Evaluating Branch Prediction Schemes with an Out-of-Order Core in gem5

Manish Patel (MCS242460), Adithya R Narayan (2024MCS2445)

September 10, 2025

Background & Hypothesis

Dynamic branch prediction reduces front-end stalls by forecasting control-flow outcomes before resolution in deep pipelines; efficacy depends on history modeling, aliasing control, and chooser design in hybrid predictors. We hypothesize that TAGE/LTAGE, with tagged geometric histories, will outperform Local/Bimodal/Perceptron on branch-sensitive and loop-regular codes, yielding lower misprediction rates and modest IPC gains under O3 execution, whereas compute-bound `mm` will show near-constant IPC across predictors.

Experimental Methodology (Setup & Baseline)

Benchmarks. We evaluate three native microbenchmarks under gem5 SE mode: `mm` (matrix multiply from `mm.c`), `fft.1` (FFT from `fft.c`), and `branchy_test` (branch-stress microbenchmark from `branchy_test.c`), each built with fixed `N` for repeatable work.

gem5 configuration and CPU model. O3CPU (X86), single core; DDR3_1600_8x8 main memory; L1 I/D caches and a unified L2 cache. Predictors are the only variable across runs. Common CLI flags mirror: `-cpu-type=O3CPU -num-cpus=1 -mem-type=DDR3_1600_8x8 -l1d_size=32kB -l1i_size=32kB -l2cache -cmd=<binary> -options=<args>`.

Branch predictors. BiModeBP, LocalBP, TournamentBP, TAGE, LTAGE, and PerceptronBP (gem5 defaults unless stated).

Run procedure and data collection.

1. Build gem5 (optimized X86): `scons -j2 build/X86/gem5.opt`
2. Build benchmarks (in `Binaries/`): `make all`
3. Run experiments: `python3 script.py` (writes `Stats_BP/<cpu>_<bp>_<workload>/stats.txt`)
4. Collect stats: `python3 collect_stats_bp.py -src Stats_BP -out summary_for_plots_bp.csv`
5. Plot/analyze: `python3 compute_and_plot_accuracy.py -csv summary_for_plots_bp.csv -outdir branch_analysis` and `python3 plot_bp_accuracy.py`

Warmup/ROI. ROI is a fixed window of 100M committed instructions from program start for consistent cross-run comparisons. **Baseline.** O3CPU + LocalBP.

Metrics. From `stats.txt`: IPC (`system.cpu.ipc`), `simSeconds`, `simInsts`, `branchPred.lookups`, `branchPred.committed`, `branchPred.mispredicted`, `branchPred.mispredictDueToPredictor`. Derived: misprediction rate (committed) = `mispredicted`/`committed`; MPKI = `mispredicted`/(`simInsts`/1000).

Results: Tables

Table 1: `branchy_test`: IPC and misprediction rate (committed) per predictor (100M instr)

Predictor	IPC	Misprediction rate
BiModeBP	0.553988	0.0307567407
LocalBP	0.554012	0.0307249928
PerceptronBP	0.554012	0.0307249928
TournamentBP	0.554038	0.0307122678
TAGE	0.555315	0.0290625864
LTAGE	0.555246	0.0291269879

Table 2: `fft.1`: IPC and misprediction rate (committed) per predictor

Predictor	IPC	Misprediction rate
BiModeBP	1.827046	0.0330157189
LocalBP	1.763233	0.0470959519
PerceptronBP	1.763233	0.0470959519
TournamentBP	1.903487	0.0143297391
TAGE	1.929010	0.0086787643
LTAGE	1.931516	0.0083550808

Table 3: `mm`: IPC and misprediction rate (committed) per predictor (100M instr)

Predictor	IPC	Misprediction rate
BiModeBP	0.208620	0.0026344423
LocalBP	0.208647	0.0024654733
PerceptronBP	0.208647	0.0024654733
TournamentBP	0.208657	0.0024570112
TAGE	0.208635	0.0024209790
LTAGE	0.208638	0.0010048059

