



HP-UX 11i Knowledge-on-Demand

HP technical Webcast series: software optimization



Technology for better business outcomes

HP-UX 11i v3 Knowledge-on-Demand

- Objective: Support software development partners and customers in achieving better business outcomes with HP-UX 11i.
- What HP is providing: a series of technical on-demand training Webcasts
 - Focused on helping developers increase performance through application optimization for HP-UX 11i v3 on HP Integrity servers
 - Access to HP for follow-up questions
 - Available at www.hp.com/go/knowledgeondemand

HP-UX 11i v3 Knowledge-on-Demand Webinars – planned curriculum

- Foundation Track
 - Module 1: How to upgrade to HP-UX 11i v3
 - Module 2: HP-UX open source resources
 - Module 3: Unified file cache
 - Module 4: Caliper
 - Module 5: NUMA Tuning: Getting the Most Out of Your Cellular Server by using NUMA
 - Module 6: The Mercury Library – Increasing Application Performance
 - Module 7: Software Transition Kit's (STK's) for HP-UX 11i v3
- Java Developers Track
 - Module 8: Java Memory Management - Internals and Performance
 - Module 9: HPjmeter – measure Java application performance on HP-UX 11i
 - Module 10: Solving Java performance problems
- C/C++ Developers Track
 - Module 11: pthreads enhancements in HP-UX 11i v3
 - Module 12: Kernel tracing & profiling tools (internal tools)
 - Module 13: Using compilers to get optimal performance
 - Module 14: HP Code Advisor: A Powerful New C/C++ Analysis Tool for HP-UX
 - Module 15: Montecito Hyper-Threading on HP-UX 11i v3

Additional Webinars
published going forward!

Related HP-UX 11i v3 resources

- All developers' resources
 - HP-UX 11i developers' content
www.hp.com/go/hpuxdev
 - HP-UX 11i v3 news, functionality, product download and services resources
www.hp.com/go/hpux11i
 - HP Integrity server ISV resources for DSPP members
www.hp.com/go/dspp_integrity
 - HP Integrity server product information
www.hp.com/go/integrity
- Software partner promotional opportunity
 - HP promotion for HP-UX 11i v3-ready software partner application
www.hp.com/go/v3promotion

Enjoy this Knowledge-on-Demand topic!

Thank you for taking time to learn about HP-UX 11i v3 and related technologies.

Please send comments on today's topic and/or requests for future topics to:

hpuxquestions@hp.com



Unified File Cache

An HP-UX 11i Knowledge-on-Demand software optimization Webcast



Technology for better business outcomes

Introducing today's speaker

- Saleem Mohideen joined HP in August, 1988. He has been working in the HP-UX kernel group for over 13 years. He is a system architect for the HP-UX Virtual Memory subsystem. He has made significant contributions to the HP-UX Virtual Memory subsystem.
- Saleem Mohideen was the technical lead for the Unified File Cache project. He helped lead the project as well as made significant code changes to implement a unified file cache.

