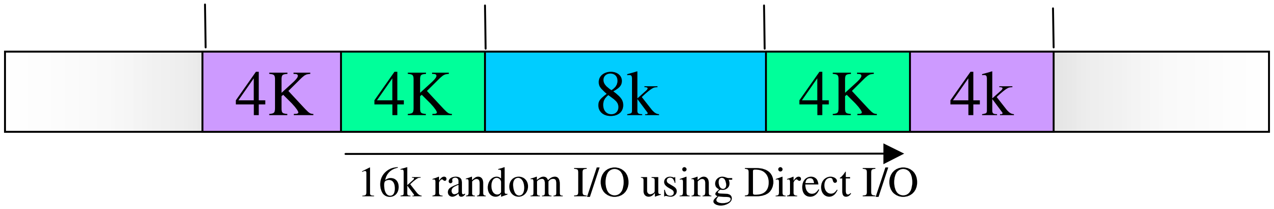
Operation	JFS 3.3 JFS 3.5 log	JFS 3.3 delaylog	JFS 3.5 delaylog	JFS 3.3 JFS 3.5 tmplog
Async Write	Delayed	Delayed	Delayed	Delayed
Sync Write	Flushed	Flushed	Flushed	Flushed
Read	n/a	n/a	n/a	n/a
Sync (fsync())	Flushed	Flushed	Flushed	Flushed
File Creation	Flushed	Delayed	Delayed	Delayed
File Removal	Flushed	Flushed	Delayed	Delayed
File Timestamp changes	Flushed	Delayed	Delayed	Delayed
Directory Creation	Flushed	Flushed	Delayed	Delayed
Directory Removal	Flushed	Flushed	Delayed	Delayed
Symbolic/Hard Link Creation	Flushed	Delayed	Delayed	Delayed
File/Dir Renaming	Flushed	Flushed	Flushed	Delayed

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

Direct I/O and unaligned data

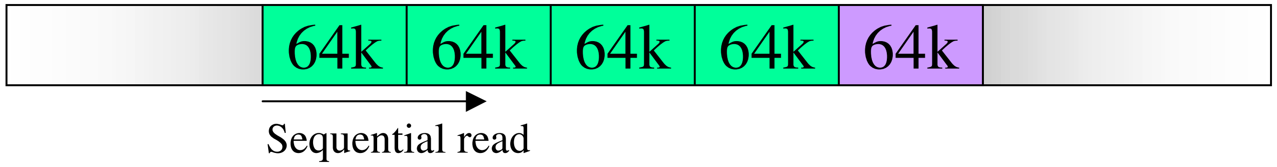


- Direct I/O must be aligned on a file system block boundary
- Unaligned portions of the I/O must be buffered.
- The buffered portions are transferred synchronously
- Unaligned Direct I/O results in added data transferred from and to disk
- Use smallest block size for greatest chance of doing aligned Direct I/O

Dynamic File System Tunables

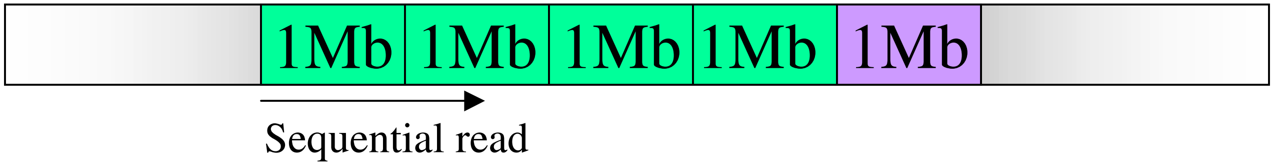
- Read Ahead (*read_pref_io* and *read_nstream*)
- Fancy Read Ahead (*read_ahead*)
- Flush Behind (*write_pref_io* and *write_nstream*)
- I/O throttling (*max_diskq and write_throttle*)
- Buffer sizes (*max_buf_data_size*)
- Discovered Direct I/O (*discovered_direct_iosz*)
- Extent allocation policies (*initial_extent_size* and *max_seqio_extent_size*)

