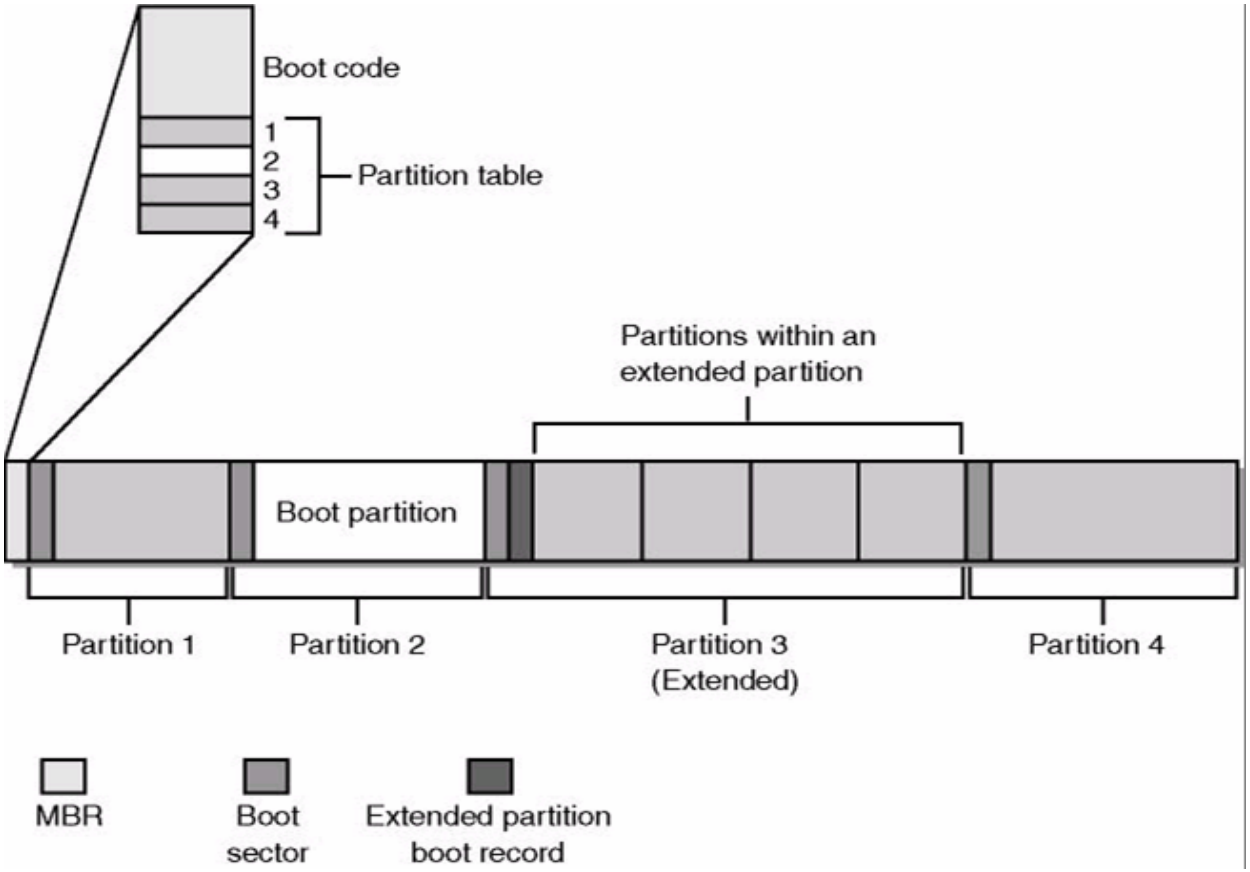
Windows NT File System

„Ausgewählte Betriebssysteme“
Institut Betriebssysteme
Fakultät Informatik

Outline

- Storage Management
  - Partitioning
  - Multi-disk Volumes
- Cache Manager
- NTFS
  - Next class

Disk Partitioning



Boot Process

- BIOS read MBR and executes code
- Code uses partition table to allocate active partition
- Reads 1. sector from it and executes it

