Operating System

lina liu Apr. 13

1. how ls cmd process executed?
2. what is process?
3. what is virtual memory?
4. what the difference between memory cache and virtual memory?
5. what is CPU protect mode?
6. difference between BIOS and UEFI under CPU protect mode?

3 pieces operating system

Virtualization

- The Abstraction: The process
- Interlude: Process API
- Mechanism: Limited Direct Execution
- Scheduling: Introduction
- Scheduling: The multi-level feedback queue
- Scheduling: Introduction
- Mechanism: Limited Direct Execution
- Scheduling: Proportional Share
- Multiprocessor Scheduling
- CPU Virtualization
- Memory Virtualization
- The abstraction: Address Space
- Interlude: Memory API
- Mechanism: Address Translation
- Segmentaion
- Free-Space Management
- Paging: Introduction
- Paging: Faster Translation(TLBs)/Smaller tables
- Beyond Physical Memory: Mechanisms/Policies
- Complete Virtual Memory Systems
- Memory Virtualization Summary

Concurrency

- Concurrency: An Introduction
- Interlude: Thread API
- Locks
- Lock-based concurrent data structure
- Condition Variables
- Semaphores
- Common Concurrency Problems
- Event-based concurrency(Advanced)
- Summary of Concurrency

Persistence

- A Dialogue on Persistence
- I/O Devices
- Hard Disk Drives
- Redundant Arrays of Inexpensive Disks(RAIDs)
- Interlude: Files and Directories
- File System Implementation
- Locality and the fast file system
- Crash Consistency: FSCK and Journaling
- Log-structured file systems
- Flash-based SSDs
- Data Integrity and Protection
- Summary Dialogue on Persistence
- A Dialogue on Distribution
- Distributed Systems
- Sun’s Network File System(NFS)
- The Andrew File System(AFS)
- Summary Dialogue on Distribution

CPU Virtualization

Process

process control

process tools

process state

process create

API

System Call

exec()

wait()

fork()

