Common Misconfigured HP-UX Resources

Mark Ray
Steven Albert
Jan Weaver
Hewlett Packard Company

HP WORLD 2003
Solutions and Technology Conference & Expo

Common Misconfigured HP-UX Resources

- Memory is a finite resource
- Memory is a shared resource
- Operating system (kernel) and user processes compete for memory
- Many kernel resources are configurable (tunable)
- Misconfigured resources can consume a lot of memory
- There is no one set of tunables is best for all systems
- Presentation identifies some commonly misconfigured HP-UX resources

Common Misconfigured HP-UX Resources

- Resources discussed in presentation:
  - The HFS Inode Cache
  - The HP-UX Buffer Cache
  - The JFS Inode Cache
  - The JFS 3.5 Metadata Buffer Cache
  - Semaphore Tables

Common Misconfigured HP-UX Resources

- Presentation will answer the following questions for each resource:
  - What is the purpose of the resource?
  - How is the resource managed?
  - How much memory does each resource require?
  - How can the resource be tuned?
  - Are there any guidelines to consider when configuring the resources?

The HFS Inode Cache

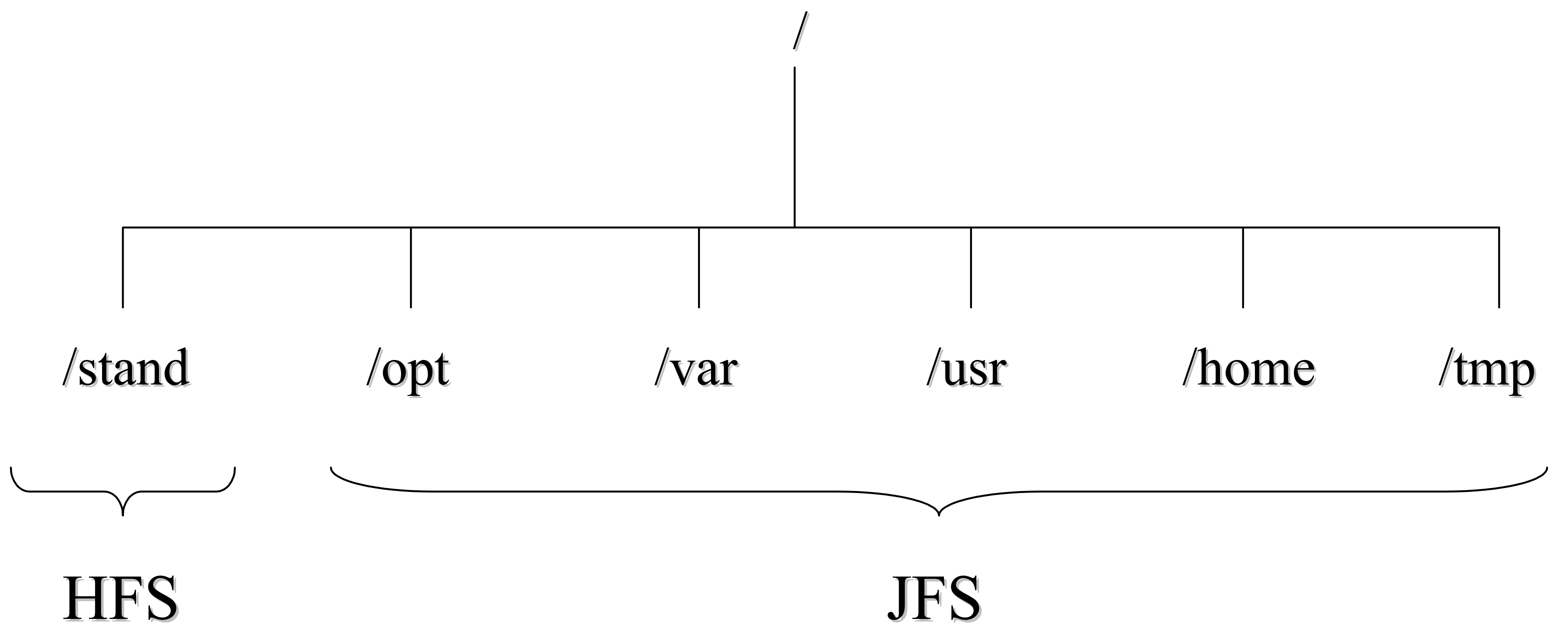
Mark Ray

Global Solutions Engineering
Hewlett Packard Company

HP WORLD 2003
Solutions and Technology Conference & Expo

The HFS Inode Cache

- Directory tree



The HFS Inode Cache

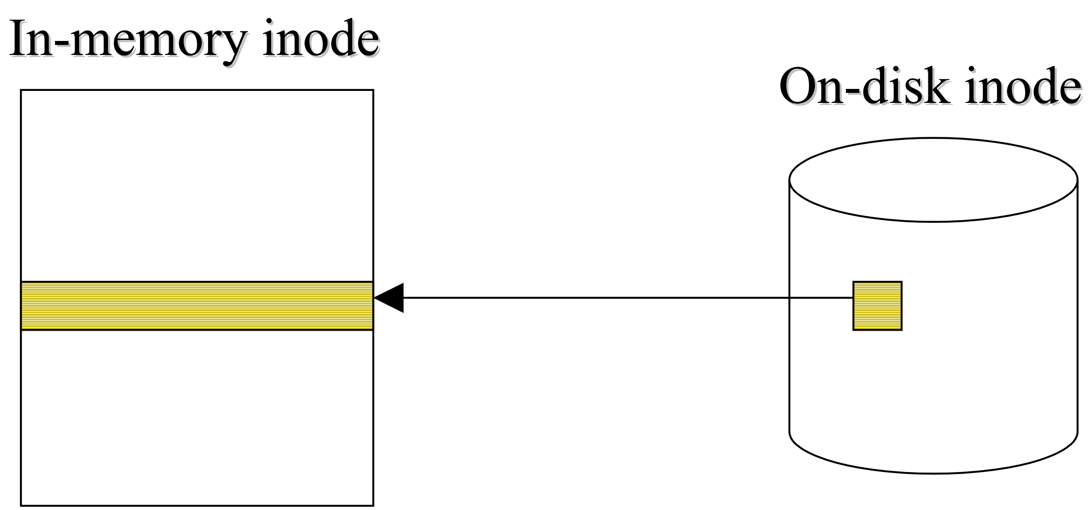
- What is an inode cache?
- How is the HFS inode cache managed?
- How much memory is required for the HFS inode cache?
- What dependencies are there on the HFS inode cache?
- Are there any guidelines for configuring the HFS inode cache?

The HFS Inode Cache

- What is an inode cache?
  - Holding location in memory for inodes on disk
  - One inode entry in cache must exist for every opened file
  - Inodes for closed files are on free list
  - Superset of data from disk