Results: Figures and Discussion

Global accuracy overview

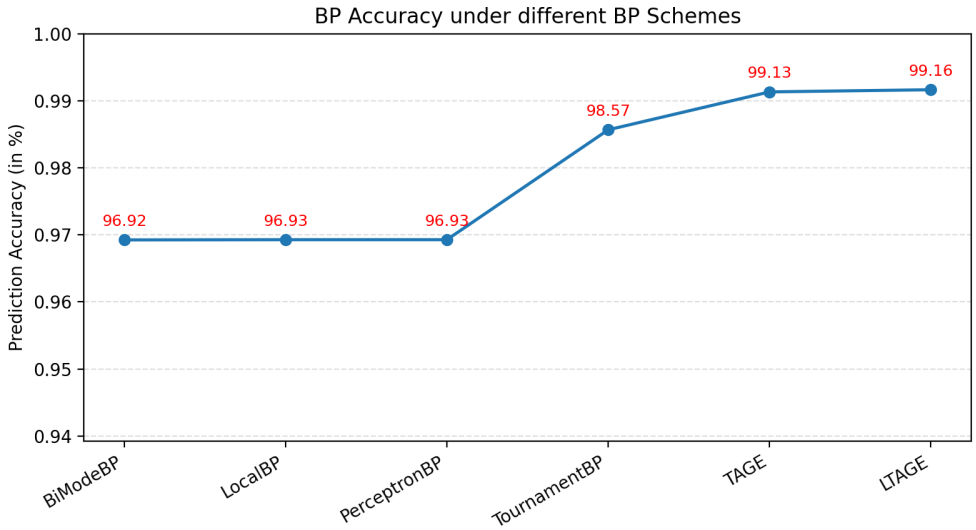

Figure 1: Overall prediction accuracy across predictors (all workloads combined).

The overall accuracy view shows a consistent ordering where Tournament and tagged predictors (TAGE/LTAGE) sit above Local/Bimodal/Perceptron, foreshadowing the per-workload trends discussed below.

branchy_test (branch-heavy)

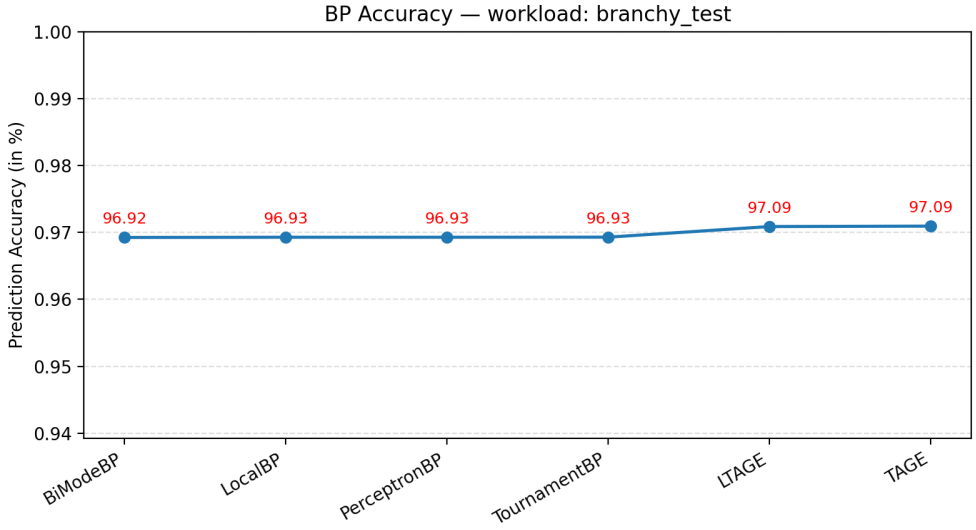

Figure 2: `branchy_test`: prediction accuracy by predictor.

Accuracy clusters near 97% for all predictors with a small but consistent lead for TAGE/LTAGE, indicating that longer tagged histories slightly reduce aliasing and capture more of the control-flow regularity even in a branch-intensive setting.

