EEE4120F

High Performance Embedded Systems

Term 1

Summary

Lecturer:
Simon Winberg

Summary for Class Test 1 Preparation

- Preparation for Test on 16 April 2026
- Focus:
  Part A - Microprocessor-based parallel computing and theory
- Lectures 1 to 12b (Term 1 lectures)

The Landscape of Parallel Computing (Reading 1)

- Based on Asanovic et al. (The Berkeley View)
- Important to read this paper, see guides on R1 to focus your reading
- Key concept: "DWARFS" as algorithmic patterns
- Focus is on structural patterns rather than specific implementations
- Understanding the shift toward many-core and heterogeneous systems

Core Terminology: The Foundations

- Verification: Ensuring the system is built correctly to what was agreed
- Validation: Ensuring the right system is being built
- Golden Measure: A trusted, accurate, usually sequential code solution
- Golden measure used as the benchmark for correctness

Performance Benchmarking Essentials

- Main objective: Improve speed, power efficiency, and resource use
- Wall-clock time: Measuring real-world execution time
- ACPI: Average Cycles Per Instruction calculations
- Benchmarking metrics: Cycle count, memory usage, and power consumption

Speedup and Amdahl's Law

- Speedup formula:

$$Speedup = T_{p1}/T_{p2}$$

- Amdahl's Law:

$$Speedup_{parallel} = \frac{1}{(1-f)+\frac{f}{n}} \; f$$

- *f* = Fraction of computation that can be parallelized
- *n* = Number of processors
- Best practice: Run multiple times and discard the first result (cold start)
- BUT for real-time above can give false result

Flynn's Taxonomy of Architectures

- Used to classify parallel computer architectures …
- SISD: Single Instruction, Single Data
- SIMD: Single Instruction, Multiple Data
- MISD: Multiple Instructions, Single Data
- MIMD: Multiple Instructions, Multiple Data

Base Core Equivalents (BCE)

- Abstracted hardware model to complement Amdahl's software model
- Bound of N resources per chip for cores
- Factors: Area, power, and cost constraints
- Used to compare resource use across different multicore layouts

Designing Parallel Programs (Steps 1-4)

- 1. Understand the problem: Identify sequential vs parallel parts
- 2. Partitioning: Separating the problem into main tasks
- 3. Granularity: Determining the size of tasks
- 4. Communications: Mapping how data is shared between tasks

Hint: You want to remember the sequencing of the steps too.

Designing Parallel Programs (Steps 5-8)

- 5. Identify data dependencies: Where tasks rely on others
- 6. Synchronization: Coordinating task execution
- 7. Load balancing: Ensuring processors remain equally busy
- 8. Performance analysis and tuning: Iterative optimization

Memory Partitioning and Infrastructure

- Shared Memory: Global address space accessible by all
- Distributed Memory: Local memory per node; needs explicit passing
- Contiguous: Data stored in a single, continuous block
- Interleaved/Interlaced: Data striping across memory modules
- Stride: Size of blocks alternated in data striping

Parallel Programming Frameworks

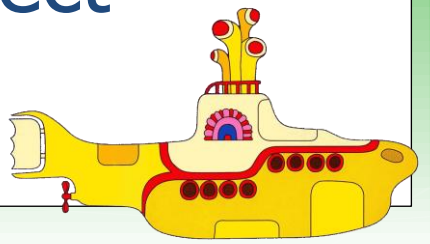
- (A shift from pthreads, comms and shared memory to MPI and OpenMP for 2026)
- Software Framework: An abstraction where generic functionality is customized by user code
- MPI: Message Passing Interface for distributed systems
- OpenMP: Standard for shared-memory multiprocessing
- Focus on SPMD (Single Program Multiple Data) patterns *

* SPMD a classic term from early days when things like SIMD had not yet been thought of.

Study Tips for Class Test 1

- Review the "Unserious Quiz" in Lecture 1 for MCQ practice
- Be prepared for GA2 Validation Essay Questions on design methods*
- Practice ACPI and Amdahl's Law calculations
- Focus on the 8 steps of parallel design as they relate to the course project

*Something like a yellow submarine might appear which you should rather not singalong to, but knowing things like pressure and the bends.



Practice Class Test: Invitation

- There's a MCQ Practice Test available
- The actual test will have a bunch of MCQ and some short written answers