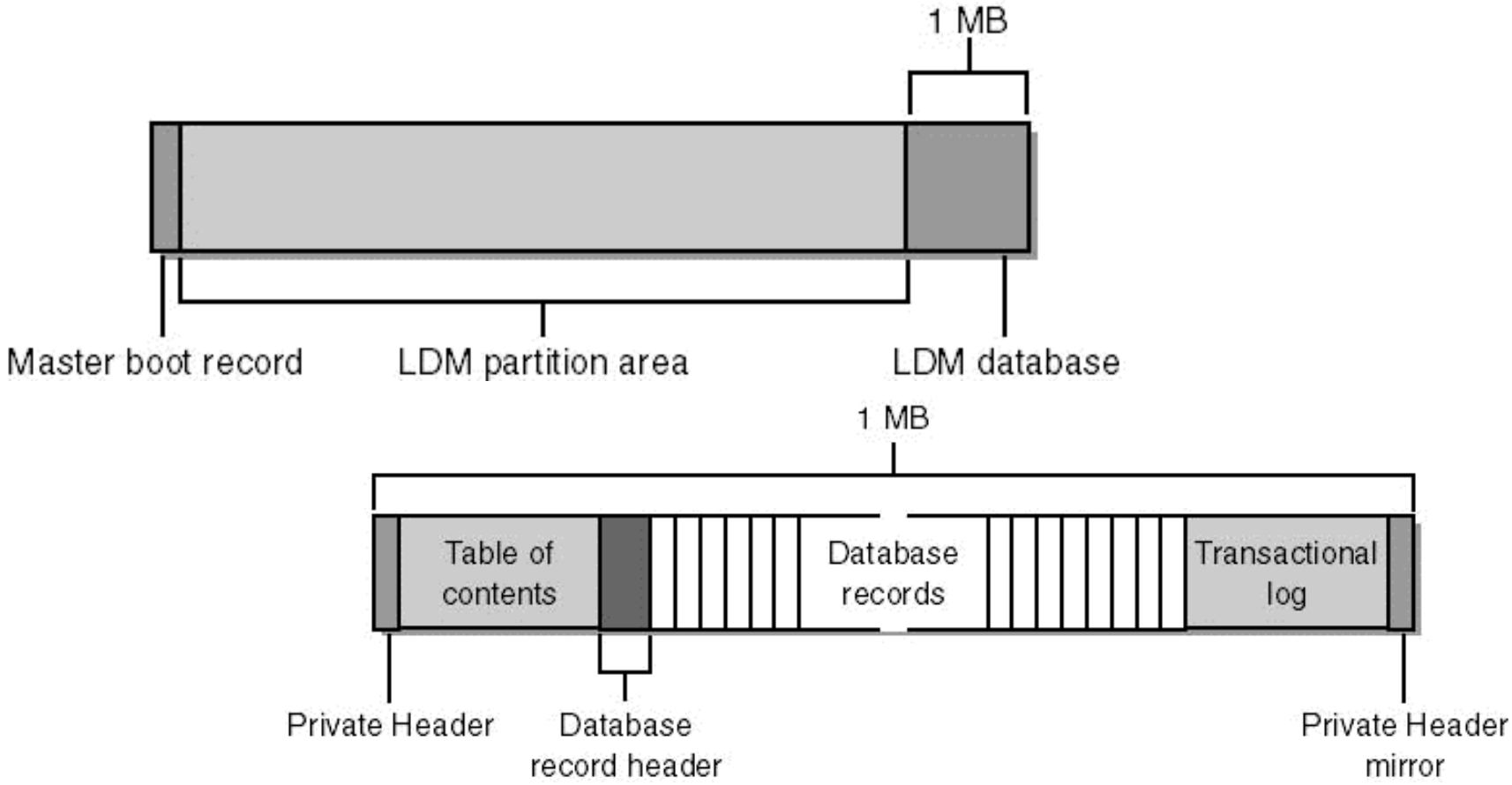
Partition Table

- 4 partitions defined in MBR
- Primary or extended partitions
- Extended partition contains MBR, which allows further partitions
- Partition table entry contains:
  - Type of partition (FAT, NTFS, Linux swap, ...)
  - Start of partition
- „Basic partitions“

„Dynamic Partitioning“

- Logical Disk Manager (LDM) ported to Windows
- Define partition of type „LDM“
- LDM database in reserved last 1 MB of partition
- Provides „old style“ partition table for legacy applications (Ntldr, ...)
- Not used for laptops, disks on IEEE 1394 and USB buses as well as shared cluster servers

„Dynamic Partitioning“ (2)



LDM Database

- Private header contains
  - GUID of disk
  - Name of disk group (default Win2K has only one)
  - Pointer to begin of database table of contents
- Table of contents:
  - Size: 16 sectors
  - Contains information about layout of database
- Database record header:
  - Number of records
  - Name and GUID of disk group
  - Next free entry
- Transactional log

LDM Database Entry

- 128 byte fixed-size record
- Four types:
  - Partition = contiguous region on disk
  - Component = connector between one or more partitions and volume
  - Volume stores GUID, total size, state, drive-letter hint
  - Disk represents dynamic disk
- Entry may span multiple records

Sample Output

```
---------- Dynamic Disk Information -----------
DiskGroup: Btsdelln2Dg0
Group-ID: e81df72b-f373-41d2-9a5a-351fbc4928ca

 Subdisk  Rel Sec   Tot Sec  Tot Size    Plex       Vol Type  Col/Ord  DevName    State
 ========  =======   =======  ========    ====       ========  =======  =========  ======
 LDM-DATA  0         0

 Disk2-01  0         4096512    0       Volume1-01    Simple    1/1      MISSING
 Disk2-02  4096575   4096512    0       Volume2-01    Simple    1/1      MISSING
 Disk2-03  8193150   4096512    0       Volume3-01    Simple    1/1      MISSING
 Disk2-04  12289725  4096512    0       Volume4-01    Simple    1/1      MISSING
 Disk2-05  16386300  4096000    0       Volume8-01    Simple    1/1      MISSING
 LDM-DATA  0         0

 Disk3-01  63        4096512  17771136  Volume5-01    Simple    1/1      Harddisk9  ONLINE
 Disk3-02  4096638   4096512  17771136  Volume6-01    Simple    1/1      Harddisk9  ONLINE
 Disk3-03  8193213   4096512  17771136  Volume7-01    Simple    1/1      Harddisk9  ONLINE
 Disk3-04  12289788  4096000  17771136  Volume9-01    Simple    1/1      Harddisk9  ONLINE
 LDM-DATA  17769088  2048
```

Dynamic Storage Terms

- *Volume:* storage unit made from free space on one or more disks
- *Simple volume:* uses free space from a single disk (can be a single region or consist of multiple, concatenated regions)
- *Spanned volume:*
  - Linked together from multiple disks (up 32 disks)
  - Can be extended onto additional disks
  - Cannot be mirrored
- *Mirrored volume:*
  - Fault-tolerant
  - Data duplicated on two physical disks (RAID-1)

Dynamic Storage Terms (2)

- *Striped volume:* data is interleaved across two or more physical disks. (RAID-0)
- *RAID-5 volume:*
  - Fault-tolerant
  - Data is striped across an array of three or more disks
  - Parity is also striped across the disk array.
- *System volume:* contains hardware-specific files needed to load Win2K (Ntldr, Boot.ini, Ntdetect.com)
- *Boot volume:* contains Win2K operating system files (%Systemroot%)