Mechanism

Problem

swtiching between process

restricted operations

technique

limited direct execution

Scheduling

Multiprocessor

Architecture

Synchronization

Cache Affinity

Single-Queue Scheduling

Multi-Queue Scheduling

Linux Multi-Processor Scheduling

Proportional Share

Linux Completely Fair Scheduler

Tickets

mechaism

implementation

Example

how to assgin tickets

why not determinstic

Multi-Level Feedback Queue

basic rules

Tuning MLFQ issue

Attempt

how to change priority

the priority boost

better accounting

introduction

workload

Metrics, repsonse time

Incorporating I/O

FIFO

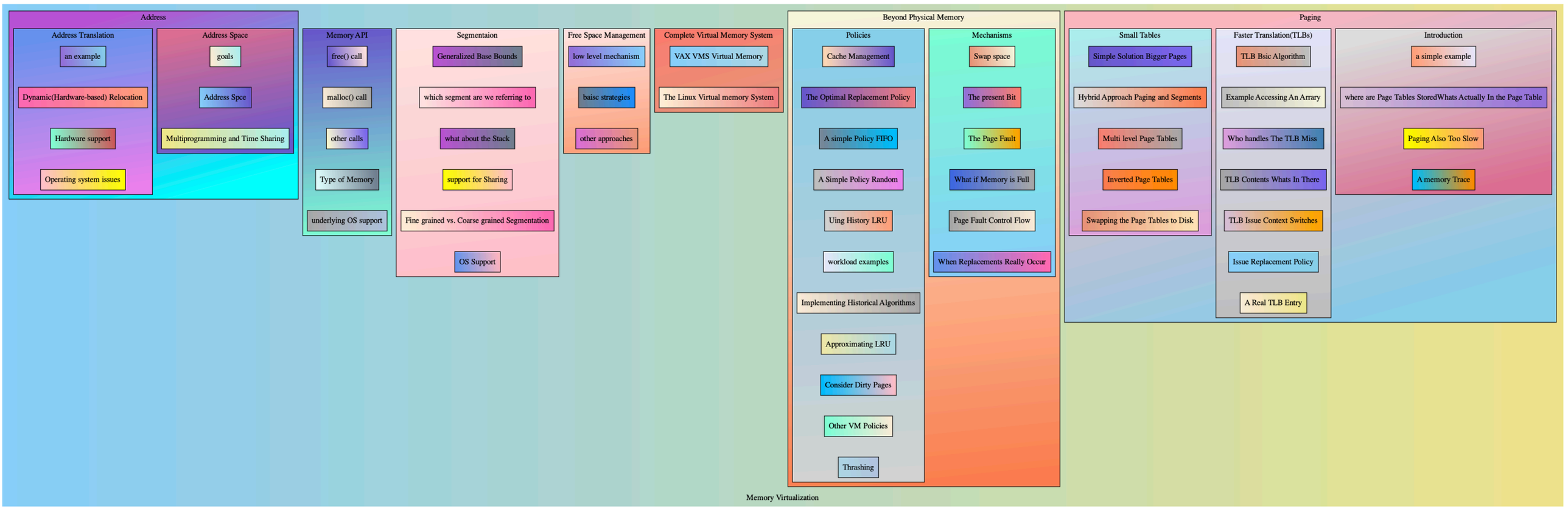
Shorted Job First

Shortest Time-to-Completion First

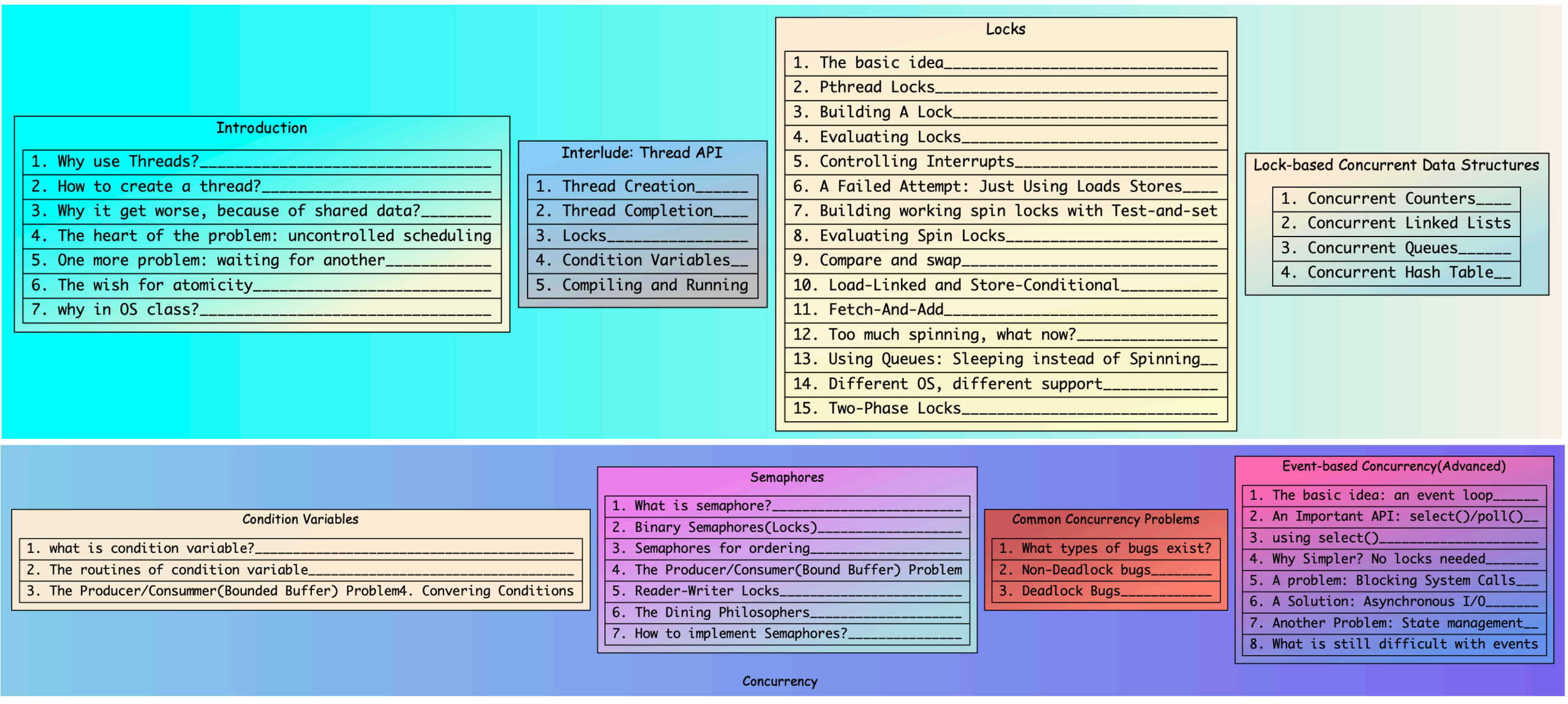
Round Robin

Nor more Oracle

Memory Virtualization



Concurrency



Persistence

I/O Devices

1. System Architecture
2. A Canonical Device
3. THe Canonical Protocol
4. Lowering CPU Overhead with the Interrupts
5. More Efficient Data Movement with DMA
6. Methods of Device Interaction
7. Fitting into the OS: The Device Driver
8. Case Study: A simple IDE Disk Driver

Hard Disk Drives

1. The Interface
2. Basic Geometry
3. A Simple Disk Drive
4. I/O Time, Doing the math
5. Disk Scheduling

Redundant Arrays of Inexpensive Disks(RAIDs)

1. Interface And RAID Internals
2. Fault Model
3. How to Evaluate A RAID
4. RAID Level 0: Striping
5. RAID Level 1: Mirroring
6. RAID Level 4: Saving Space with Parity
7. RAID Level 5: Rotating Parity
8. RAID Comparison: A Summary
9. Other Interesting RAID Issues

Interlude: Files and Directories

1. Files and Directories
2. The File System Interface
3. Creating Files
4. Reading and Writing Files
5. Reading and Writing, but not Sequentially
6. Shared File Table Entries: fork() and dup()
7. Writing Immediately with fsync()
8. Renaming Files
9. Getting Information about Files
10. Removing Files