Read Ahead



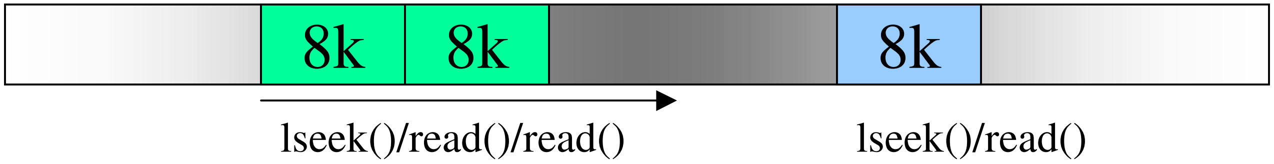
- JFS detects sequential pattern, prefetches data into buffer cache
- Read ahead size is calculated using *read_pref_io* and *read_nstream*
- Maintains 4 ranges of read ahead size
- Sequential read ahead affected by other processes or threads

Read Ahead with VxVM stripes



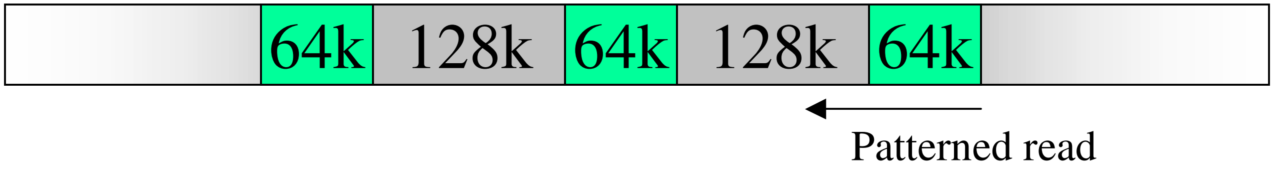
- For VxVM volumes
  - read_pref_io defaults to stripe size
  - read_nstream defaults to the number of stripes (or columns).
- Too much readahead can negatively affect performance.

False sequential I/O patterns and Read Ahead



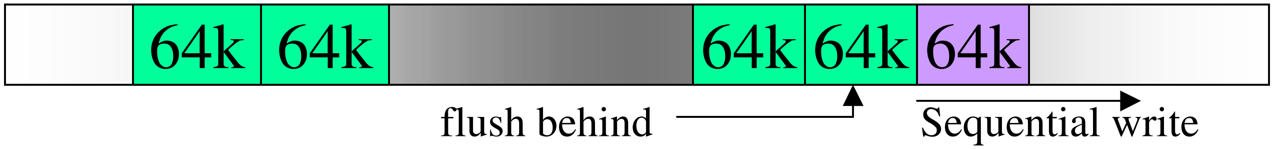
- Application does mostly random I/O
- Occasionally, it does 2 sequential I/Os
- Read ahead is triggered, prefetching unwanted data

Fancy Read Ahead



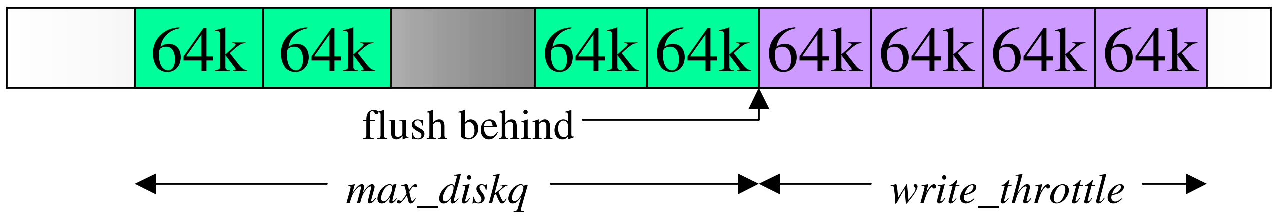
- Detects non-sequential patterns
- Capable of handling multiple patterns from one or more threads
- Enabled using system wide tunable *vx_fancyra_enable (JFS 3.3),* or by tuning *the read_ahead* tunable using vxtunefs *(JFS 3.5)*

Flush Behind



- Flush behind amount is calculated using *write_pref_io * write_nstream*
- When the number of dirty buffers for a file exceeds the flush behind amount, JFS will start to issue asynchronous writes to flush the dirty buffers.