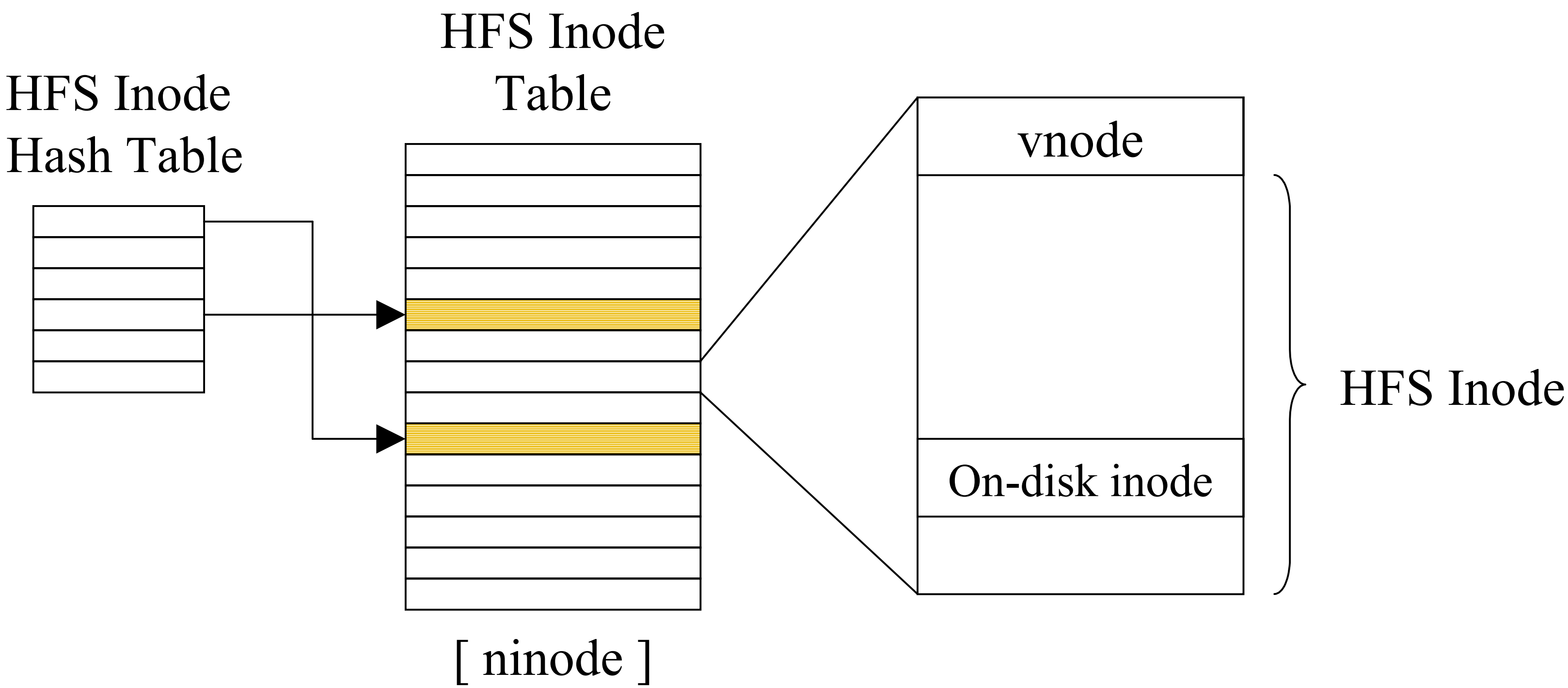
The HFS Inode Cache

- What is an inode cache?
  - On-disk inode - file type, permissions, timestamps, size, block map
  - In-memory inode - on-disk inode, inode number, linked list pointers, pointers to other structures, lock primitives



The HFS Inode Cache

- How is the HFS inode cache managed?



The HFS Inode Cache

- Memory cost of HFS Inode Cache

	10.20	11.0 32-bit	11.0 64-bit	11.11 32-bit	11.11 64 bit
vnode	100	100	176	100	176
inode	316	336	488	367	496
hash entry	8	8	16	8	16

Table 1: Number of bytes per entry

The HFS Inode Cache

- Memory cost of HFS Inode Cache

- Example:
  - HP-UX 11.11 - 64-bit
  - ninode = 10,000
  - 176+496 = 672 bytes per inode
  - 672*10,000 = 6563 Kb for inode cache
  - Previous power of 2 - 8192
  - 16 bytes * 8192 = 128 Kb for inode hash table

The HFS Inode Cache

- The HFS inode cache and the DNLC
  - Default Directory Name Lookup Cache (DNLC) size:

    (NINODE+VX_NCSIZE)+(8*DNLC_HASH_LOCKS)

  - Tunables
    - ncsize - Introduced with PHKL_18335 on 10.20. Determines the size of the DNLC.
    - vx_ncsize - Introduced in 11.0. Used with ncsize to determine the overall size of the DNLC. Obsolete with JFS 3.5.

The HFS Inode Cache

- Configuring the HFS inode cache
  - Default value:

    `((NPROC+16+MAXUSERS)+32+(2*NPTY)`

  - Consider number and size of HFS file systems
  - If only HFS file system is /stand, then...
    - Consider small HFS inode cache.
    - Tune DNLC appropriately for other file systems
  - Configure large enough to all HFS files that are open at a given time
  - If many HFS file systems, default values are good for most systems.
  - Consider larger value for ninode for file servers.

The HPUX Buffer Cache

Jan Weaver
Hewlett Packard

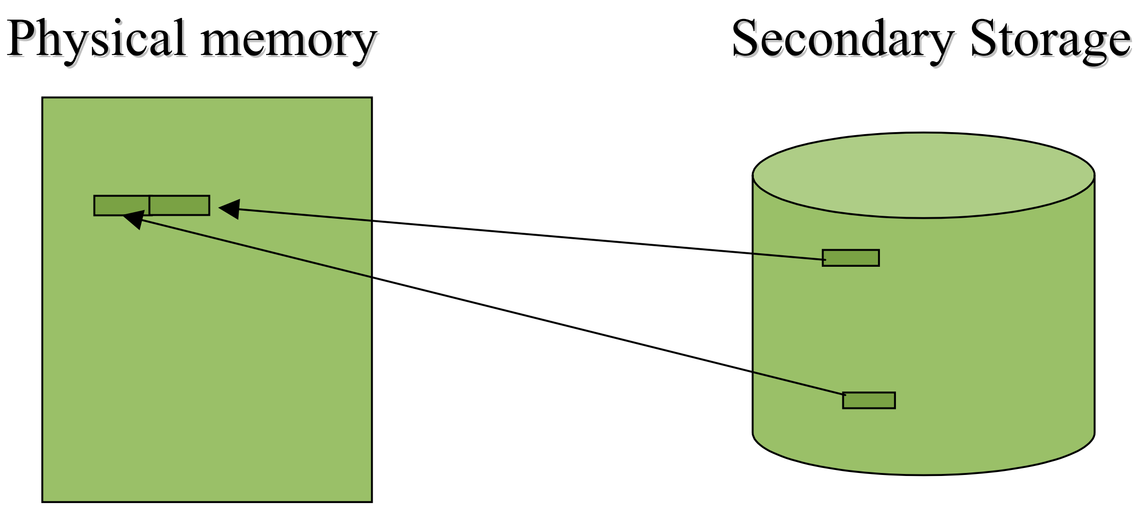
HP WORLD 2003
Solutions and Technology Conference & Expo

The HPUX Buffer Cache

- What is the buffer cache?
- How does a static buffer cache differ from a dynamic buffer cache?
- How does the buffer cache work?
- How much memory is required for the buffer cache and its related structures?
- What are the advantages and disadvantages of using the buffer cache?
- Can the buffer cache be bypassed?
- Are there any guidelines for configuring the buffer cache?

The HPUX Buffer Cache

- What is the Buffer Cache?



The HPUX Buffer Cache

- Dynamic Buffer Cache
  - Default with a max of 50% of memory and a min of 5% of memory
  - Configured by setting dbc_max_pct, dbc_min_pct and nbuf = 0 and bufpages = 0
  - Buffer cache grows until dbc_max_pct is reached. Under memory pressure, memory will be released to the general pool