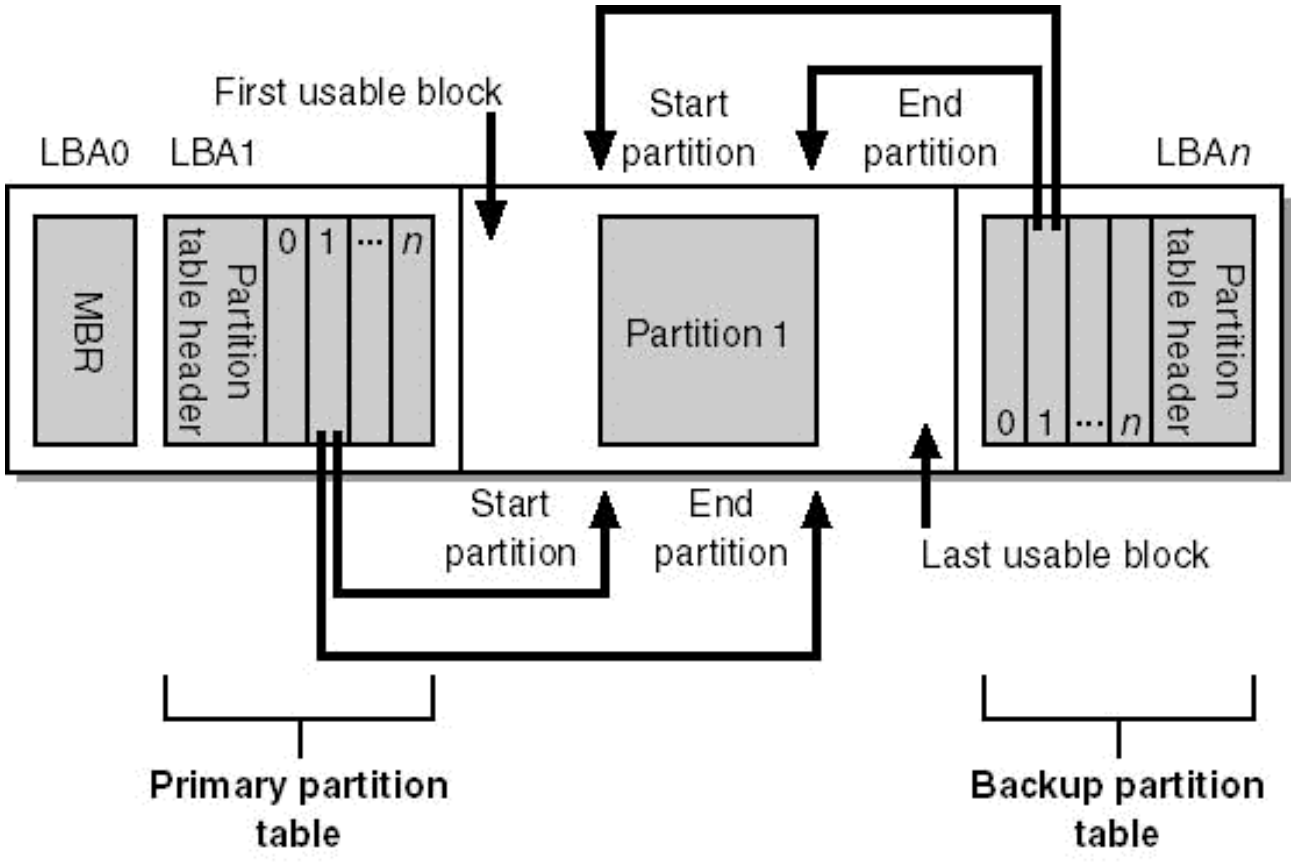
Sample Output (2)

```
---------- LDM Volume Information -----------

Volume   Volume  Mnt  Subdisk   Plex        Physical    Size      Total    Col  Rel      Vol
Name     Type    Nme  Name      Name        Disk        Sectors   Size     Ord  Sectors  State
======   ======  ===  ========  ==========  ==========  =======   =======  ===  =======  ======
Stripe1  Stripe  E    Disk1-01  Stripe1-01  Harddisk0   12288000  4096000  1/3  63       ACTIVE
Stripe1  Stripe  E    Disk2-01  Stripe1-01  Harddisk1   12288000  4096000  2/3  63       ACTIVE
Stripe1  Stripe  E    Disk4-01  Stripe1-01  Harddisk3   12288000  4096000  3/3  63       ACTIVE

Volume1  Simple  F    Disk1-02  Volume1-01  Harddisk0   4096000   4096000  1/1  4096063  ACTIVE

Volume2  Simple  G    Disk4-02  Volume2-01  Harddisk3   4096000   4096000  1/1  4096063  ACTIVE

Volume3  Mirror  H    Disk1-03  Volume3-01  Harddisk0   9350917   9350917  1/1  8192063  SYNC

Volume3  Mirror  H    Disk4-03  Volume3-02  Harddisk3   9350917   9350917  1/1  8192063  SYNC

Raid1    RAID5   I    Disk3-01  Raid1-01    Harddisk2   35084288  17542144  1/3  63      SYNC
Raid1    RAID5   I    Disk5-01  Raid1-01    Harddisk4   35084288  17542144  2/3  63      SYNC
Raid1    RAID5   I    Disk6-01  Raid1-01    Harddisk5   35084288  17542144  3/3  63      SYNC

Volume4  Simple  J    Disk2-02  Volume4-01  Harddisk1   13446917  13446917  1/1
```

GUID Partition Table (GPT)

- Part of Extensible Firmware Interface Specification (EFI)
- EFI targets IA-64 (Ithanium)
- Sector address 64 bit wide
- Uses CRC to ensure integrity of partition table
- Maintains backup copy of partition table
- Assigns each partition GUID

GPT



Note: LBA = Logical Block Address

Storage Drivers

- Follow class/port/miniport architecture
  - Class driver: e.g. disk.sys (common functionality for disks)
  - Port driver: e.g. scsiport.sys
  - Miniport driver: e.g. aha154x.sys
- Class and Port driver mostly provided by Microsoft
- Miniport driver provided by manufacturer