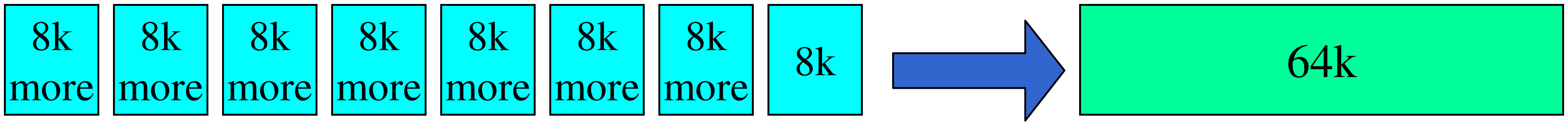
I/O Throttling



- Amount of data being flushed per file cannot exceed *max_diskq* (default 1MB)
- Sync processes block until amount of outstanding flushes drops below *max_diskq*
- Amount of outstanding dirty buffers per file cannot exceed *write_throttle* (default 0)
- Write processes block until amount of dirty buffers drops below *write_throttle*

HP WORLD 2004 Solutions and Technology Conference & Expo

Buffer Sizes


8k more 8k more 8k more 8k more 8k more 8k more 8k more 8k

64k

- 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
- IA-64 systems do not perform merging of JFS buffers.

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

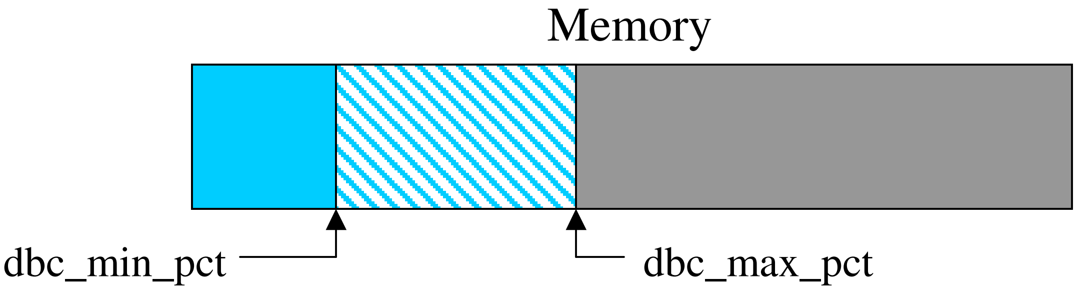
- First extent is usually the smallest
- *initial_extent_size* can be used to change the size of the initial extent (default 8 blocks)
- *max_seqio_extent_size* can be used to change maximum size of an extent (default 2048 blocks)

System Wide Tunables

- Buffer Cache (*nbuf, bufpages, dbc_min_pct, dbc_max_pct*)
- JFS Metadata Buffer Cache (*vx_bc_bufhwm*)
- JFS Inode Cache (*vx_ninode*)
- Directory Name Lookup Cache (*ncsize, vx_ncsize*)