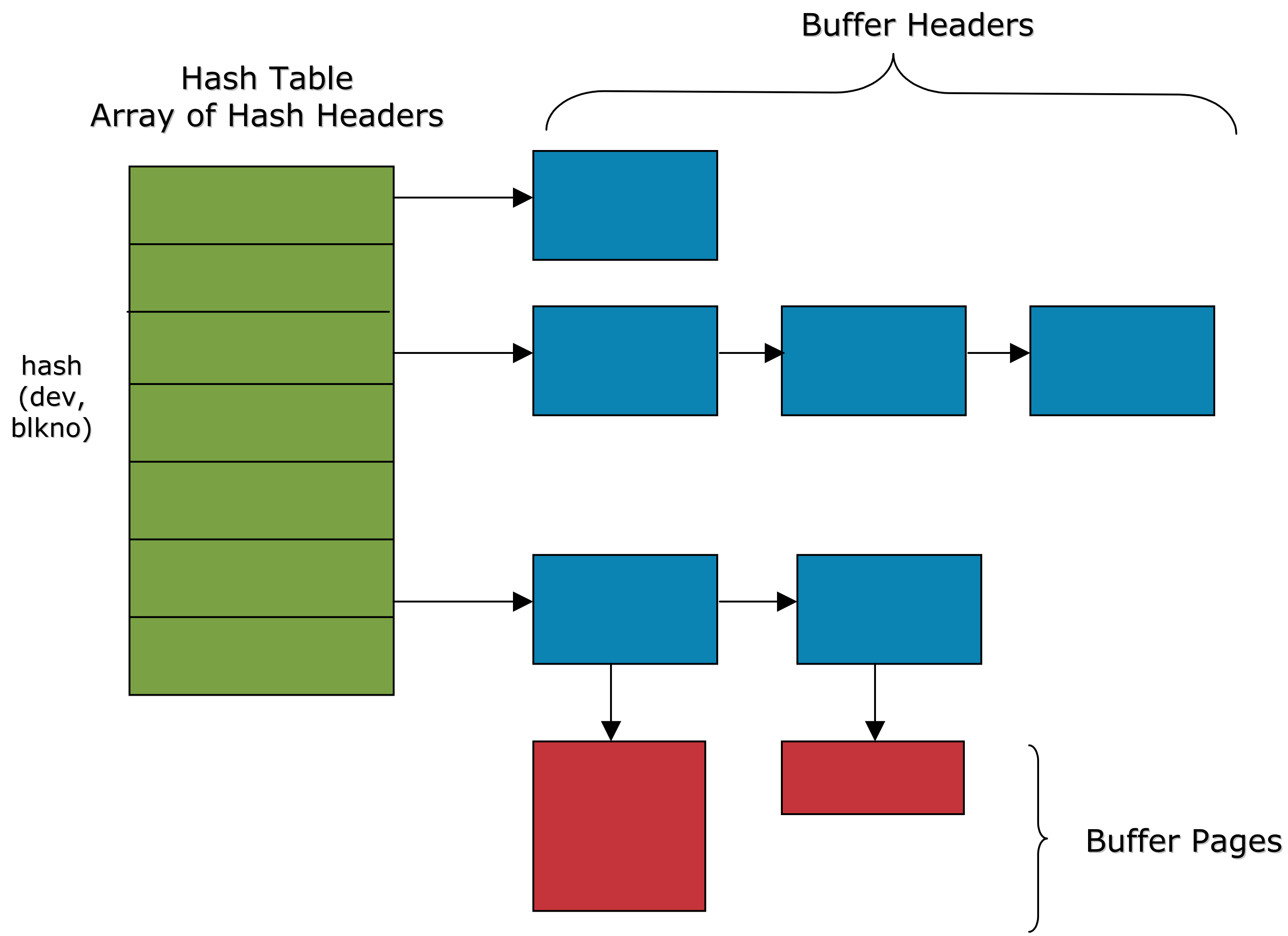
The HPUX Buffer Cache

HP WORLD 2003 Solutions and Technology Conference & Expo

- Static Buffer Cache
  - nbuf and/or bufpages != 0
  - nbuf !=0 you get nbuf headers and nbuf * 2 pages in the buffer cache
  - bufpages !=0 you get bufpages pages in the buffer cache and bufpages/2 buffer headers
  - Memory is permanently allocated to the buffer cache

The HPUX Buffer Cache

- How the Buffer Cache Works



The HPUX Buffer Cache

- How the Buffer Cache Works
  - Hash to the slot in the hash table
  - If buffer is present, use it or wait for the buffer if it is busy.
  - If the buffer is not present, then do one of the following:
    - allocate a new buffer if buffer cache is dynamic, then read the data from the disk
    - reuse an existing buffer (invalid or aged), then read the data from the disk.
  - Note: Buffers remain in the buffer cache even after a file is closed as they could be used again later.

The HPUX Buffer Cache

- Buffer Cache Memory Requirements
  - Buffer Headers
  - Buffer Hash Table
  - Buffer cache Hash Locks
  - Buffer Cache Address Map/Virtual Map

The HPUX Buffer Cache

- Buffer Cache Memory Requirements

System Memory Size	10% Bufpages/Buf Headers	20% Bufpages/Buf Headers	50% Bufpages/Buf Headers
1GB	100MB/7.3MB	200MB/15MB	500MB/36MB
2GB	200MB/15MB	400MB/30MB	1GB/75MB
4GB	400MB/30MB	800MB/60MB	2GB/150MB
8GB	800MB/60MB	1.6GB/120MB	4GB/300MB
12GB	1.2GB/90MB	2.4GB/180MB	6GB/450MB
32GB	3.2GB/240MB	6.4GB/480MB	16GB/1.2GB
256GB	25.6GB/2GB	51GB/4GB	128GB/9.6GB

The HPUX Buffer Cache

- Buffer Cache Memory Requirements

System Memory Size	Hash Table Size	Hash Entries per Lock	Hash Table Memory Requirements
1GB	65536	512	2.5MB
2GB	131072	1024	5MB
4GB	262144	2048	10MB
8GB	524288	4096	20MB
12GB	1048576	8192	40MB
32GB	2097152	16384	80MB
256GB	16777216	131072	640MB

The HPUX Buffer Cache

- Advantages of using Buffer Cache
  - Small sequential I/O
  - Readahead
  - Hot Blocks
  - Delayed Writes

The HPUX Buffer Cache

- Disadvantages of using Buffer Cache
  - Memory
  - Flushing the buffer cache
  - Throttling
  - Large I/O
  - Data accessed once
  - System crash

The HPUX Buffer Cache

- Bypassing the buffer cache
  - mincache=direct, convosync=direct
  - ioctl(fd, VX_SETCACHE,VX_DIRECT
  - discovered_direct_io
  - raw I/O
  - /dev/async

The HPUX Buffer Cache

- Tuning guidelines
  - Favor database global area over buffer cache
  - 11.11 scales better
  - Minimum 200 MB on most systems
  - Check relevant patches

The JFS Inode Cache

Mark Ray

Global Solutions Engineering
Hewlett Packard Company

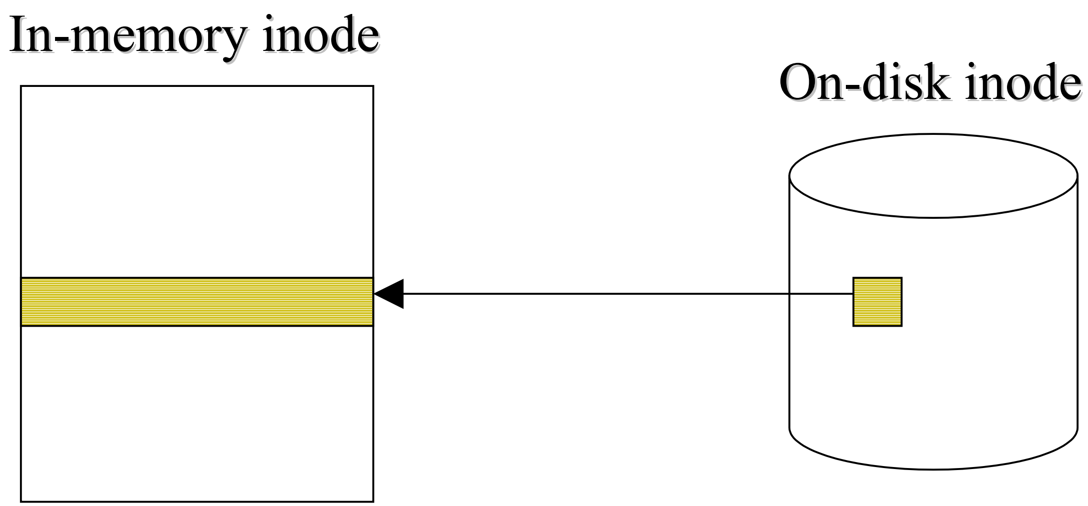
HP WORLD 2003
Solutions and Technology Conference & Expo

The JFS Inode Cache

- What is an inode cache?
- What is the maximum size of the inode cache?
- How can I determine the number of inodes in the cache?
- How can I determine the number of inodes in use?
- How does JFS manage the inode cache?
- How much memory is required for the JFS inode Cache?
- Are there any guidelines for configuring the buffer cache?