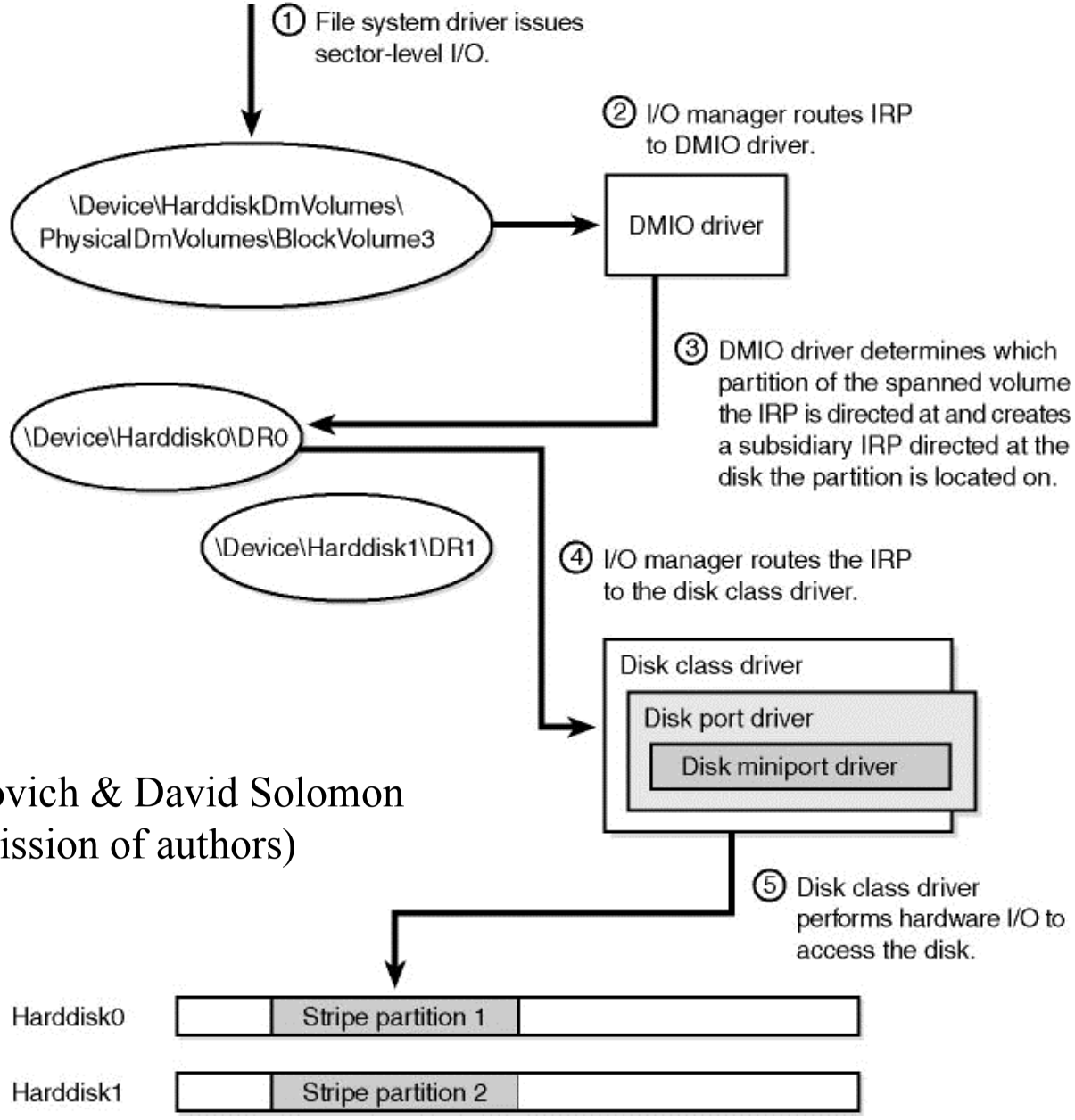
Storage Drivers

- Disk driver creates device objects for disks (\Device\HarddiskX\DRX)
- Calls function *IoReadPartitionTable* to enumerate partitions of disks
- Disk driver creates device objects for partitions (\Device\Harddisk0\DP(1)0x7e00-0x14...+1)
- Disk driver creates symbolic links for legacy drivers (\Device\Harddisk0\Partition0)

Multipartition Volume Management

- Disc I/O for simple partition adds start of volume to volume-relative offset
- For multi-partition volumes „complicated“:
  - Need to check if I/O over multiple volumes (initiate additional IRPs)
  - Calculate which of the volumes has to be used
  - Perform parity checks (RAID-5)

Volume I/O Operation



Volume Namespace

- Volumes are mounted (mountvol.exe)
- Can mount volumes to directories:
  - Directory entry is reparse point (see later)
  - Reparse point redirects I/O to other driver
  - E.g. D:\Test\Test.txt and D:\Test is mounted to CD
    - D: is translated to \??\D:, which links to partition
    - Driver of partition is asked to open „\Test\Test.txt“, which parses until „\Test“ and finds reparse point
    - Driver for reparse point (CD driver) is asked to parse „\Test.txt“

Outline

- Storage Management
  - Partitioning
  - Multi-Disk Volumes
- Cache Manager
- NTFS
  - Structure
  - FS drivers
  - MFT
  - Logging

