

# Final Review

Yizhou Wang

<wangyzh2024>

# Topics Covered

- Parallelism & Amdahl's Law (L19)
- Thread-Level Parallelism (L20-L22)
- Advanced Cache & Coherence (L23)
- Operating Systems & I/O (L24)
- Virtual Memory (L25)
- DMA, Networking, WSC (L26)
- Fault-Tolerance & Security (L27)
- Heterogeneous Computing & Summary (L28)

# Parallelism & Amdahl's Law

# Parallelism: Motivation & Amdahl's Law

- CPU Trends:
  - Moore's Law, Power Wall, Frequency Plateau, More Cores
  - Need for Parallelism.
- Amdahl's Law:  $\text{Speedup} = 1 / [(1 - F) + (F / S)]$ 
  - F: Parallelizable fraction.
  - S: Speedup of parallel part.
  - Sequential part (1-F) limits overall speedup.

# Scaling & Flynn's Taxonomy

- Strong vs. Weak scaling
- Load Balancing
- Flynn's Taxonomy: SISD/SIMD/MISD/MIMD
- SPMD: Single Program Multiple Data

# DLP & SIMD Execution

- DLP: Same operation on multiple data items (vector ops).
- SIMD Architecture: Wide registers, specialized FUs. One instruction -> multiple ops.
- Applications: Scientific, Graphics, AI/ML.
- SIMD Instructions: Operate on "packed" data. (e.g., VADDPS). Intel SSE/AVX (XMM/YMM/ZMM regs), Intrinsics.
- Loop Unrolling: Increase DLP by reducing overhead, exposing independent ops. (Manual/Compiler).

TLP & OpenMP (L20-L22)

# TLP Intro & OpenMP Basics

- Threads: Own regs/PC/SP, Share memory (heap, globals). HW/SW threads.
- Fork-Join Model: Main forks -> parallel tasks -> threads join.
- OpenMP: API for shared-memory TLP. `#pragma omp ...`
- Key Concepts:
  - Shared/Private vars.
  - `omp_get_thread_num()`
  - `omp_get_num_threads()`



# TLP: Synchronization & Locks

- Need for Sync: Correctness, avoid data races
- Critical Sections: Code executed by one thread at a time
  - OpenMP: `#pragma omp critical/barrier`
- Locks (Mutexes): Exclusive access. Operations: Acquire, Release
  - Analogy: Buying Milk (L21)
- Naïve Locks: Prone to races. Need atomic hardware ops.

# Hardware Atomics for Synchronization

- Atomic Operations: Indivisible hardware-supported read-modify-write.
- RISC-V Atomics (RV-A Ext.):
  - Load-Reserved (lr) / Store-Conditional (sc)
    - Builds atomic sequences (swap, test-and-set).
  - Atomic Memory Operations (AMOs): Single instructions (`amoswap.w`, `amoadd.w`).
    - Used for lock implementation (`amoswap.w.aq/.rl`).
- Note: Use library locks (`pthread`s, `std::mutex`, OpenMP critical).

# Advanced OpenMP & HW Multithreading

- Work-Sharing: `#pragma omp sections/single/master`
- OpenMP Reduction: `reduction(operator:list)` for safe parallel aggregation (e.g., sum).
- Multicore: Multiple independent cores. True parallelism.
- Hardware Multithreading (SMT/Hyper-Threading): Single core, multiple HW threads (duplicate state, shared FUs). Improves core utilization.
- Modern: Multicore + SMT.

# Advanced Cache & Coherence (L23)

# Multiprocessor Caches & Coherence Problem

- Shared Memory MPs: Multiple CPUs, single physical address space. Private Caches (L1/L2).
- Cache Coherence Problem: Multiple copies of a block; one CPU writes, others become stale. Example (L23, S8-10).
- Coherence: Read returns most recent write.
- Snooping Protocols: Controllers monitor shared bus.
  - Write-Invalidate (Common): Writer broadcasts invalidate; others invalidate.
  - Write-Update: Writer broadcasts new data; others update. (More bus traffic).

# Cache Coherence States & False Sharing

- Coherence States (e.g., MSI, MESI, MOESI): Track block state (Modified, Shared, Invalid, Exclusive, Owned). State transitions on CPU/Bus ops.
- False Sharing: Different data items in same cache block, accessed by different CPUs.
  - Write to one item invalidates block for others -> unnecessary misses.
  - Mitigation: Alignment, padding.

# Cache Inclusiveness (Multi-Level)

- Inclusive: in L1  $\Rightarrow$  in L2
- Exclusive: in L1  $\Rightarrow$  not in L2
- Non-inclusive: No strict relation
- Trade-offs: Coherence ease vs. capacity/performance

# Operating Systems & I/O (L24)



# OS Role & I/O Basics

- OS: Manages HW/SW resources, provides services (Booting, Resource/Process/I/O Mgmt, File Systems, Protection).
- I/O Interaction: CPU controls devices, transfers data.
- Memory-Mapped I/O (MMIO): (RISC-V) Device regs in physical addr space. Use lw/sw.
- Speed Mismatch: CPU vs. Devices -> Synchronization needed.
- Polling: CPU busy-waits checking device status. Simple, but wasteful.
- Interrupts: Device signals CPU. Efficient for slow/infrequent I/O.