The JFS Inode Cache

- What is an inode cache?
  - On-disk inode - file type, permissions, timestamps, size, block map
  - In-memory inode - on-disk inode, inode number, linked list pointers, pointers to other structures, lock primitives



The JFS Inode Cache

- JFS Inode Cache is dynamic
  - Expands and contracts based on need
  - Open() or stat() can cause inode to be brought in from disk to memory
  - Active and inactive inodes are maintained in the cache
  - Inactive inodes belong to a free list.
  - Periodically, free list is scan for inodes to that have not been access for a certain amount of time

The JFS Inode Cache

- Maximum Inodes in the JFS Inode Cache
  - Defaults based on memory size

Memory Size (Mb)	JFS 3.1	JFS 3.3/3.5
256	18666	16000
512	37333	32000
1024	74666	64000
2048	149333	128000
8192	149333	256000
32768	149333	512000
131072	149333	1024000

The JFS Inode Cache

- Maximum Inodes in the JFS Inode Cache
  - adb command to verify maximum size

JFS 3.1 (32-bit)	vxfs_fshead+0x8/D
JFS 3.1 (64-bit)	vxfs_fshead+0xc/D
JFS 3.3	vxfs_ninode/D

```
# echo "vxfs_ninode/D" | adb -k /stand/vmunix /dev/mem
```

  - vxfsstat with JFS 3.5

```
# vxfsstat -v /  | grep maxino
vxi_icache_maxino      128000    vxi_icache_peakino     128002
```

The JFS Inode Cache

- Current Number of Inodes in the JFS Inode Cache
  - adb command to display current size:

JFS 3.1 (32-bit)	vxfs_fshead+0x10/D
JFS 3.1 (64-bit)	vxfs_fshead+0x14/D
JFS 3.3	vxfs_cur_inodes/D

```
# echo "vxfs_cur_inodes/D" | adb -k /stand/vmunix /dev/mem
```

  - vxfsstat with JFS 3.5

```
# vxfsstat -v /  | grep curino
vxi_icache_curino          3087    vxi_icache_inuseino        635
```

The JFS Inode Cache

- Number of Active Inodes in the JFS Inode Cache
  - adb command to display number of active inodes:

JFS 3.1 (32-bit)	vxfs_fshead+0x14/D
JFS 3.1 (64-bit)	vxfs_fshead+0x18/D
JFS 3.3	N/A

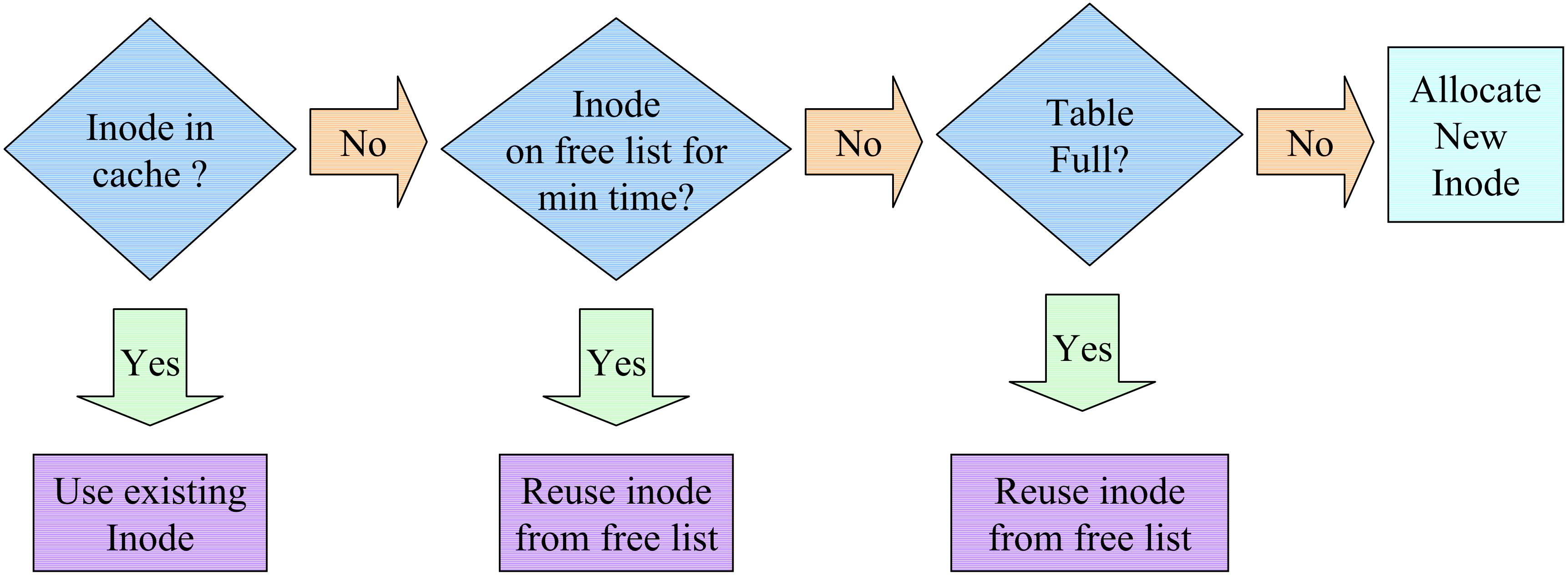
```
# echo "vxfs_fshead+0x18/D" | adb -k /stand/vmunix /dev/mem
```

  - vxfsstat with JFS 3.5

```
# vxfsstat -v /  | grep inuse
vxi_icache_curino      128001     vxi_icache_inuseino       635
```

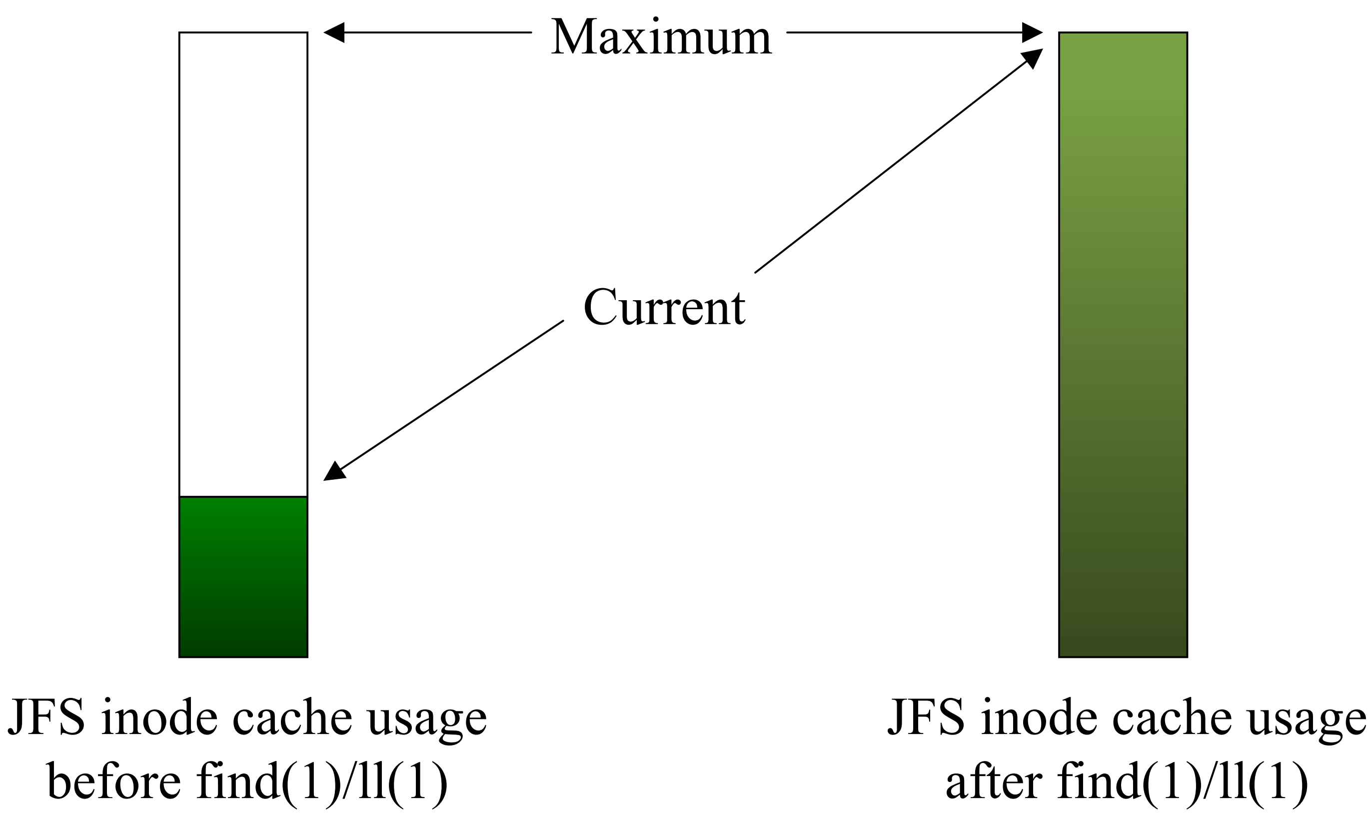
The JFS Inode Cache

- Growing the JFS Inode Cache



The JFS Inode Cache

- Growing the JFS Inode Cache



The JFS Inode Cache

- Growing the JFS Inode Cache

  – vxfsstat(1M) before find(1)

```
# vxfsstat / | grep "inodes current"
  3087 inodes current     128002 peak            128000 maximum
```

