Master Thesis Proposal: Storage-/I/O-Aware Extension of BottleMod for Modeling I/O-Intensive Workloads

Student: <Your Name>
Supervisor: <Supervisor Name>
Institution: <University / Chair>

February 9, 2026

1 Motivation

Workflow and performance models are most useful when they correctly identify the limiting resource and explain *why* a task is slow. For I/O-intensive tasks, bottlenecks depend on (i) access pattern (sequential vs. random, request sizes), (ii) cache reuse and eviction, and (iii) tier characteristics (bandwidth and IOPS). Treating I/O as a single "throughput" number often fails for workloads where small random reads are IOPS-bound or where warm-cache effects dominate.

2 Background: BottleMod

BottleMod models a workflow task as a process with:

- **Requirement functions** over progress p for inputs (data) and resources (e.g., CPU), independent of time.
- **Input functions** over time t describing availability/allocations for those inputs/resources.
- A derived progress trajectory $P(t)$ and outputs as a function of progress.

This separation of concerns is attractive for scheduling and bottleneck analysis, but I/O behavior often depends on a mapping from logical accesses to physical tiers (cache vs. disk), which itself depends on reuse and cache capacity.

3 Problem Statement

Given a task with significant storage interaction, the baseline BottleMod abstraction can misrepresent:

- **Warm-cache speedups**: second pass over the same dataset is faster.
- **IOPS bottlenecks**: many small requests saturate IOPS despite available bandwidth.
- **Tier shifts and thrashing**: performance changes when working set exceeds cache.

The goal is to extend BottleMod so that these effects can be represented with minimal additions, while keeping the core progress solver unchanged.

4 Research Questions

1. How can I/O-intensive tasks be modeled in BottleMod such that bottlenecks caused by *bandwidth* and *IOPS* are distinguishable and correctly attributed to storage tiers?
2. What is a practical interface between process descriptors (access pattern/reuse) and environment descriptors (tier capacities and performance) that remains progress-based (time-independent) on the process side?
3. Does the extension measurably reduce prediction error and improve bottleneck classification on representative I/O workloads compared to the baseline model?

5 Proposed Approach

5.1 Model Extension (Storage-/I/O-Aware BottleMod)

The extension models storage as a set of resource constraints per tier. On the **process side**, for each dataset k we describe a *logical access profile* over progress p:

- Logical bytes read/write: $A_k^r(p), A_k^w(p)$
- Logical operations read/write (IOPS proxy): $Q_k^r(p), Q_k^w(p)$

On the **environment side**, for each tier j (e.g., memory/page cache, SSD, HDD) we expose time-dependent capacities:

- Read/write bandwidth $I_j^{bw,r}(t), I_j^{bw,w}(t)$
- Read/write IOPS $I_j^{iops,r}(t), I_j^{iops,w}(t)$

To connect both sides, a *tier mapping* (hit/miss behavior) provides fractions $H_{k,j}(p)$ of accesses served by tier j over progress (either measured directly or approximated from cache size and reuse descriptors). From A, Q, H we derive ordinary BottleMod resource requirement functions per tier for bandwidth and IOPS. The progress computation remains unchanged; only the set of resources expands.

5.2 Parameterization Strategy

Two levels of parameterization are planned:

- **Minimal (manual/fit)**: piecewise $H(p)$ and piecewise-linear $A(p), Q(p)$ fitted from short calibration runs.
- **Heuristic (cache-aware)**: derive $H(p)$ from cache capacity vs. working set size and access pattern class (sequential, random, strided), validated against measurements.

6 Evaluation Plan

Experiments will target workloads where baseline models often fail:

- **Sequential scan (cold vs. warm)** with file sizes below/near/above memory.
- **Two-pass analytics** to validate warm-cache speedup and bottleneck shifts.
- **Random reads** (e.g., 4 KiB) to validate IOPS binding and request-size sensitivity.
- **Thrashing scenarios** where working set exceeds cache.

Measurements will include wall-clock time, achieved bandwidth/IOPS, and cache indicators. Predictions will be produced for (i) baseline BottleMod and (ii) the storage-/I/O-aware extension. Metrics:

- Runtime prediction error (e.g., MAPE / RMSE)
- Bottleneck classification accuracy (which resource binds over time)
- Qualitative correctness of phase changes (cold → warm, thrash)

7 Expected Contributions

- A BottleMod extension that models storage tiers and separates bandwidth vs. IOPS constraints.
- A practical parameterization workflow for I/O workloads (measurement + fitting / heuristics).
- An experimental evaluation demonstrating improved precision for I/O-intensive tasks.

8 Scope and Non-Goals

- Focus is on single-node storage behavior (page cache and block device). Distributed storage and network effects are out of scope unless required by the baseline BottleMod setup.
- The goal is improved modeling precision with minimal solver changes; not a full OS-level cache simulator.

9 Work Plan (Tentative)

1. Literature review: BottleMod, storage hierarchy modeling, cache/page-cache behavior, IOPS vs. bandwidth modeling.
2. Formalize the model interface (logical access profiles, tier mapping, tier resources).
3. Implement the extension in the existing codebase and add small synthetic examples.
4. Design and run microbenchmarks (e.g., fio-based) to characterize tier capacities.
5. Run evaluation workloads; compare baseline vs. extension; analyze sensitivity.
6. Write thesis and finalize artifacts (code + reproducible experiments).

10 Risks and Mitigations

- **Risk:** Cache behavior too complex to model accurately.

- **Risk:** Measurement noise / interference.
  Mitigation: Controlled single-machine experiments, repeated trials, cache control (drop caches) where applicable.