Meeting Note 1

Date: 2025-10-01 **Time:** 14:00 - 14:45 **Location:** Supervisor's Office (PQ706) **Attendees:** LIN Zhanzhi, CAO Yixin (Supervisor)

1. Objectives

- Review the initial draft of the Capstone Project Proposal.
- Discuss the feasibility and scope of implementing Dual-Pivot Quicksort in C++.
- Identify key academic and technical requirements for the project.

2. Progress Report

- Selected project topic: "Implementing Dual Pivot Quicksort in C++".
- Drafted a preliminary proposal outlining the basic implementation plan.
- Conducted initial reading on Yaroslavskiy's 2009 paper.

3. Discussion & Feedback

Key Discussion Points

- **Scope Definition:** The initial proposal focused primarily on the coding aspect. We discussed the need to elevate this to an academic research project by adding rigorous performance analysis and theoretical backing.
- **Language Standards:** Discussed the choice of C++23. Supervisor agreed it is a good choice to demonstrate modern software engineering practices (e.g., using `std::ranges` and concepts).
- **Benchmarking:** Simply implementing the algorithm is not enough. The core value lies in comparing it against existing giants like `std::sort` (Introsort) and PDQSort.

Supervisor Feedback

> "The current draft is too engineering-focused. You need to justify *why* this is a research gap. Java adopted Dual-Pivot years ago; why hasn't C++? You should investigate the 'scanned elements model' by Wild to explain cache behavior, rather than just counting comparisons."

> "Expand the methodology to include a systematic comparative analysis. Don't just measure time; measure cache misses and branch mispredictions."

4. Issues & Challenges

- **Theoretical Depth:** I admitted I was unfamiliar with the mathematical analysis of Dual-Pivot Quicksort (e.g., Aumüller's framework).
  - *Resolution:* Supervisor provided references to key papers by Aumüller and Dietzfelbinger to strengthen the literature review.
- **Benchmarking Fairness:** How to ensure a fair comparison between my implementation and the highly optimized STL?

- *Resolution:* We agreed to use a standardized benchmarking framework (like Google Benchmark) and test across various distributions (random, sorted, reverse).

5. Action Items

- [ ] Read Aumüller and Dietzfelbinger's paper on Dual-Pivot analysis. (Due: 2025-10-10)
- [ ] Read Wild's paper on the "scanned elements model". (Due: 2025-10-10)
- [ ] Rewrite the "Background" section to highlight the gap in C++ standard libraries. (Due: 2025-10-15)
- [ ] Expand "Methodology" to include specific metrics (L1/L2 cache misses, branch prediction). (Due: 2025-10-15)
- [ ] Submit the revised, final proposal. (Due: 2025-10-24)

6. Next Meeting Plan

- **Target Date:** 2025-11-05
- **Focus:** Review of the revised proposal and initial code structure.

Meeting Note 2

Date: 2025-11-05 **Time:** 14:00 - 14:30 **Location:** Supervisor's Office (PQ706) **Attendees:** LIN Zhanzhi, CAO Yixin (Supervisor)

1. Objectives

- Review progress on the initial C++ implementation of Dual-Pivot Quicksort.
- Discuss findings from the literature review (Wild and Aumüller papers).
- Plan the structure for the upcoming Interim Report.

2. Progress Report

- **Proposal:** Final project proposal submitted on Oct 24.
- **Literature Review:** Completed reading of "Dual-Pivot Quicksort: Optimality and Analysis" (Aumüller) and Wild's work on scanned elements.
- **Implementation:** Created a basic working prototype of 1-pivot and 2-pivot quicksort in C++.
    - Current status: Works for `std::vector<int>`.

3. Discussion & Feedback

Key Discussion Points

- **Code Quality:** Showed the initial code. Supervisor noted that while it works for `int`, it needs to be generic.
- **C++23 Concepts:** Discussed using `std::sortable` and `std::random_access_iterator` concepts to ensure type safety.
- **Interim Report:** Supervisor outlined the expectations for the Interim Report.