  – vxfsstat(1M) after find(1)

```
# vxfsstat / | grep inodes
   128001 inodes current     128002 peak            128000 maximum
```

The JFS Inode Cache

- Shrinking the JFS Inode Cache
  - JFS daemon (vxfsd) thread scans free list
  - If inodes on free list for a given period of time, inode is freed back to the kernel memory allocator
  - JFS inode “free rate” - **vx_ifree_timelag**

	JFS 3.1	JFS 3.3	JFS 3.5
Minimum seconds on free list before being freed	300	500	1800
Maximum inodes to free per second	1/300th of current inodes	50	1-25

The JFS Inode Cache

- Shrinking the JFS Inode Cache
  - vxfsstat(1M) after find(1)/ll(1)

```
# date; vxfsstat -v / | grep -i curino
Thu May  8 16:34:43 MDT 2003
vxi_icache_curino       127526    vxi_icache_inuseino        635
```

  - 134 seconds later

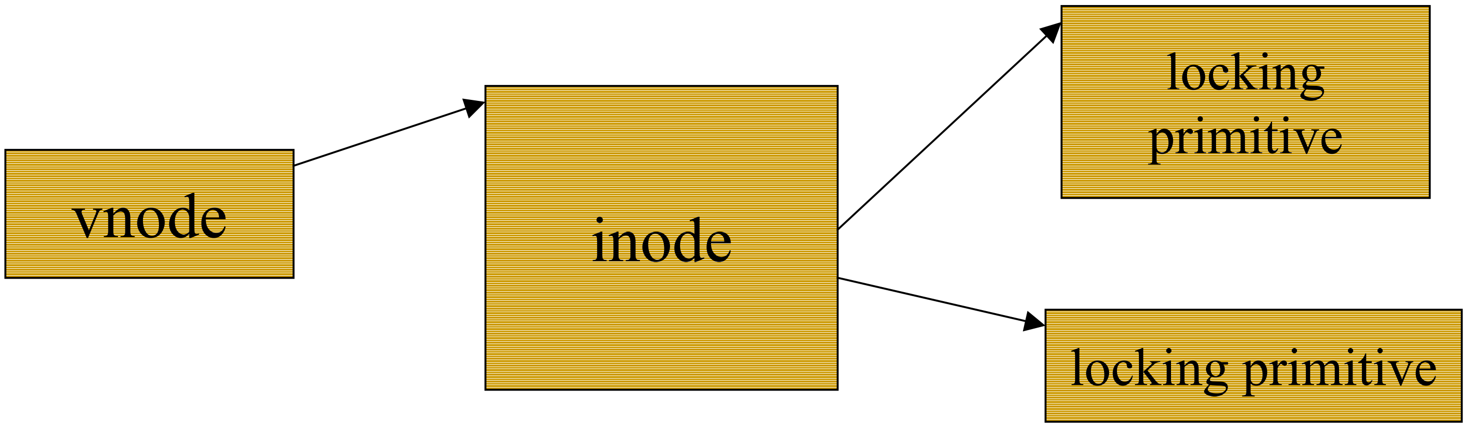
```
# date; vxfsstat -v / | grep -i curino
Thu May  8 16:36:57 MDT 2003
vxi_icache_curino       127101    vxi_icache_inuseino        635
```

  - Next day

```
# date; vxfsstat -v / | grep -i curino
Fri May  9 14:45:31 MDT 2003
vxi_icache_curino         3011    vxi_icache_inuseino        636
```

The JFS Inode Cache

- Memory Cost Associated with each JFS inode
  - Need space for vnode, inode, locking primitives
  - Other space allocated for free list headers, hash headers, enhanced (fancy) readahead, ACLs, and quotas.



The JFS Inode Cache

- Memory Cost Associated with each JFS inode
  - Estimated memory cost per inoded (in bytes)

Structures	JFS 3.1 11.00 32-bit	JFS 3.1 11.00 64-bit	JFS 3.3 11.00 32-bit	JFS 3.3 11.00 64-bit	JFS 3.3 11.11 32-bit	JFS 3.3 11.11 64-bit	JFS 3.5 11.11 64-bit
inode	1024	2048	1024	1024	1024	1364	1364
vnode	*	*	128	256	128	184	184
locks	196	196	352	352	272	384	352
total	1220	2244	1494	1632	1352	1902	1850

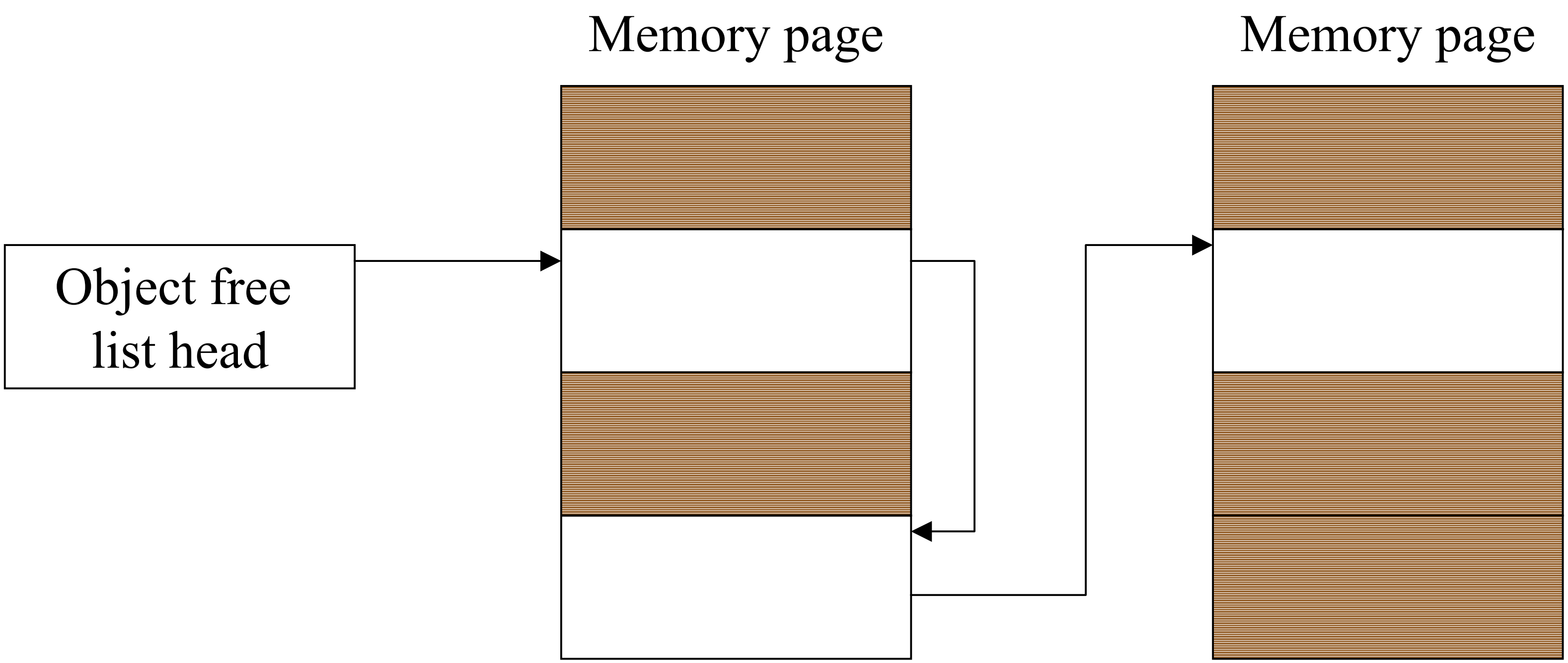
*vnode allocated with inode

The JFS Inode Cache

- Memory Cost Associated with each JFS inode
  - Example 1
    - HP-UX 11.0, 64-bit OS, 2 Gb memory, JFS 3.1
    - 149333 * 1 bytes = 319 Mb or 15% of total memory
  - Example 2
    - Upgrade to JFS 3.3
    - 128000 * 1632 bytes = 200 Mb or 10% of total memory
  - Example 3
    - Add 6 Gb memory (from 2 GB to 8 Gb)
    - 256000 * 1632 bytes = 400 Mb or 5% of total memory