11. Making/Reading/Deleting/ Directories
12. Hard Links
13. Symbolic Links
14. Permission Bits and Access Control Lists
15. Making and Mounting a file system

File System Implementation

1. The way to think
2. Overall Organization
3. File Organization: The Inode
4. Directory Organization
5. Free Space Management
6. Access Paths: Reading and Writing
7. Caching and Buffering

Locality and The fast file system

1. The Problem: Poor Performance
2. FFS: Disk Awareness is the solution
3. Organizing Structure: The Cylinder Group
4. Policies: How to Allocate Files and Directories5. Measuring File Locality
6. The Large-File Exception
7. A Few Other things about FFS

Crash Consistency: FSCK and Journaling

1. A Detailed Example
2. Solution #1: The File System Checker
3. Solution #2: Journaling(or write-Ahead Logging)
4. Solution #3: Other Approaches

Log-structed File Systems

1. Writing to Disk Sequentially
2. Writing Sequentially And Effectively
3. How much to Buffer?
4. Problem: Finding Inodes
5. Solution Through Indirection: The Inode Map
6. Completing The Solution: The Checkpoint Region
7. Reading a file from Disk: A recap
8. What about directories?
9. A new problem: Garbage Collection
10. Determing Block Liveness
11. A Policy Question: Which Blocks to clean, and when?
12. Crash Recovery And the log

Flash-based SSDs

1. Storing a single bit
2. From bits to banks/planes
3. Basic Flash Operations
4. Flash Performance And Reliability
5. From Raw Flash to Flash-Based SSDs
6. FTL Organization: A bad Approach
7. A log-structured FTL
8. Garbage Collection
9. Mapping Table Size
10. Wear Leveling
11. SSD Performance and cost

Data Integrity and Protection

1. Disk Failure Modes
2. Handling Latent Sector Errors
3. Detecting Corruption: The Checksum
4. Using Checksums
5. A new Problem: Misdirected Writes
6. One last Problem: Lost Writes
7. Scrubbing
8. Overheads of Checksumming