# Interrupts, Exceptions, Traps & Syscalls

- Interrupt: Asynchronous, external (I/O, timer).
- Exception: Synchronous, from instruction (fault, syscall).
- Trap: HW mechanism (jump) to OS handler.
  - Handling: Save state (SEPC, SCAUSE), jump to handler (STVEC), OS runs, restore, resume.
- Precise Traps: Handler sees consistent state. Crucial. Pipeline handling at commit point.
- System Calls (Syscalls): User programs request OS services via ECALL trap.

# Virtual Memory (L25)

# Virtual Memory: Motivations & Addressing

- VM Motivations: Illusion of large, private memory; Protection/Isolation; Efficient RAM use; Disk as backing store.
- VA (Virtual Address): CPU/program generated.
- PA (Physical Address): HW memory.
- Address Translation (VA- $\rightarrow$ PA): OS managed, HW assisted (MMU).
- Paged Memory: VA/PA in fixed-size pages (e.g., 4KB).
  - $VA = VPN + \text{Offset}$ .  $PA = PPN + \text{Offset}$ . (Offset same).

# Page Tables & Translation Process

- Page Table (PT): Maps VPNs -> PPNs for a process. Array of PTEs. PTBR (satp) points to active PT.
- PTE Contents: PPN, Valid, Protection (R/W/X), Dirty, Accessed bits, Disk address.
- Translation & Page Faults:
  - CPU generates VA. MMU gets VPN. Use PTBR+VPN to find PTE in memory.
  - Check validity and protection. If okay, access PPN+Offset.
  - Otherwise trap to OS. OS decides to load or to kill.
- Demand Paging: Load pages only on fault.

# TLB & Hierarchical Page Tables

- TLB (Translation Lookaside Buffer): Small, fast HW cache for recent VPN->PPN translations (PTEs).
  - TLB Hit: Fast translation.
  - TLB Miss: PTW (HW/OS). Load PTE into TLB.
  - Context Switch: Flush TLB or use ASIDs.
- Hierarchical PTs: Multi-level PTs for large VAs (saves space if sparse).
  - Pros: Space saving. Cons: More PTW accesses on TLB miss.

Miscellaneous

# DMA & Storage (SSDs) (L26)

- DMA (Direct Memory Access): Device transfers data directly to/from memory. DMAC manages. CPU programs DMAC, gets interrupt on completion. Frees CPU.
  - Issues: Cache Coherence, Bus Arbitration.
- SSDs (Solid State Drives): Flash Memory. Non-volatile. Faster, durable, lower power than HDDs. Wear Leveling.



# Networking: Basics & TCP/IP (L26)

- Networking: Inter-computer communication. Nodes, Links, Packets (header, payload).
- Layering & Protocols: Hierarchy (e.g., TCP/IP). Encapsulation.
- TCP/IP Suite:
  - App: HTTP, SMTP.
  - Transport: TCP (reliable), UDP (unreliable).
  - Network: IP (addressing, routing).
  - Link: Ethernet, Wi-Fi (MAC addr).
  - Physical: Bits.

# Warehouse-Scale Computers (WSCs) (L26)

- WSCs (Datacenters): 10k-100k+ servers. For large Internet services.
- Characteristics: Ample Parallelism (RLP, DLP), Scale (failure tolerance needed), Operational Costs (power).
- Design Goals: Energy Efficiency (PUE), Energy-Proportional Computing.
- MapReduce Model: For parallel processing of large datasets.
  - Flow: Input -> Map -> Shuffle/Sort -> Reduce.
  - Frameworks: Hadoop, Spark.

# Dependability via Redundancy (L27)

- Spatial (extra HW/info). Temporal (retry).
- Metrics: MTTF, MTTR, Availability, AFR
- Error Detection/Correction:
  - Hamming Distance
  - Parity Bit(s)
  - Hamming ECC
- RAID: RAID0, RAID1, RAID5

# Security & Side-Channel Attacks (L27)

- Heartbleed: Software Flaw (missing length check)
- Rowhammer: Hardware Flaw (DRAM disturbance)
- Side-Channel Attacks: Timing/Power/...
  - E.g. Cachebleed, Flush & Reload
  - Meltdown: Out-of-Order Execution
  - Spectre: Speculative Execution

# Heterogeneous Computing (L28)

- Heterogeneous: More than one kind of processor/core
  - GPU, DSP, NPU, System-on-Chip, ASIC/FPGA accelerator, etc.
- Field-Programmable Gate Array
  - Logic & interconnection is reprogrammable
  - HDL => Implementation => HW Logic

# The Great Ideas (L28)

- Abstraction to simplify design
- Design for Moore's Law
- Make the common case fast
- Memory hierarchy
- Performance via parallelism/pipelining/prediction
- Performance measurement & improvement
- Dependability via redundancy