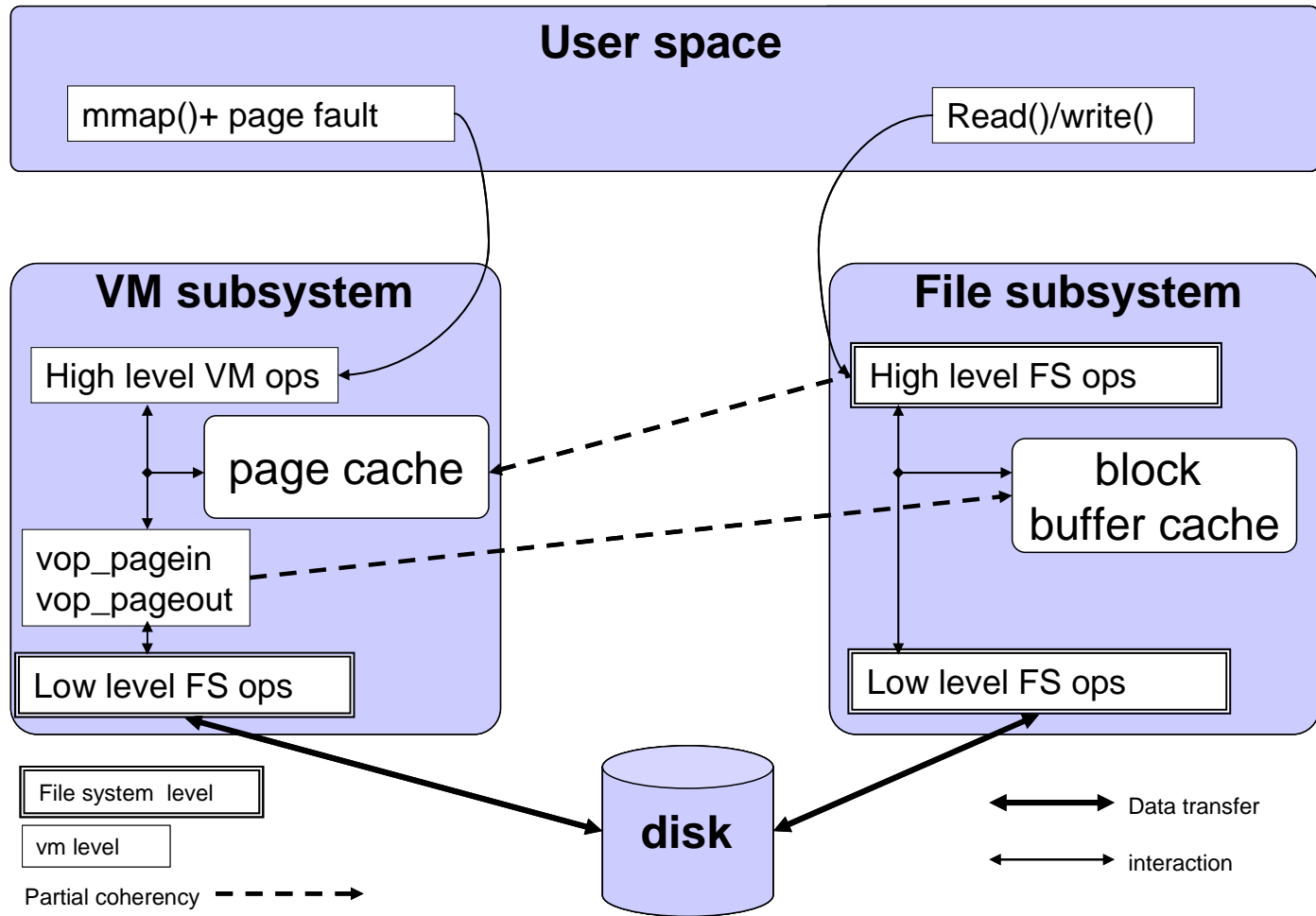
Agenda

- Motivation
- UFC architecture
- Key features
- Application and file system impact
- Performance data
- Summary