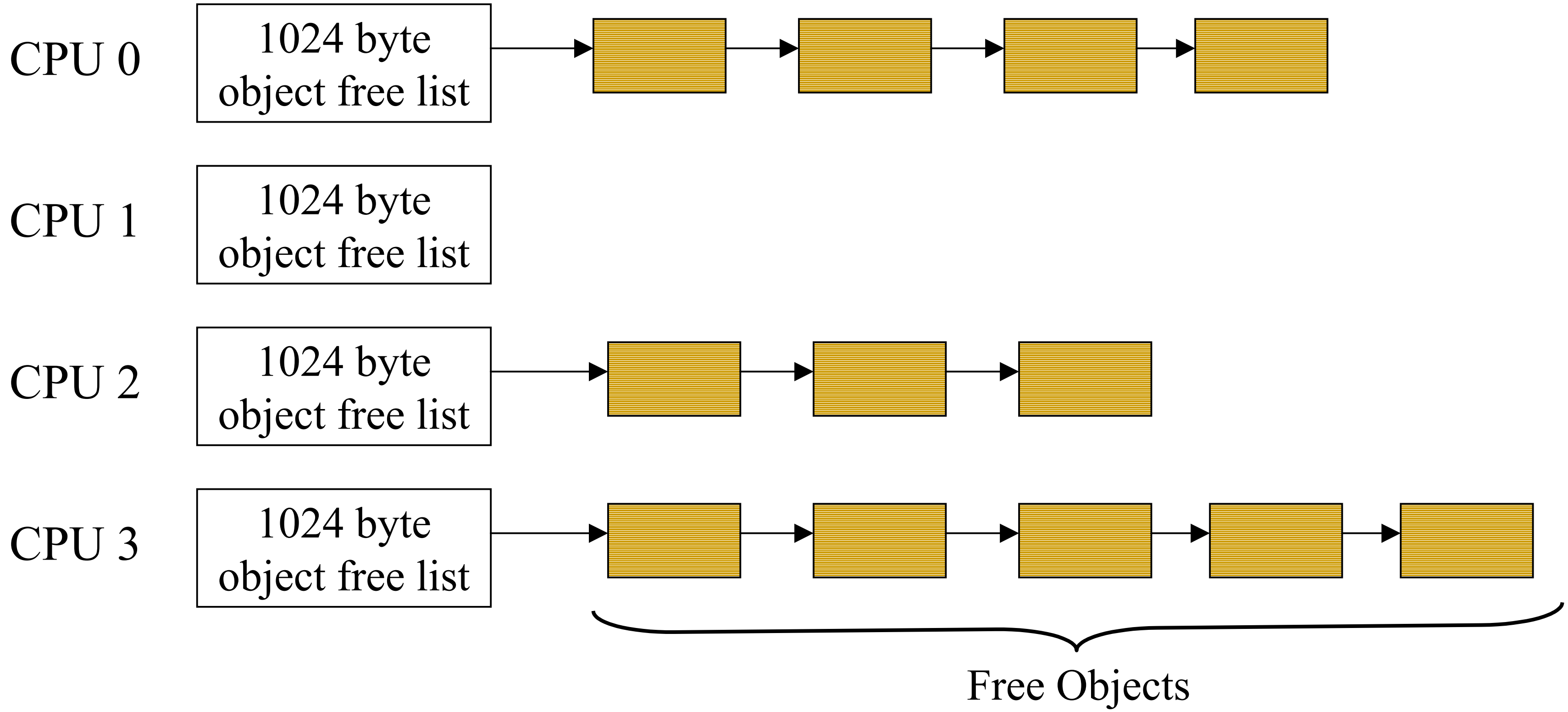
The JFS Inode Cache

- Effects of Kernel Memory Allocator
  - kernel memory allocated from pages and subdivided into equally sized “objects”



The JFS Inode Cache

- Effects of Kernel Memory Allocator



The JFS Inode Cache

- Tuning the Maximum Size of the JFS Inode Cache
  - Every customer environment is unique
  - Must have one inode entry for each file opened at a given time.
  - Most systems will run fine with with 1-2% of memory used for JFS Inode Cache
  - Large file servers (web servers, NFS servers) which randomly access a large set of inodes benefit from the large inode cache
  - Inode cache typically “appears” full after accessing many files sequentially (find(1), ll(1), backups)
  - ninode (HFS tunable) does not impact JFS inode cache

The JFS Inode Cache

- Tuning your System to Use a Static Inode Cache
  - inodes are freed to kernel memory allocator
  - may not be available for immediate use for other objects
  - massive kernel memory allocations and subsequent frees add overhead to the kernel
  - Static inode cache can keep inodes in cache longer
  - System-wide tunable **vx_noifree** available with JFS 3.1 on 11.00.
  - Similar tunable for JFS 3.3/JFS 3.5 does not currently exist.

The JFS Inode Cache

- Summary
  - Memory is a finite resource
  - Plan your usage
  - Consider the following:
    - How are the files accessed
    - Working set size of files being accessed
    - Datebase applications vs. file server
    - File lookup performance vs. memory utilization

The JFS 3.5 Metadata Buffer Cache

Mark Ray

Global Solutions Engineering
Hewlett Packard Company

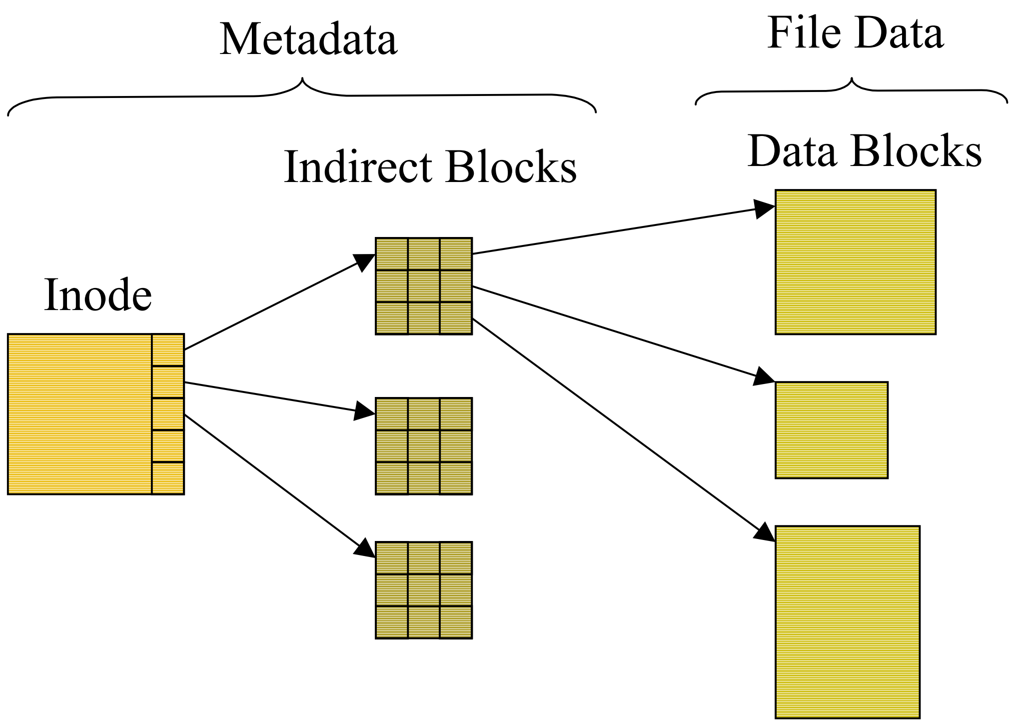
HP WORLD 2003
Solutions and Technology Conference & Expo

JFS 3.5 Metadata Buffer Cache

- New feature in JFS 3.5 introduced on 11.11
- Separate buffer cache managed by JFS for file system metadata
- Increased performance for metadata intensive operations
- Questions to address
  - What is metadata?
  - Is the metadata cache static or dynamic
  - How much memory is required for the metadata cache
  - How can the metadata cache be tuned
  - Are there any guidelines for configuring the metadata cache?