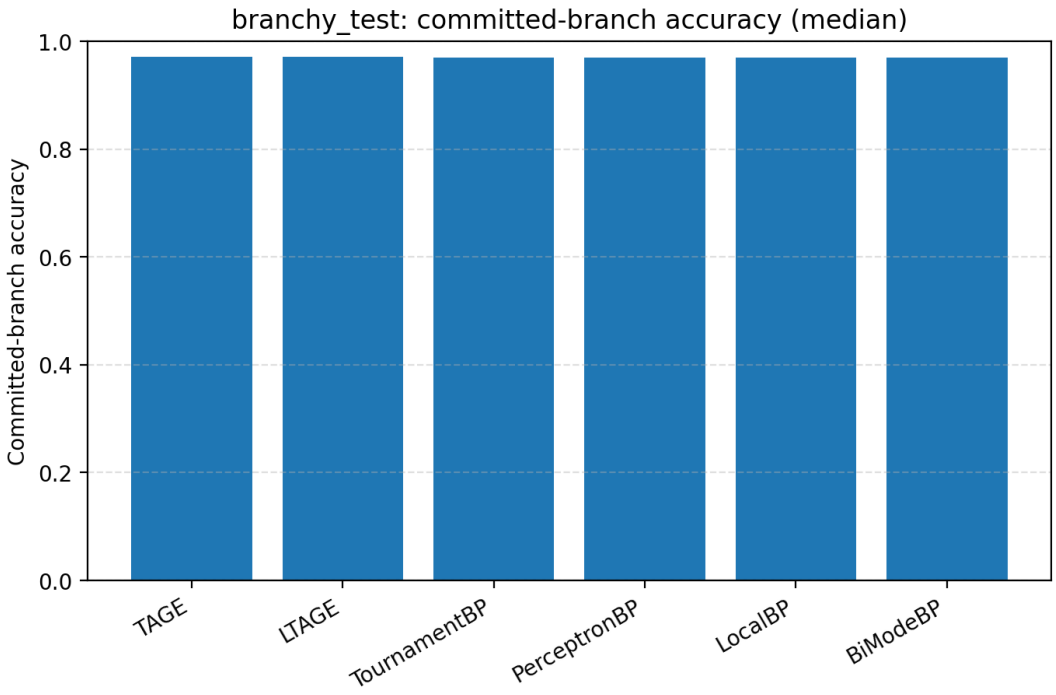

Figure 3: `branchy_test`: committed-branch accuracy (bar).

Committed-branch accuracy bars corroborate the line plot, showing a narrow spread where the tagged designs edge out hybrid and local/bimodal, aligning with the lower misprediction rates seen in the table for TAGE/LTAGE.

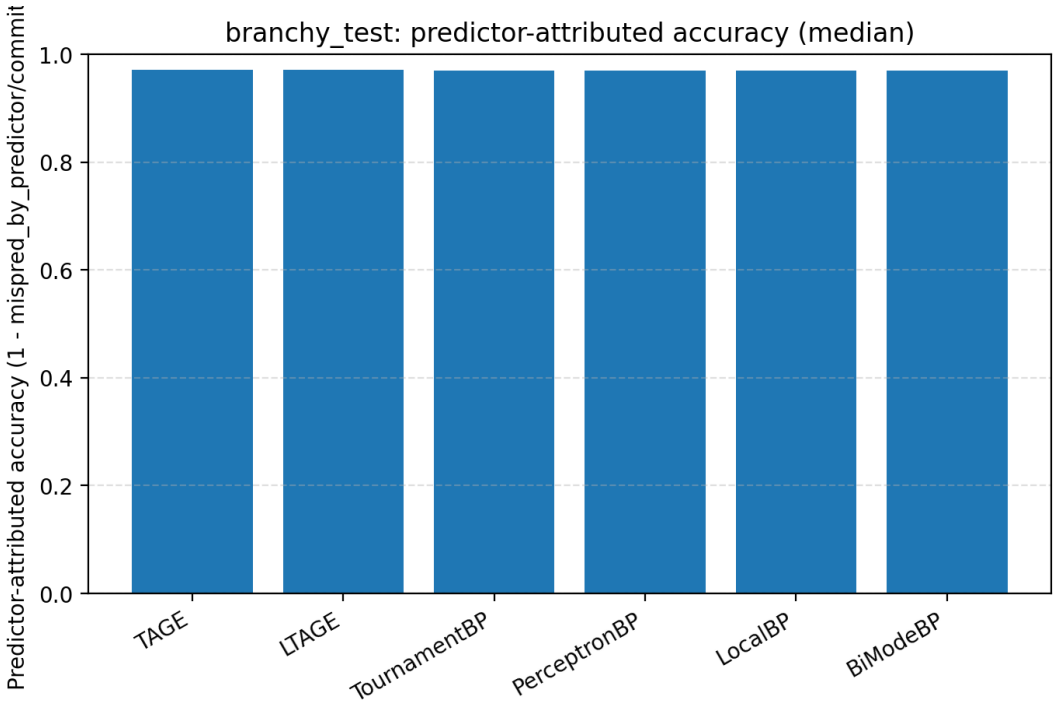

Figure 4: `branchy_test`: predictor-attributed accuracy (bar).