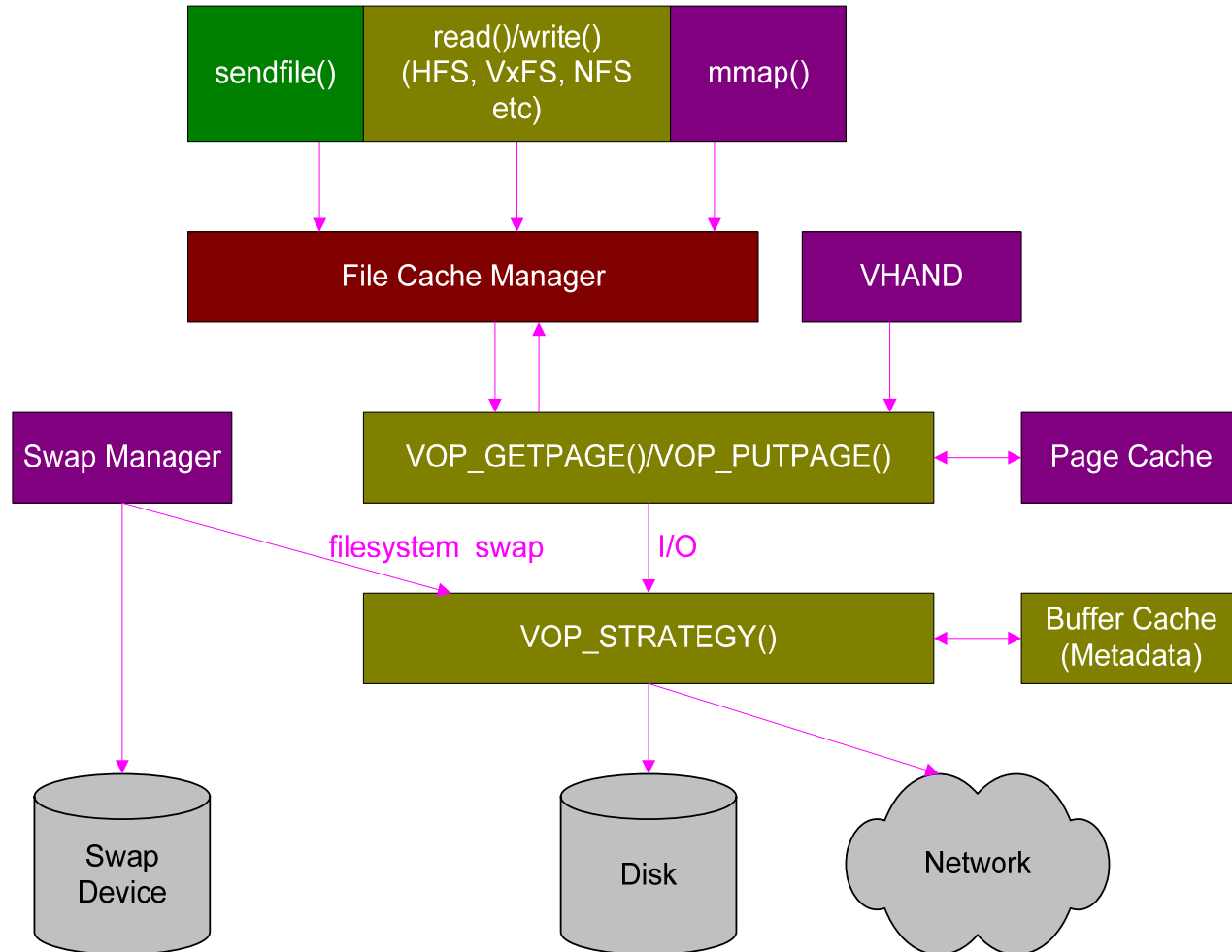
Motivation

- To improve application portability by providing coherency for file access for read()/write and mmap() system calls
- To improve performance for mixed-mode applications: Mixed mode applications access files using both read()/write() system calls and memory mapped address
- To provide well architected, modular, and expandable kernel interfaces to file systems: These common interfaces will enable port of NFS(ONC 2.3), VxFS and ease port of third party file systems from other operating systems, (e.g. Linux, Solaris)
- HP-UX is the only major Unix vendor that does not support a unified cache

HP-UX 11i v2 architecture



HP-UX 11i v3 UFC architecture



Key improvements vs. buffer cache

- All file access in kernel space is virtualized
- VM has been enhanced to allow faults in kernel address space. This is limited to UFC today but can be enhanced in the future to allow other types of kernel access.
- Pages are cached until the page is re-used by another process. Pages removed from the buffer cache cause an I/O. This should improve overall performance.
- File pages cached on behalf of kernel access can be paged out similar to user pages. This allows easier paging and helps OLD (online deletion of cells) and memory migration (HPVM and vpars) to evacuate kernel pages (only UFC).