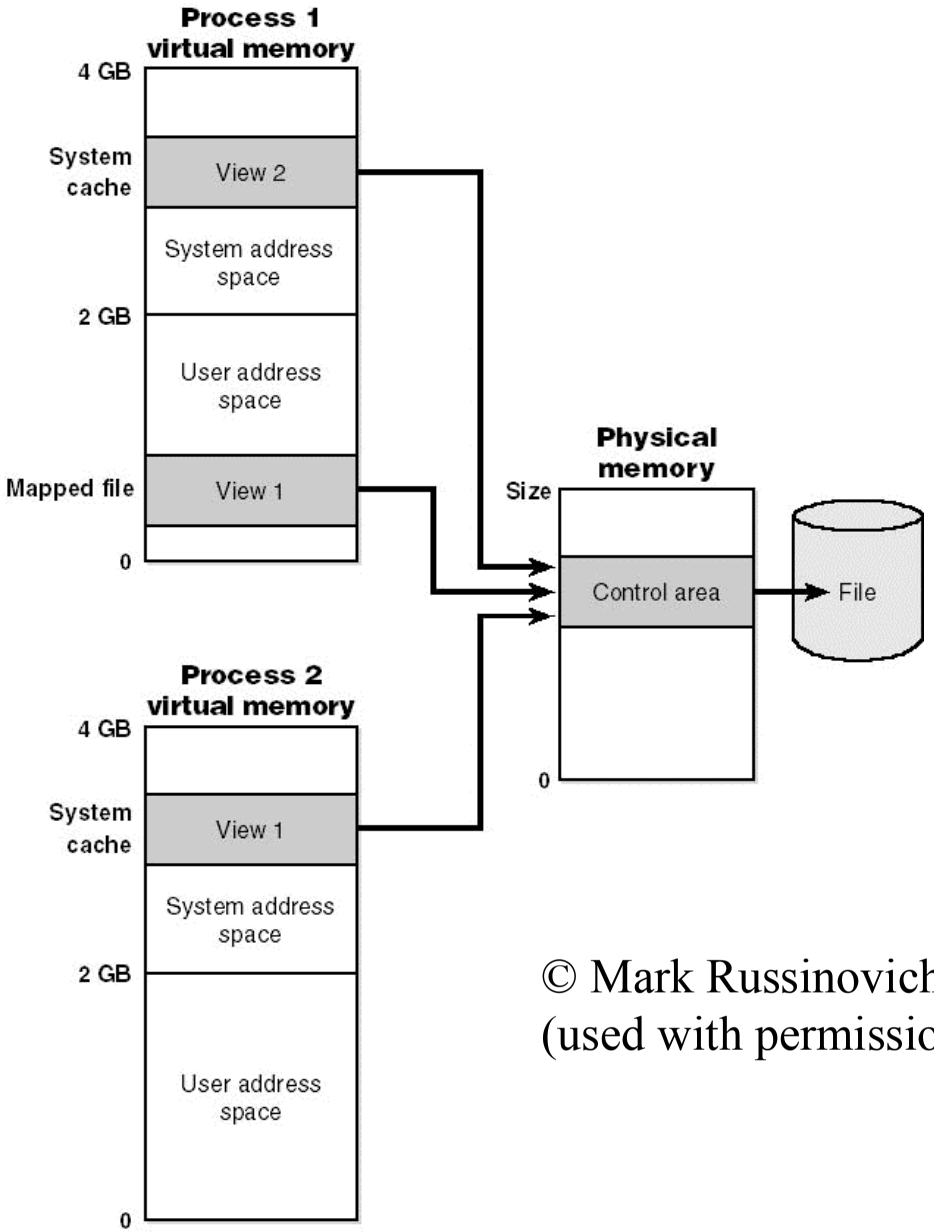
Cache Manager

- Set of kernel-mode functions
- Cooperates with memory manager
- Provides caching for all file system types (local and network)
- Caches on virtual block basis (offset within file)
- Supports „hints“ passed by application at file open time
- Supports recoverable file systems

Cache Manager (2)

- Single, centralized system cache
- Use file mapping object
  - Map view of file into memory
  - Guarantees same data for all open views (guaranteed by memory manager)
- Map 256KB views

Coherent Caching Scheme



Recoverable File System Support

- Changes to FS structure are logged before intended update
- Disk writes are cached → Cache and FS work together:
  1. FS writes log file record of intended update
  2. FS calls Cache to flush this record
  3. FS updates metadata in cache
  4. Cache flushes altered metadata

Recoverable File System Support

- When FS writes data to cache, it can provide LSN (*logical sequence number:* identifies log entry corresponding to change)
- Pages with corresponding log entry are marked „no write“
- When cache manager intends to flush pages:
  - It determines highest LSN
  - Reports this LSN to FS
  - FS instructs Cache to flush log up to LSN
  - *After that*, cache flushes pages

Cache Structure

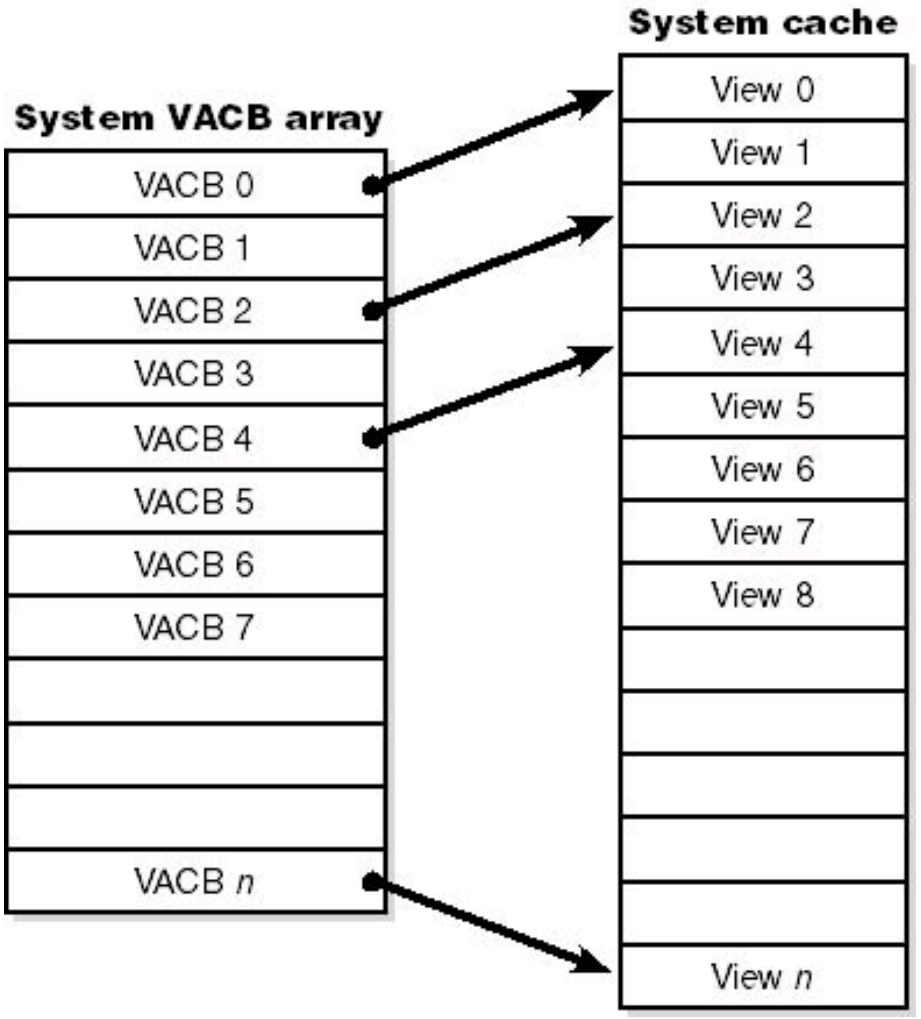
- Cache manager divides system cache memory region in 256KB slots (views)
- At file I/O cache manager maps 256KB aligned and sized region from file into free slot
- Slots are used on round robin basis
- Only „active“ views are mapped into address space
  - View becomes inactive depending on I/O (sequential/ random access)
  - Inactive view is placed at end or front of memory manager‘s standby or modified list