Attribution to the predictor (excluding non-predictor causes) preserves the same ordering, reinforcing that most differences arise from prediction quality rather than external redirect causes.

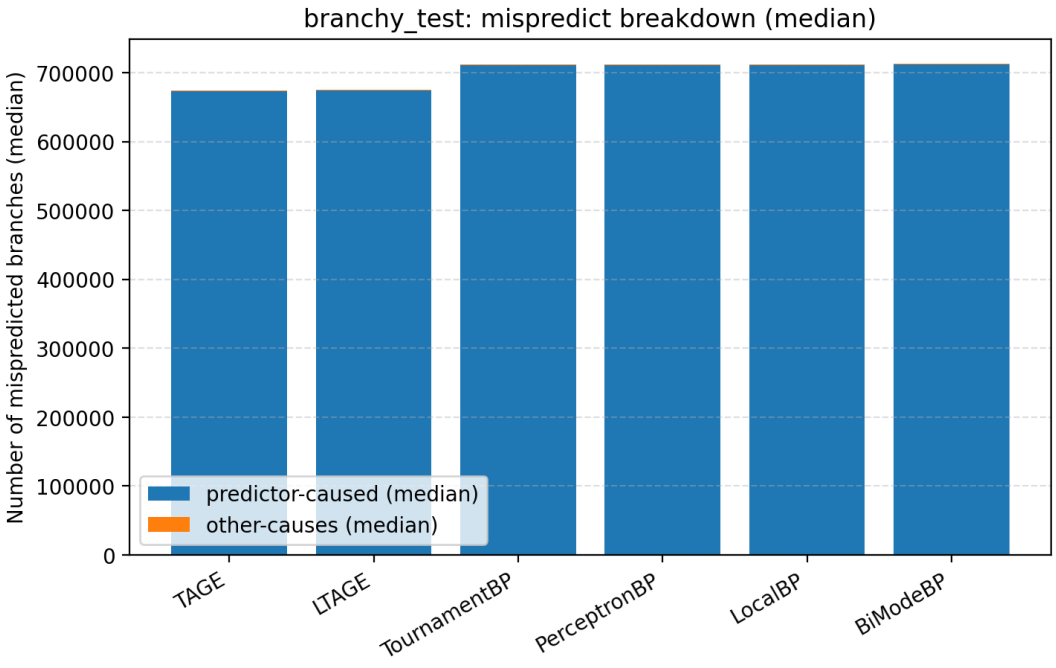

Figure 5: `branchy_test`: misprediction breakdown (median).

The breakdown indicates that the majority of mispredictions are predictor-caused across designs;

reductions from TAGE/LTAGE thus directly translate to fewer pipeline redirects and explain the small IPC gains for these predictors.

fft.1 (loop-regular compute)

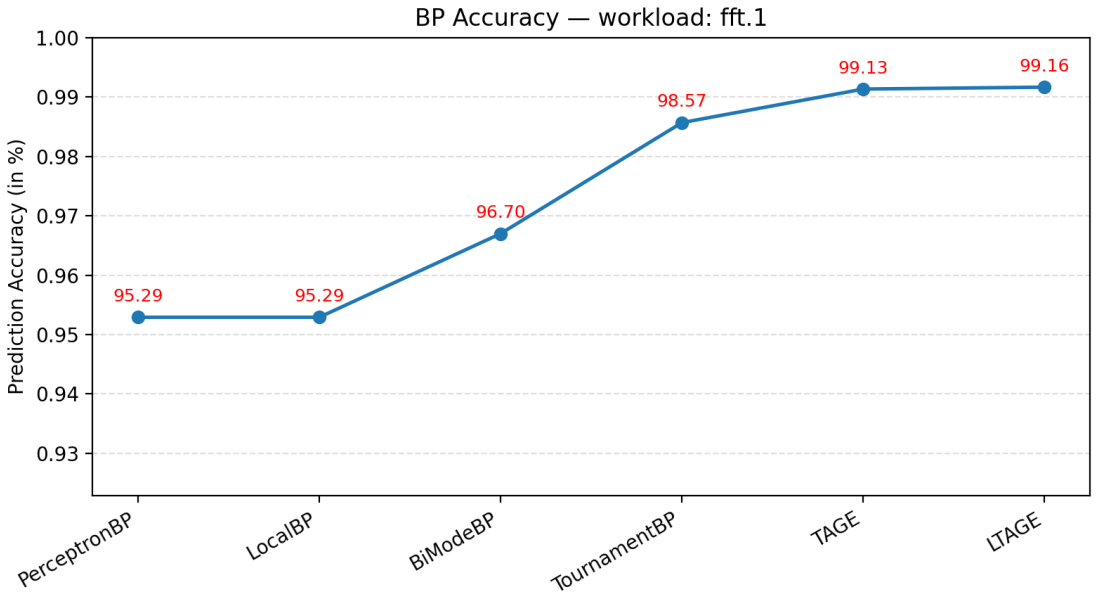

Figure 6: `fft.1`: prediction accuracy by predictor.