Key features

- Unified page cache and buffer cache for coherency
- UFC interface is a “best of class” of HP-UX, Solaris and Tru64 interfaces
- Modular interfaces for easy port of file systems to HP-UX
- Very scalable architecture
- Large page support for both read()/write() access and memory mapped files
- ccNuma support for both read()/write() access and memory mapped files
- File level attributes that can be controlled at file level

Application impact

- There is no application impact. All applications that use Unix standard interfaces are binary compatible.
- Applications that worked around or did not port to HP-UX due to lack of coherency in HP-UX, can now take advantage of file cache coherency in HP-UX 11i v3

File system impact

- All physical file systems need to port to new UFC interfaces
- File systems that use old interfaces will not work in 11.31
- Stacked file systems that are layered on top of physical file systems may not be impacted
- All file systems developed by HP have been ported to UFC. This includes HFS, NFS (ONC 2.3), CDFS, and UDF.
- VxFS 4.1 has already been ported
- We are working closely with all impacted ISVs

Key benefits of UFC

- Applications that access files in mixed mode (i.e. use read(), write(), and mmap() simultaneously to access files)
- Number of applications that do mixed mode is increasing because direct memory map access is much faster than read()/write()
- Examples are Netscape Messaging server, sendmail, Samba server, etc.
- There should be a noticeable performance improvement for these applications since no flushing is needed when switching between read()/write() and mmap'd access

File cache tunables

- `dbc_min_pct` and `dbc_max_pct` has been replaced with `filecache_min` and `filecache_max`
- New tunables, `filecache_min` and `filecache_max` are compatible with `dbc_min` and `dbc_max` tunables and maintain the same semantics. The new tunables can be specified as a percent or an absolute value.
- The tunables control the memory size for `read()/write()` access
- The tunables do not limit memory mapped access
- Metadata is maintained in buffer cache and is controlled by a small percentage of the `filecache_min` and `filecache_max` tunables
- Tunables `bufpages` and `nbuf` have been removed

fcntl() system call

- Gives file level control
- Application can specify attributes at a file level such as syncer frequency, large page, ccNuma

Kernel address space

- For read()/write() system calls file system will map files into kernel address space; this portion of the kernel address space is called fcache address space
- Architecture similar to user address space with vas, preregions, and regions
- All VM data structures are cached on an unmap unlike user address space
- Default for the size of the kernel address space is 1TB for PA and 1Petabyte for IPF
- fcache address space is faultable just like user address space (helps memory evacuation for OL*, memory migration and vhand)

Page cache

- All file data is stored in the page cache
- Pages in the page cache are stored in the hash table and are indexed by the $\langle \text{vnode}, \text{file offset} \rangle$ tuple
- Each vnode has a list of pages that are in the page cache

Sendfile – Introduction

- Before Sendfile()
 - File was mmaped and send() was used
 - Data was copied from user to kernel buffer
- Sendfile() on buffer cache
 - Use buffer cache kernel address
 - Avoid copy from user to kernel
- Sendfile() on UFC
 - Use fcache kernel address
 - Avoid copy from user to kernel

Sendfile – Users

- Used by FTP
- Used by web-server applications
- Used by SpecWeb99 benchmark

Stealback

- Ported from Tru64
- Avoids cache-wiping
- Pages are stolen from the same file if it exceeds a certain threshold for sequential access
- Increases performance for sequential access
- Two tunables control the amount of stealback
 - `fcache_seqlimit_system` defines the amount of memory consumed in `ufc mrg`, when stealback kicks in
 - `fcache_seqlimit_file` defines the amount of memory allowed to be consumed by the file for sequential access
- This feature is off by default since it causes performance issues for certain workloads

Performance analysis

- Detailed performance analysis is being done with major file system benchmarks
- Benchmarks being considered are SDET, iozone, postmark, Nastran, Gaussian, and specweb