Supervisor Feedback

> "Your current implementation is too specific to integers. You must use C++ templates and Concepts to make it a true library candidate. Ensure it handles custom comparators correctly."

> "For the Interim Report, don't just paste code. Focus on the *design decisions*. Why did you choose this specific partitioning scheme? Use diagrams to explain the 3-way partition."

4. Issues & Challenges

- **Generic Implementation:** I am struggling slightly with the syntax for C++20/23 Concepts for custom iterators.
    - *Resolution:* Supervisor suggested looking at the `std::ranges::sort` implementation in the GCC or LLVM open-source repositories for reference.
- **Benchmarking Setup:** Setting up Google Benchmark is proving tricky with the custom build system.
    - *Resolution:* Advised to stick to a simple `CMake` setup to manage dependencies.

5. Action Items

- [ ] Refactor code to use C++ Templates and Concepts (Generic programming). (Due: 2025-11-20)
- [ ] Implement a "Benchmark Runner" using Google Benchmark. (Due: 2025-11-25)
- [ ] Draft the "Methodology" and "Literature Review" sections for the Interim Report. (Due: 2025-12-01)

6. Next Meeting Plan

- **Target Date:** 2025-12-03
- **Focus:** Review of Interim Report Draft and preliminary benchmark results.

Meeting Note 3

Date: 2025-12-03 **Time:** 14:00 - 14:45 **Location:** Supervisor's Office (PQ706) **Attendees:** LIN Zhanzhi, CAO Yixin (Supervisor)

1. Objectives

- Review the draft of the Interim Report.
- Analyze preliminary benchmarking results.
- Discuss requirements for the Interim Presentation Video.

2. Progress Report

- **Interim Report:** Completed 80% of the report (Introduction, Literature Review, Methodology).
- **Implementation:** Refactored code to be fully generic using C++20 Concepts. Added `BenchmarkRunner` class.
- **Results:** Initial benchmarks show DPQS is faster than `std::sort` for large arrays (>1M elements) of random integers.

3. Discussion & Feedback

Key Discussion Points

- **Report Structure:** Supervisor reviewed the draft. Pointed out that the "Methodology" section was too descriptive of the code. It needs to explain the *algorithm logic* more clearly (pseudocode/flowcharts).
- **Benchmark Noise:** The current results have high variance.
  - *Advice:* Run benchmarks on a dedicated machine or ensure background processes are minimized. Use "median" instead of "average" to filter outliers.
- **Future Work:** Discussed the plan for Semester 2. Supervisor strongly suggested exploring **Parallelization** (using `std::thread` or OpenMP) as the next major step.

Supervisor Feedback

> "The results look promising, but a table of numbers is hard to read. Generate line graphs showing 'Time vs Input Size'. Also, include a 'Speedup' graph relative to std::sort."

> "For the presentation video, make sure you *show* the sorting happening (maybe a small visualization or live demo of the benchmark running). Don't just read slides."

4. Issues & Challenges

- **Small Array Performance:** DPQS is slower than `std::sort` for small arrays (<100 elements).
  - *Resolution:* This is expected. Discussed implementing a hybrid approach (switching to Insertion Sort for small partitions), similar to Introsort. This will be a key task for the next phase.

5. Action Items

- [ ] Generate graphs for the Interim Report results section. (Due: 2025-12-10)

- [ ] Record and edit the Interim Presentation Video. (Due: 2025-12-15)
- [ ] Finalize and submit the Interim Report. (Due: 2025-12-18)
- [ ] Begin research on Parallel Quicksort strategies. (Due: Jan 2026)

6. Next Meeting Plan

- **Target Date:** 2026-01-15 (Semester 2)
- **Focus:** Parallel implementation plan and feedback on Interim Assessment.