A clear staircase emerges: Local/Perceptron (∼95.3%), BiMode (∼96.7%), Tournament (∼98.6%), and TAGE/LTAGE (∼99.1–99.16%), matching the expectation that global and long histories exploit repetitive loop control extremely well and yield the highest IPC.

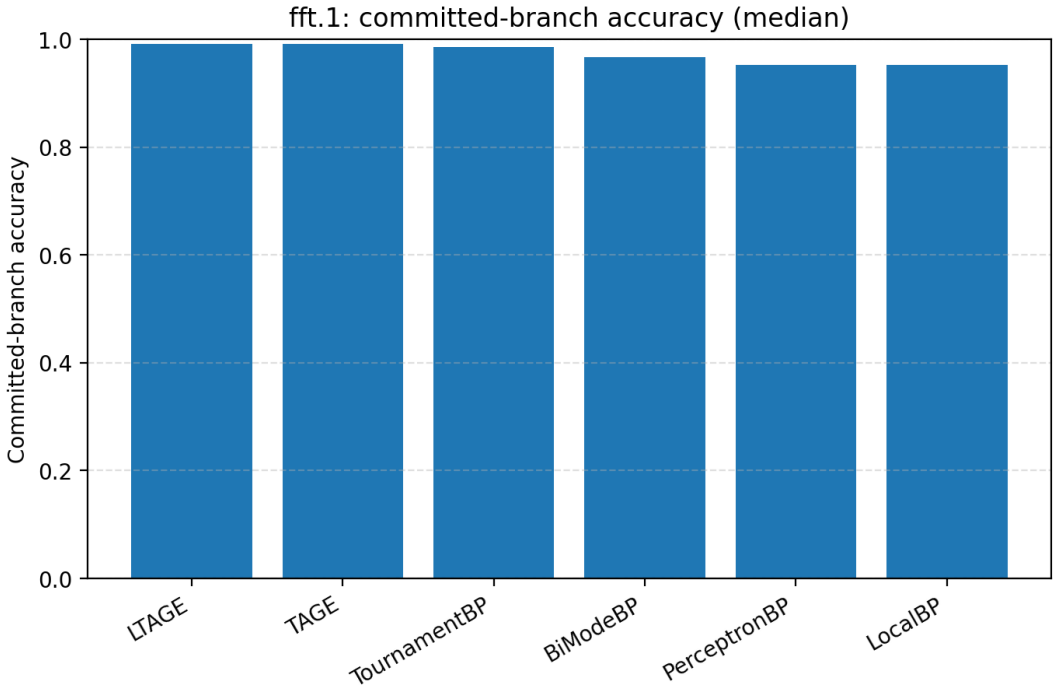

Figure 7: `fft.1`: committed-branch accuracy (bar).

The bar view highlights the large gap between Local/BiMode and Tournament/TAGE-class designs on this workload, consistent with the stronger IPC observed for Tournament and TAGE/LTAGE.