Results

- Based on preliminary data-not yet updated with final results
- % delta from 11.23/VxFS 3.5

sdet results

This is the familiar SDET benchmark, run on physical disk devices.

| IA 4-way | IA 8-way | IA 64-way | PA 4-way | PA 8-way | PA 64-way |
|-------------|-------------|--------------|-------------|-------------|--------------|
| +5% | +23% | +50% | -3% | +2% | +4% |

Mount results

A measure of the time required to mount a file system, averaged across many mounts

| IA 4-way | IA 8-way | IA 64-way | PA 4-way | PA 8-way | PA 64-way |
|-------------|-------------|--------------|-------------|-------------|--------------|
| +44% | +27% | +65% | +56% | +52% | +73% |

Unmount results

A measure of the time required to unmount file systems (after SDET has used them)

| IA 4-way | IA 8-way | IA 64-way | PA 4-way | PA 8-way | PA 64-way |
|-------------|-------------|--------------|-------------|-------------|--------------|
| +14% | +37% | +63% | -18% | -5% | +47% |

iozone scale results

This is the iozone benchmark, set up for multiple processes to do I/O to multiple files (and to a single shared file). This configuration tests I/O scalability and maximum possible throughput.

| | IA 4-way | IA 8-way | IA 64-way | PA 4-way | PA 8-way | PA 64-way |
|----------------------------|-------------|-------------|--------------|-------------|-------------|--------------|
| Children/ Parent See | +117%/31% | +84%/+18% | +141%/+60% | +48%/-3% | +100%/+44% | +55%/+7% |

PostMark results (average of 10 tests)

PostMark is a single-threaded, meta-data intensive benchmark that stresses directory lookup and file creation/deletion.

| IA 4-way | IA 8-way | PA 4-way | PA 8-way |
|-------------|-------------|-------------|-------------|
| -15.3% | -18.4% | -10% | -17.8% |

This regression will be addressed in future patches and newer VxFS releases.

iozone az results (average)

This is the iozone benchmark, set up for a single process to test a variety of read/write operations using files and record lengths (I/O buffers) of varying sizes, allowing for a total of about 300 different comparisons.

| IA 4-way | IA 8-way | PA 4-way | PA 8-way |
|-------------|-------------|-------------|-------------|
| +34% | +43% | +28% | +32% |

This is the average of many types of tests, file sizes, and record lengths.

- There are cases where the results are significantly **above parity** and **some cases where they are below parity, particularly re-writes and random write with smaller record sizes and files.**

iozone mmap results (average)

This is a variant of iozone -az, where all I/O is done via mmap interfaces instead of syscall like read()/write().

| IA 4-way | IA 8-way | PA 4-way | PA 8-way |
|-------------|-------------|-------------|-------------|
| +166% | +147% | +130% | +106% |

This is the average of many types of tests, file sizes, and record lengths.

* Many results are above parity but some are below parity.

Nastran results

| |
|-------------|
| IA 4-way |
| +1% |

Mixed mode application

- Wrote a small application to simulate mixed mode
- The application accesses the file data using memory mapped address immediately after a write()
- 11i v3 is 32x faster than 11i v2
- The main reason is that on 11i v3, we do not need to flush the data to disk before adding the translation for memory mapped address

Future UFC enhancements

- Analyze and improve ccNuma behavior for file data
- Variable pfdat: This will decrease memory consumption due to pfdats
- Page teardown and creation is expensive. To avoid this, re-use pages when a vnode is unlinked and re-used for another file.
- Improve paging efficiency
- Avoid cpu-binding of syncer threads
- Reduce memory utilization of VM data structures
- Use multiple mrgs for file cache data

Summary

- Unified File Cache improves application portability and file system portability
- Unified File Cache improves performance of mixed-mode applications
- Available in HP-UX 11i v3
- Detailed UFC ERS (External Reference Specification) and file system porting kit available for file system developers