Cache Size

- Cache Virtual Size
  - < 16MB physical memory: cache = 64 MB
  - > 16MB : 128MB + (phys. – 16MB)/4MB * 64MB
    = 128MB + 64MB for every 4MB above 16MB
  - E.g. For 64MB phys. Mem. = 896MB cache
- Cache Physical Size
  - Determined by memory manager‘s working set policy for „system working set“

System Wide Cache Data Structures

- For every view a virtual address control block (VACB)
- VACB array stored in non-paged pool
- VACB contains address of view, cached file, start of view in file, reference count

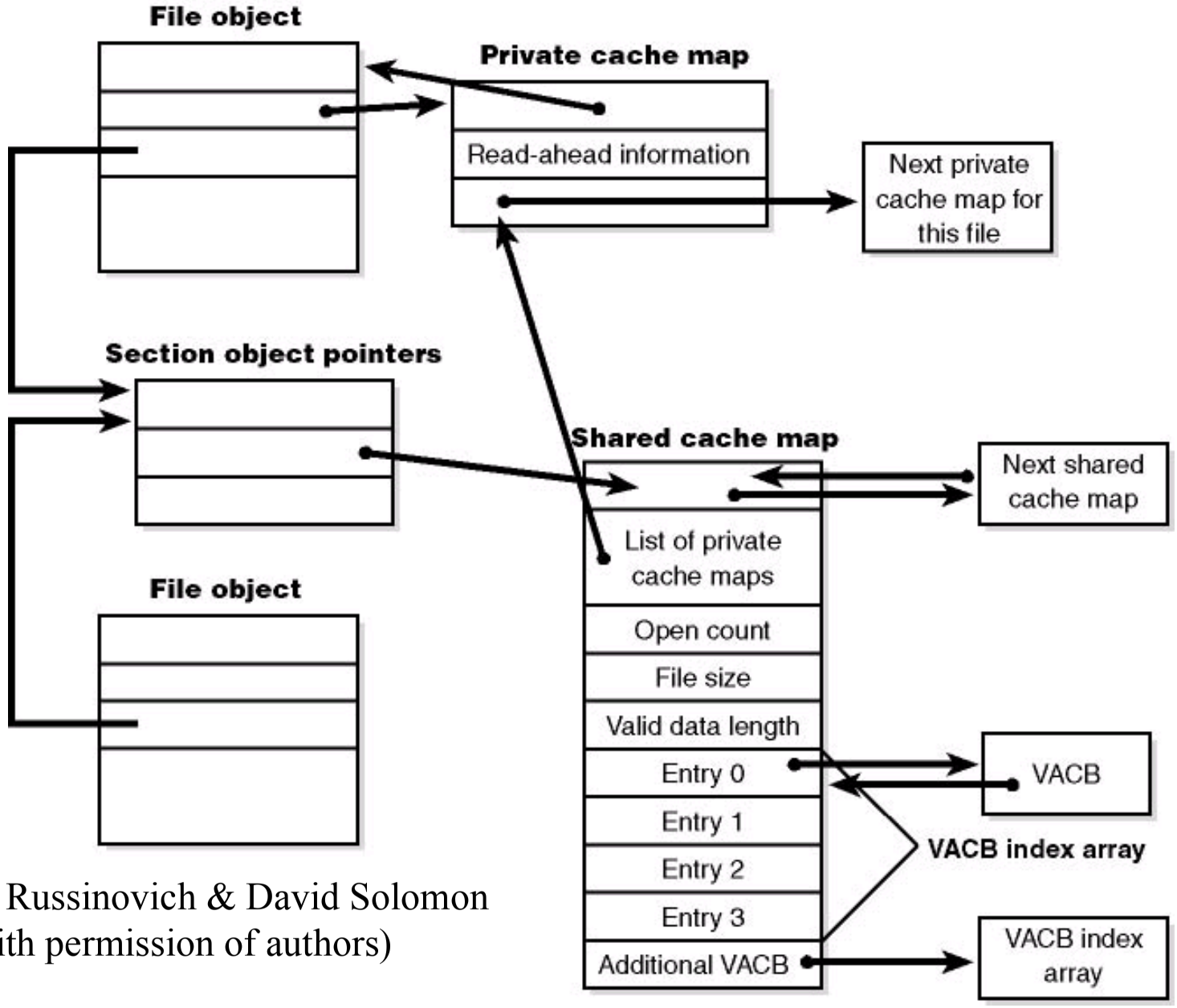


Virtual address of data in system cache
Pointer to shared cache map
File offset
Active count