Meeting Note 4

Date: 2026-01-02 **Time:** 14:00 - 15:00 **Location:** Supervisor's Office (PQ706) **Attendees:** LIN Zhanzhi, CAO Yixin (Supervisor)

1. Objectives

- Provide a comprehensive progress report on the completed sequential and parallel modules.
- Present the benchmarking results showing performance superiority over `std::sort`.
- Validate the scope and structure for the upcoming Interim Report.
- Discuss technical decisions regarding SIMD and parallel architecture.

2. Progress Report

- **Algorithmic Core**:
  - Successfully implemented **3-way partitioning** with "Median-of-5" pivot selection, achieving ~20% fewer swaps.
  - Integrated **Adaptive Run Merging** (TimSort-style) to handle pre-sorted data efficiently.
  - Implemented **Introsort Strategy** (fallback to HeapSort) to guarantee $O(N \log N)$ worst-case safety.
  - Added **Counting Sort** optimization for small integer types (`byte`, `short`).
- **Parallel Architecture**:
  - Deployed a **Work-Stealing Thread Pool** for dynamic load balancing.
  - Completed contention analysis and mutex optimization.
- **Robustness**:
  - Ensured namespace safety and fixed integer overflow issues for large arrays.
  - Verified generic support with custom comparators.
- **Infrastructure**:
  - Established a custom **Benchmarking Harness** with "Destructive Testing" support.
  - Performed rigorous sample size analysis to ensure statistical significance.

3. Discussion & Feedback

Key Discussion Points

- **Sequential Performance**: Reviewed the data showing DPQS outperforming `std::sort` on random integers and achieving massive speedups on structured data (e.g., Organ Pipe). Supervisor acknowledged the efficacy of the "Run Merging" strategy.
- **Parallel Scaling**: Discussed the "Strong Scaling" results. The linear scaling up to 4 threads and saturation beyond 8 threads (due to Memory Bandwidth) was accepted as a valid finding for the Interim Report.
- **SIMD Feasibility**: Presented the decision to prioritize Parallelism over SIMD for Semester 1. Supervisor agreed that SIMD complexity is high and fitting as a "stretch goal" for Semester 2.
- **Comparison Targets**: Confirmed that comparing against `std::sort` and `std::stable_sort` is sufficient for the interim phase. Future work can include comparisons with TBB or OpenMP.

Supervisor Feedback

> "The algorithmic tuning, especially the Adaptive Run Merging, is a strong point. Make sure to highlight *why* 3-way partitioning helps with duplicates in your report."

> "Your identification of the 'Memory Wall' in the parallel section is critical. Ensure your Interim Report explains this hardware bottleneck clearly—don't just show the graph, explain the saturation."

> "The scope for the Interim Report is appropriate. The focus on 'Implementation & Validation' is solid. You don't need AVX-512 yet; getting the thread pool working correctly was the right priority."

4. Issues & Challenges

- **Parallel Overhead**: Discussed the challenge of managing thread overhead for small-to-medium arrays.
  - *Resolution:* Masked by the dynamic threshold ($N < 4096$ falls back to sequential), which effectively amortizes the cost.
- **SIMD Complexity**: Acknowledged that AVX-512 implementation is complex.
  - *Resolution:* Formally moved to Semester 2 plan as "Advanced Optimization".

5. Action Items

- [ ] Finalize the Interim Report based on the approved outline. (Due: 2026-01-09)
- [ ] Prepare the Interim Presentation slides, focusing on the "Speedup Analysis" graphs. (Due: 2026-01-09)
- [ ] Refine the work-stealing deque implementation for better load balancing in edge cases. (Due: Feb 2026)
- [ ] Begin preliminary investigation into AVX-512 intrinsics for the "Future Work" section. (Due: Feb 2026)

6. Next Meeting Plan

- **Target Date:** 2026-02-27 (Semester 2 Start)
- **Focus:** Detailed plan for SIMD implementation and advanced parallel optimizations.