JFS 3.5 Metadata Buffer Cache

- What is Metadata?
  - Structural information - inodes, indirect blocks, bitmaps, and summaries



JFS 3.5 Metadata Buffer Cache

- Metadata Cache - Dynamic or Static
  - Dynamic - expands and contracts over time
  - Expands during heavy metadata usage (find, ll, backup)
  - Expands to maximum amount at bootup
  - Contracts slowly as buffers remain inactive
  - View usage with vxfsstat(1M) command

```
# vxfsstat -b /

12:55:26.640  Thu Jul 3 2003 -- absolute sample

buffer cache statistics
   348416 Kbyte current       356040 maximum
   122861 lookups             98.78% hit rate
```

JFS 3.5 Metadata Buffer Cache

- How much memory is required for the metadata cache?
  - Default Maximum Size (buffer pages only)

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 3.5 Metadata Buffer Cache

- How much memory is required for the metadata cache?
  - Buffer Headers (0.1% - 2.4% of memory)

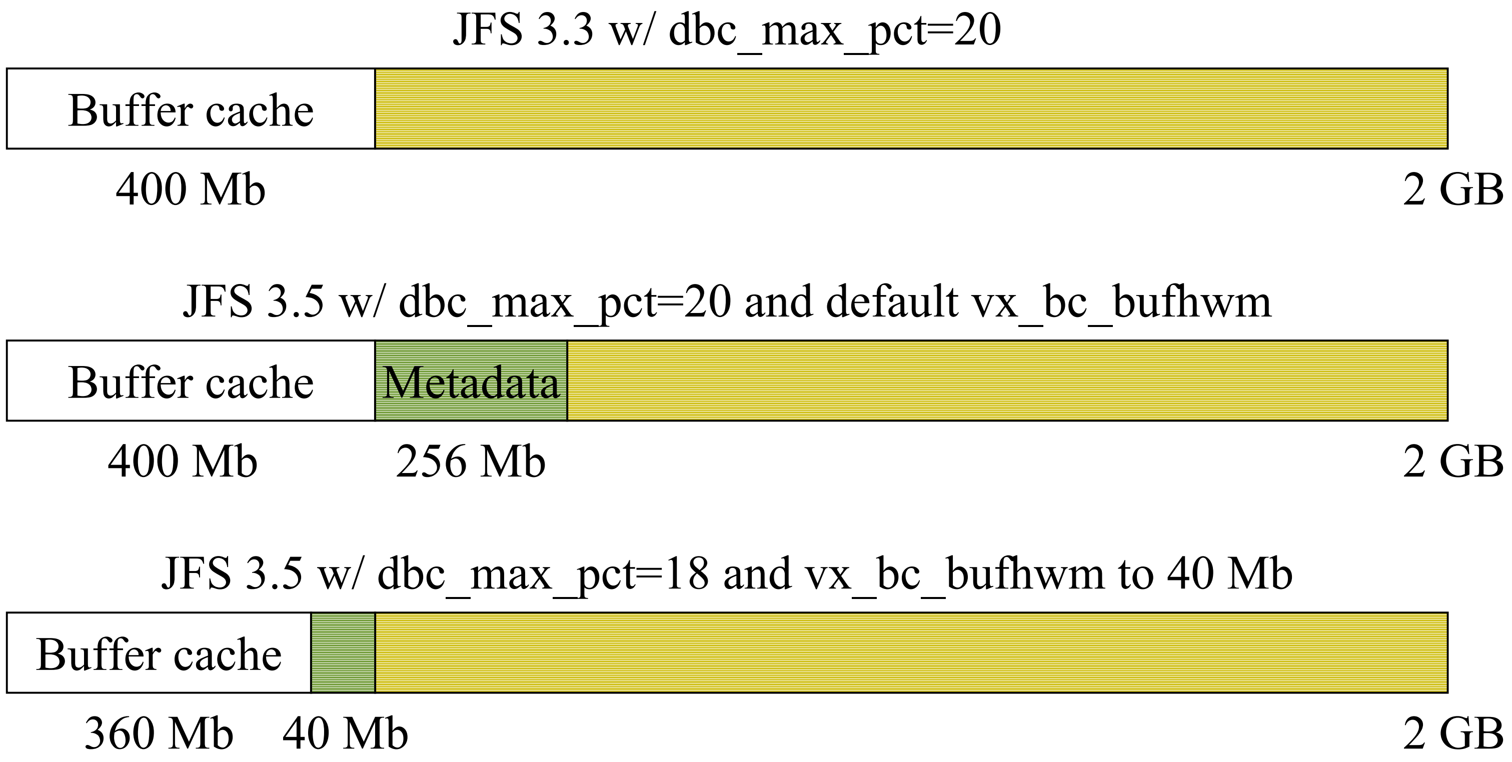
Memory Size (Mb)	JFS Metadata Cache (Kb)	Memory for buffer headers (4kb/buffer)	Memory for buffer headers (8kb/buffer)
256	32000	6.25 Mb	3.13 Mb
512	64000	12.5 Mb	6.25 Mb
1024	128000	25 Mb	12.5 Mb
2048	256000	50 Mb	25 Mb
8192	512000	100 Mb	50 Mb
32768	1024000	200 Mb	100 Mb
131072	2048000	400 Mb	200 Mb

JFS 3.5 Metadata Buffer Cache

- How can the metadata be tuned?
  - System-wide tunable **vx_bc_bufhwm**
  - Specifies maximum amount of memory in kilobytes to be used for the metadata cache’s buffer pages.

JFS 3.5 Metadata Buffer Cache

- Recommendations guidelines for tuning the JFS Metadata Buffer Cache



JFS 3.5 Metadata Buffer Cache

- Recommendations guidelines for tuning the JFS Metadata Buffer Cache
  - Memory usage is above and beyond what is already used by the HP-UX buffer cache
  - Trade memory for increased performance of metadata intensive applications
  - Consider total amount of memory HP-UX Buffer Cache and JFS Metadata Buffer Cache can take.
  - Estimate ratio of file data to metadata and configure **dbc_max_pct** and **vx_bc_bufhwm** accordingly.

Semaphore Tables

Steven Albert

Global Solutions Engineering
Hewlett-Packard

HP WORLD 2003
Solutions and Technology Conference & Expo

What is a semaphore?

- A construct to control access to a set of resources

Interfaces

- **semget**() – allocate a set of semaphores
- **semop**() – operate on a set of semaphores
- **semctl**() – perform control operations on a set of semaphores