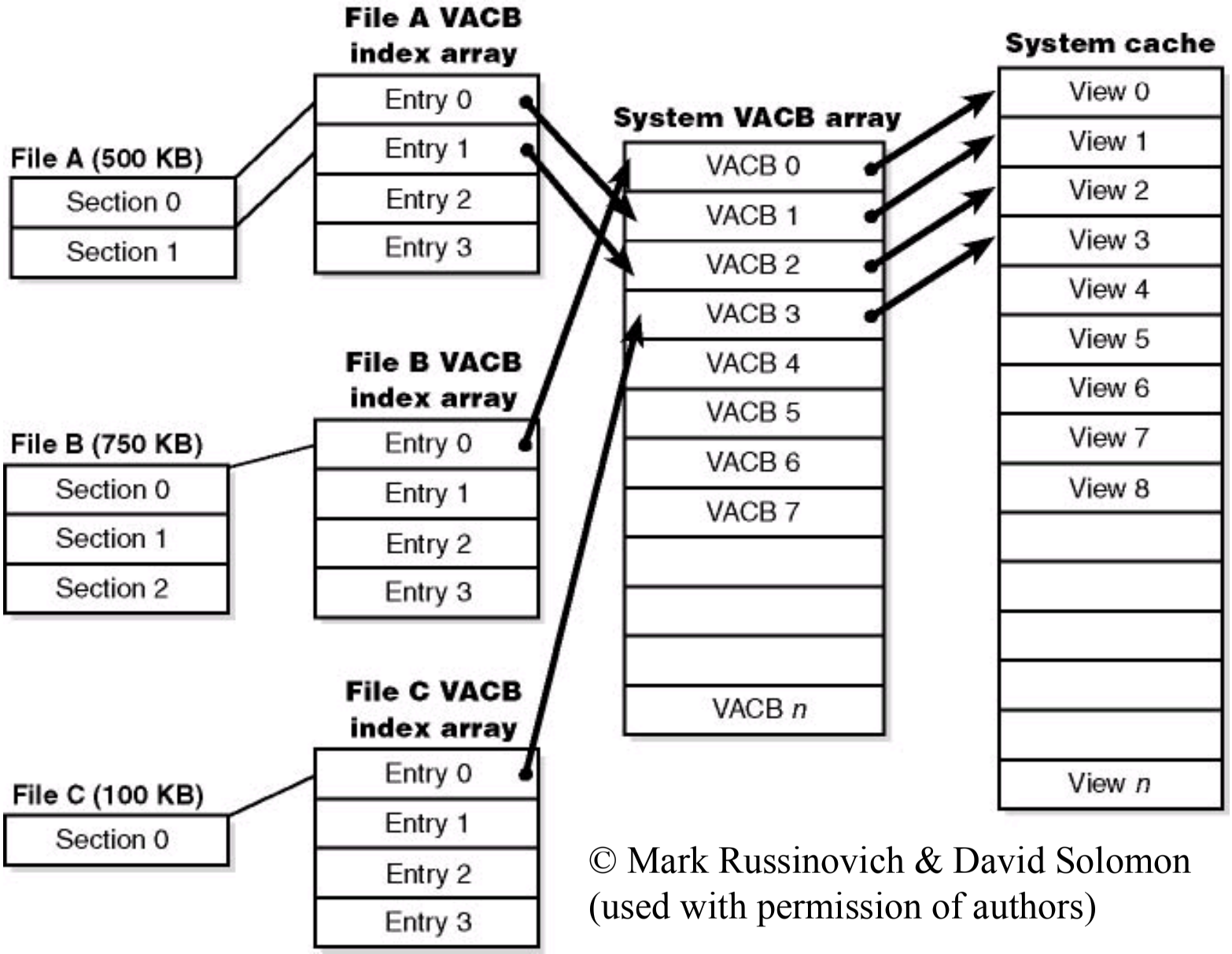
Per-File Cache Data Structures

- Each shared file object has pointer to section object (which describes mapped view of file)
- Section object points to shared cache map
- Shared cache map points to VACB index array, which contains references to VACBs used by file
- Shared cache map contains VACB index array with 4 entries (= 1MB file)
- Additional VACB index arrays contained in tree for larger files

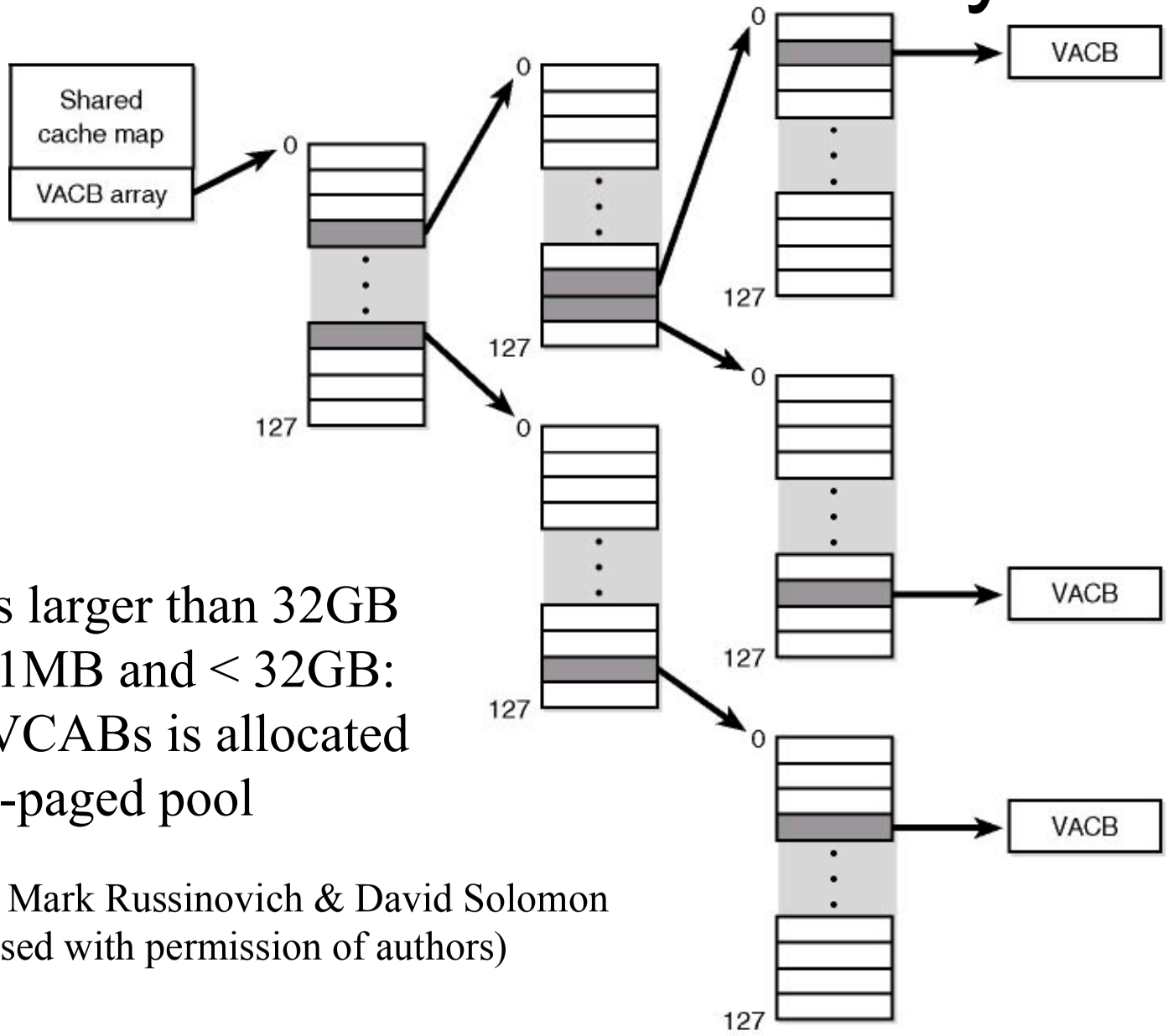
File Cache Data Structures (2)