HP WORLD 2004
Solutions and Technology Conference & Expo

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

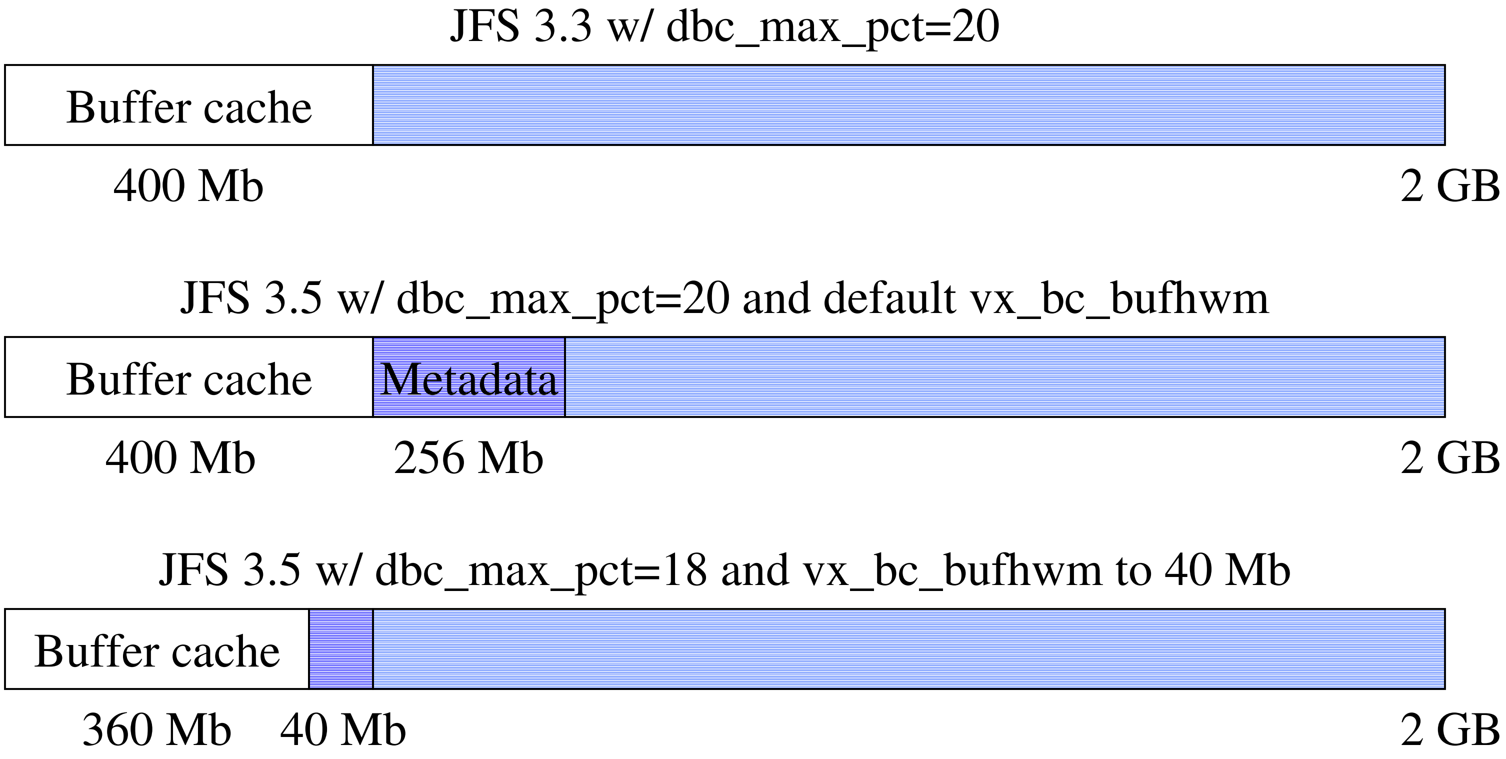
JFS Metadata Buffer Cache

- Metadata is structural data, such as superblocks, inodes, directory blocks, bitmaps, etc
- Prior to JFS 3.5, JFS metadata and user data were cached into the HP-UX buffer cache
- JFS 3.5 introduced a new cache in memory for JFS metadata only
- Performance advantages such as shared buffer reads
- Disadvantage is increased memory utilization
- Tuned using vx_bc_bufhwm

JFS Metadata Buffer Cache Default sizes

Memory Size (Mb)	JFS Metadata Cache (Kb)	JFS Metadata Cache as a percent of memory
256	32000	12.2%
512	64000	12.2%
1024	128000	12.2%
2048	256000	12.2%
8192	512000	6.1%
32768	1024000	3.0%
131072	2048000	1.5%

JFS Metadata Buffer Cache Example



JFS 3.3 w/ dbc_max_pct=20

Buffer cache

400 Mb

2 GB

JFS 3.5 w/ dbc_max_pct=20 and default vx_bc_bufhwm

Buffer cache

Metadata

400 Mb

256 Mb

2 GB

JFS 3.5 w/ dbc_max_pct=18 and vx_bc_bufhwm to 40 Mb

Buffer cache

360 Mb

40 Mb

2 GB

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 shared by HFS and JFS sized by *ncsize* and *vx_ncsize (JFS 3.3 and earlier)*
- Separate DNLC for JFS sized by *vx_ninode (JFS 3.5)*

Large Directories

- 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		

HP WORLD 2004

Solutions and Technology Conference & Expo

Co-produced by:

interex

shared knowledge • shared power

encompass

AN HP USER GROUP

RECOMMENDED TRAINING VENUE FOR THE

HP Certified Professional