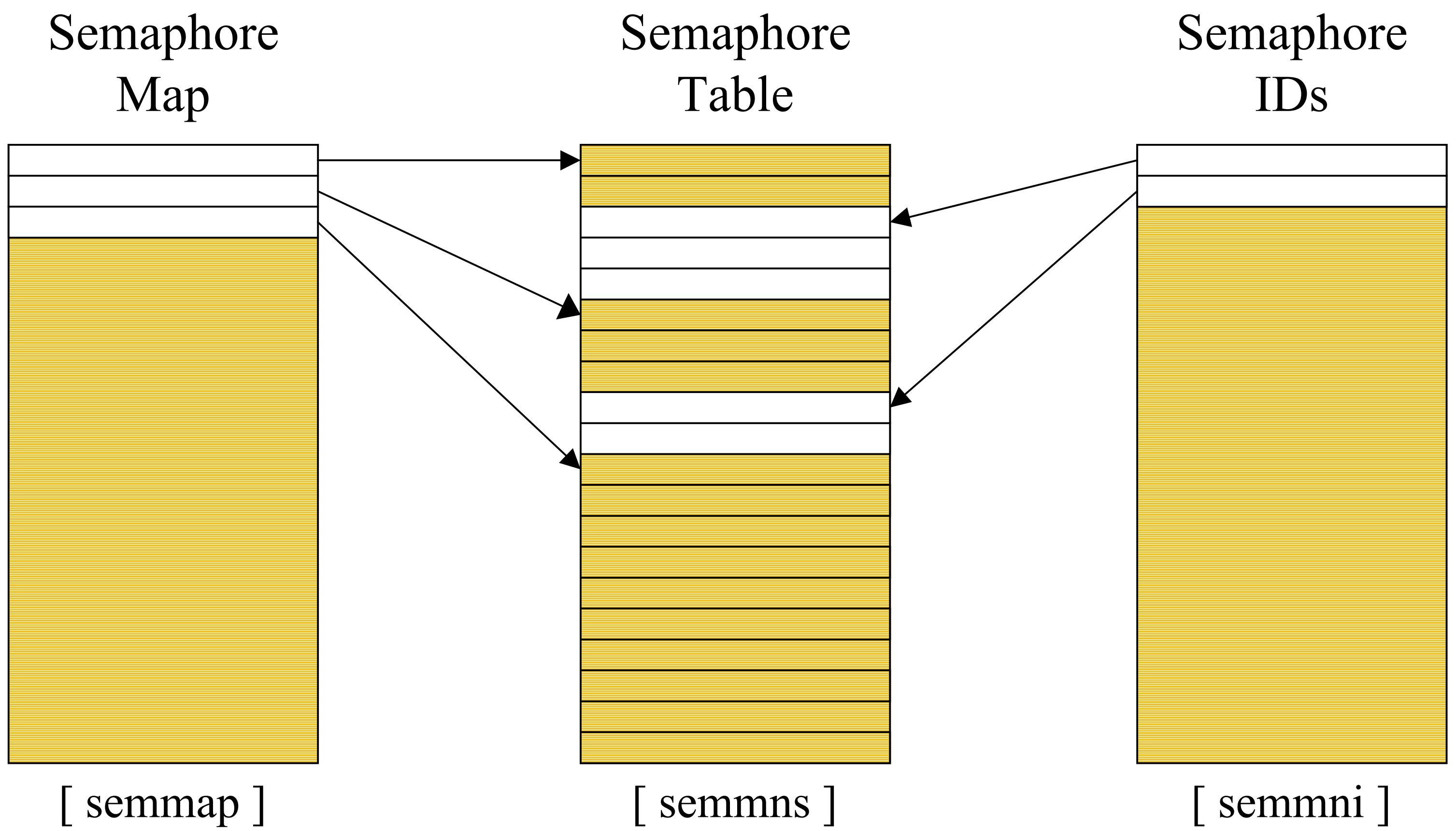
Tunables

- **sema** – enable or disable System V IPC semaphores at boot time
- **semaem** – maximum cumulative value changes per **semop**() call
- **semmap** – # of entries in a semaphore map
- **semmni** – # of system-wide semaphore identifiers
- **semmns** – # of system-wide semaphores
- **semmnu** – # of system-wide semaphore undo structures

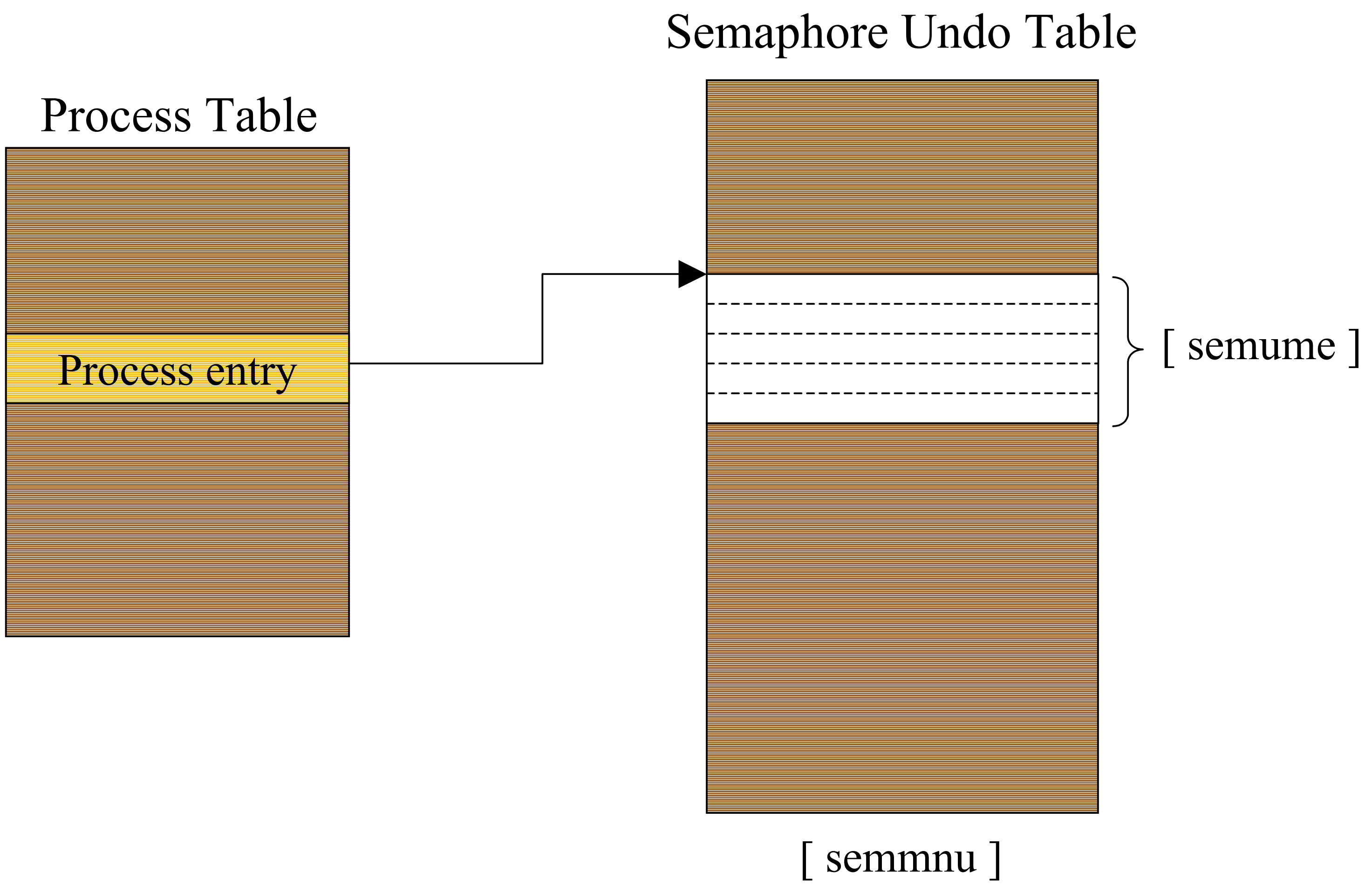
Tunables (cont)

- **semmsl** – maximum # of semaphores per identifier
- **semume** – maximum # of undo entries per process
- **semvmx** – maximum value of any single semaphore

Data Structure Linkage



Data Structure Linkage (cont)



Data Element Sizes

Kernel Table	Tunable	Element Size	Default Setting
Semaphore Map	semmap	8 bytes	semmni+2
Semaphore Ids	semmni	88 bytes	64
Semaphore Table	semmns	8 bytes	128
-	semmsl	-	2048
Semaphore Undo Table	semmnu	24 bytes + (8 * semume)	30 (semmnu) 10 (semume)

Dynamic Allocation Considerations

- Dynamic allocation introduced in PHKL_26136 (11.0) and PHKL_26183 (11.11)
- Semaphore map is obsolete
- Semaphores allocated as needed and freed when not in use
- PHKL_28703 (11.11) increases **semmns** limit to 1,000,000

Summary

- Semaphores are heavily used by many third party applications
- Follow the guidelines for changing the tunables and understand the interactions

HP WORLD 2003
Solutions and Technology Conference & Expo

Interex, Encompass and HP bring you a powerful new HP World.

interex
shared knowledge • shared power

encompass
AN HP USER GROUP

hp
invent