Project #2: Cache & Memory Performance Profiling

Modern CPUs have a deep memory hierarchy (L1/L2/L3 caches + DRAM). Each level differs by orders of magnitude in latency and bandwidth. Like storage, memory shows a throughput–latency trade-off: as concurrency (outstanding memory requests) rises, bandwidth approaches a limit while average latency increases due to queuing.

Learning Goals (What your experiments must reveal)

- **Zero-queue latency** for **L1, L2, L3, and DRAM** (read & write where meaningful).
- **Maximum DRAM bandwidth** under different **access granularities/strides** (≈64B, ≈256B, ≈1024B) and **read/write mixes** (100%R, 100%W, 70/30, 50/50).
- The **throughput–latency trade-off** in DRAM as **access intensity** grows (MLC *loaded-latency* sweep).
- **Impact of cache-miss ratio** on the speed of a lightweight kernel (e.g., multiply or SAXPY).
- **Impact of TLB-miss ratio** on the same kernel's speed.

Tools You'll Use

- **Intel Memory Latency Checker (MLC)** for latency, bandwidth, and *loaded-latency* sweeps.
- **Linux perf** for cache/TLB miss measurement and correlation to performance.

Experimental Knobs

1. **Access pattern / granularity**: sequential vs. random; strides ≈64B / ≈256B / ≈1024B.
2. **Read/write ratio**: 100%R, 100%W, 70/30, 50/50.
3. **Access intensity (concurrency)**: multiple outstanding requests or threads; MLC's *loaded-latency* mode.

Required Experiments & Plots

1. **Zero-queue baselines**: Measure per-level latencies; table results.
2. **Pattern & granularity sweep**: Matrix covering pattern × stride; plot latency & bandwidth.
3. **Read/Write mix sweep**: Four ratios; discuss hardware effects.
4. **Intensity sweep**: ≥3 intensities; plot throughput vs latency; mark and explain "knee" using Little's Law.
5. **Working-set size sweep**: Show locality transitions; annotate regions.
6. **Cache-miss impact**: Use a light kernel; vary miss rate; correlate with performance.
7. **TLB-miss impact**: Vary page locality; test huge pages; correlate.

Reporting & Deliverables (post everything on Github)

- Scripts/commands, raw data, plotting code.
- System configuration and methodology.
- Clearly labeled plots/tables with units and error bars.
- Analysis tied to theory and counter data.
- Discussion of anomalies/limitations.

Grading Rubric (Total 230 pts)

1. **Zero-queue baselines (30 pts)**
    - (10 pts) Correct methods for isolating single-access latency.
    - (10 pts) Accurate per-level latency values with correct units.
    - (10 pts) Clear table format with CPU frequency conversions.
2. **Pattern & granularity sweep (40 pts)**
    - (15 pts) Complete coverage of required patterns/strides.
    - (15 pts) Plots showing both latency & bandwidth from same runs.
    - (10 pts) Insightful discussion of results, including prefetch and stride effects.

3. **Read/Write mix sweep (30 pts)**
    - (10 pts) Correct implementation of four R/W ratios.
    - (10 pts) Coherent explanation of observed performance differences.
    - (10 pts) Properly labeled and formatted plots.
4. **Intensity sweep (60 pts)**
    - (20 pts) ≥3 distinct intensity levels; single throughput–latency curve.
    - (15 pts) Clear identification and justification of "knee."
    - (15 pts) % of theoretical peak bandwidth with discussion of diminishing returns.
    - (10 pts) Logical tie-in to Little's Law.
5. **Working-set size sweep (20 pts)**
    - (10 pts) Annotated plots showing L1/L2/L3/DRAM transitions.
    - (10 pts) Correctly matched transition points to measured latencies.
6. **Cache-miss impact (25 pts)**
    - (10 pts) Proper control of miss rate through footprint/pattern changes.
    - (10 pts) Accurate correlation between perf counters and runtime.
    - (5 pts) Correct application of AMAT model to explain results.
7. **TLB-miss impact (25 pts)**
    - (10 pts) Methodologically sound variation of page locality and huge pages.
    - (10 pts) Accurate TLB miss measurement and correlation.
    - (5 pts) Discussion of DTLB reach and its effect on performance.

Tips for Successful Execution

- **Pin and isolate cores** using taskset or numactl to avoid interference from other processes.
- **NUMA awareness**: Keep threads and memory on the same NUMA node unless measuring remote access.
- **Disable frequency scaling**: Set CPU governor to performance for consistent results.
- **Warm-up runs**: Perform initial iterations to stabilize caches and clock frequency.
- **Randomize experiment order**: Helps avoid systematic bias from thermal or frequency drift.
- **Use error bars**: Repeat each test ≥3 times and report mean ± standard deviation.
- **Document everything**: SMT state, prefetcher configuration, and exact tool versions.
- **Check tool flags**: MLC and perf event names can vary by version—record what you used.
- **Be mindful of prefetchers**: Large strides or random patterns reduce their effectiveness.
- **Think about bottlenecks**: Past the knee, latency rises sharply with little throughput gain.