VACB Index Arrays



Multilevel VACB arrays



- For files larger than 32GB
- Files > 1MB and < 32GB: array of VCABs is allocated from non-paged pool

Cache Operation

- Writes into cache are buffered
- Lazy writer:
  - Wakes once per second
  - Writes one eighth of dirty pages
  - Wakes only if dirty page threshold (~3/8 phys. mem.) has been reached
  - If more dirty pages are generated than written, the number of written pages is adapted
  - Can be disabled (FILE_ATTRIBUTE_TEMPORARY) except if memory shortage
- Writes can go through cache (FILE_FLAG_WRITE_THROUGH)
- Buffer can be flushed explicitly

Cache Operation (2)

- Read-Ahead
  - Cache stores last two read addresses and calculates next address for read-ahead
  - If sequential file access specified: no history, but sequential read-ahead (if read is past a view, this view is freed)
  - For random access files: no read-ahead
- When application reads:
  - Read is satisfied from cache and next I/O is initiated
  - Background thread reads while app. executes

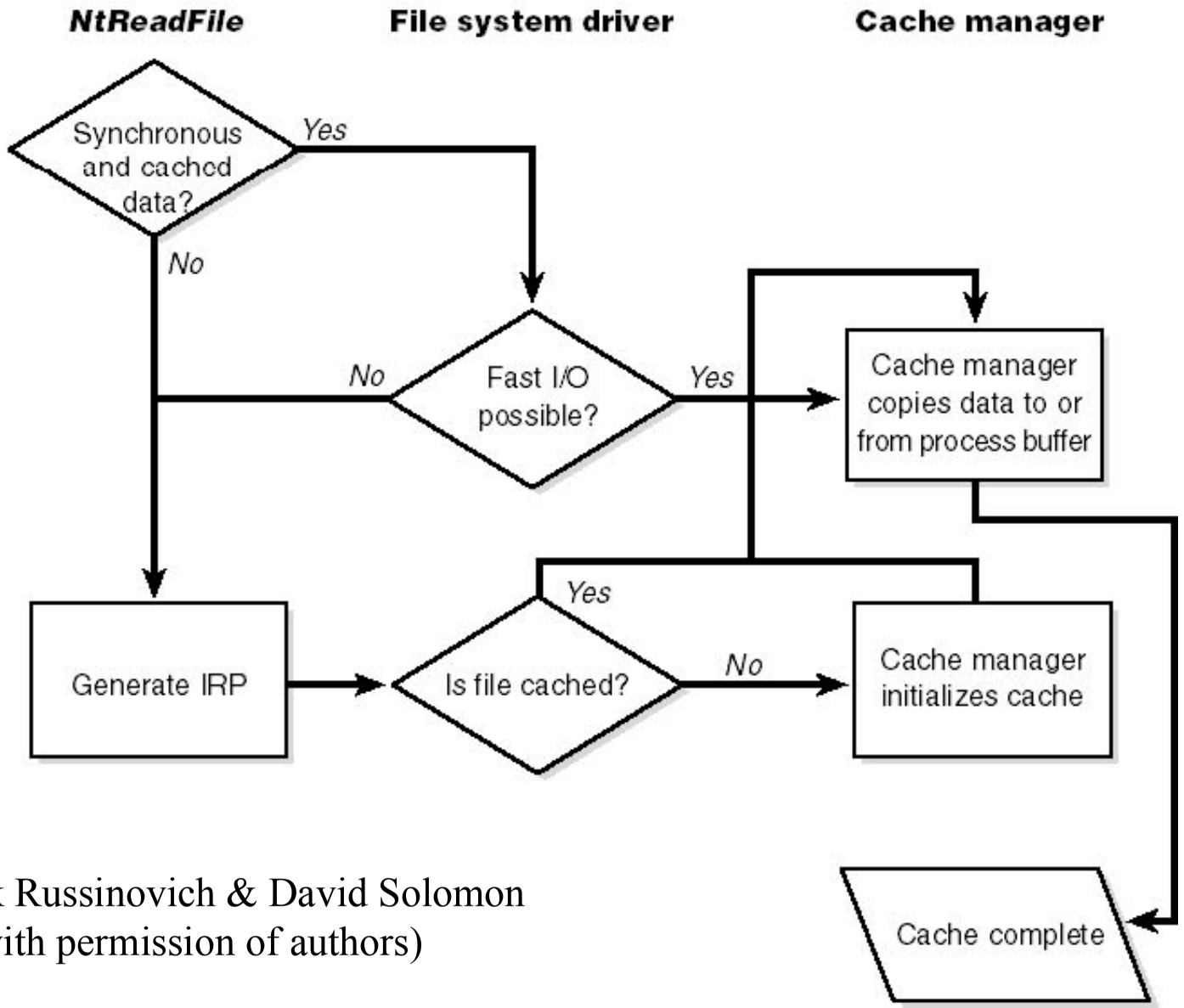
System Threads

- Get work from worker queue
- Cache manager organizes work in two lists:
  - *express queue* for read-ahead
  - *regular queue* for lazy-write scans, write behinds, and lazy closes
  - items in per-processor look-aside list
  - number of items depends on system size

Fast I/O

- I/O manager calls file system driver‘s fast I/O to check whether cache can satisfy request
- No need to set up IRP
- If page is in cache, FS driver can read from memory
- Sometimes not fast I/O even if page is in memory
  - File is locked,
  - Asynchronous I/O, ...

Fast I/O



Fast I/O

- After copy:
  - For reads: read-ahead information is updated
  - For writes: dirty bits of modified pages are set so lazy writer will flush page
  - For write-through: modifications are flushed to disk
- Note: Cache manager copies to from virtual page → relies on memory manager to map page from file

Cache Support Routines

- Copy data to and from Cache to user space buffers
  - Two read version for cached and non-cached read provided by file system driver
- Access (meta-)data directly in the cache (for file system drivers) – data has to be present in physical memory (has to be pinned ≡ not flushed)
- Obtain Memory Description List (MDL) for DMA (describes physical address)
- Write Throttling:
  - Restrict number of writes if this would hurt performance
  - First free physical pages if required by flushing dirty pages