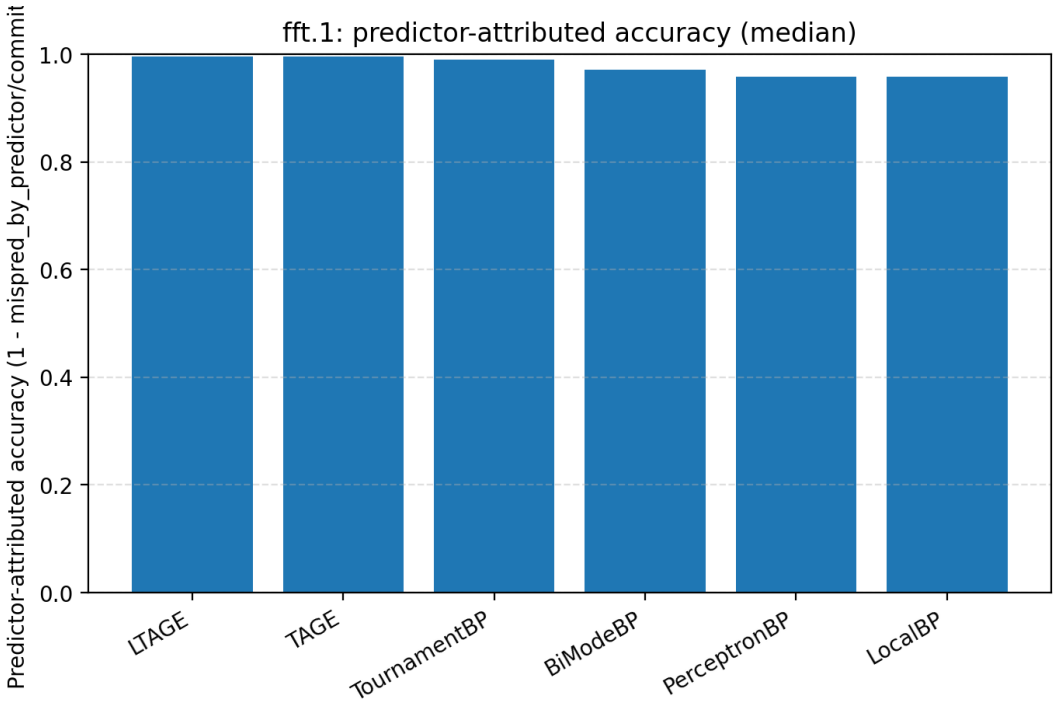

Figure 8: `fft.1`: predictor-attributed accuracy (bar).

Predictor-attributed accuracy emphasizes that the improvement is rooted in the predictor itself (rather than target or recovery artifacts), reinforcing the mechanism-driven advantage of tagged and hybrid designs.

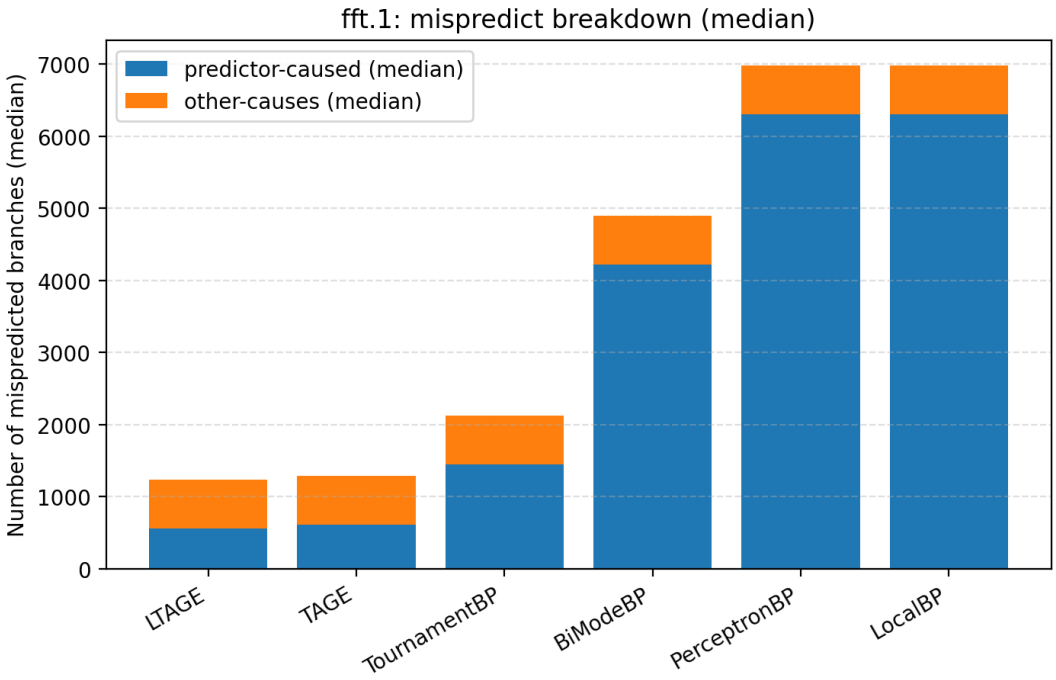

Figure 9: `fft.1`: misprediction breakdown (median).

Here, the predictor-caused component drops noticeably for Tournament and TAGE/LTAGE compared to Local/BiMode, explaining the sharper IPC gains due to fewer flushes in tight loops.

mm (compute-heavy)

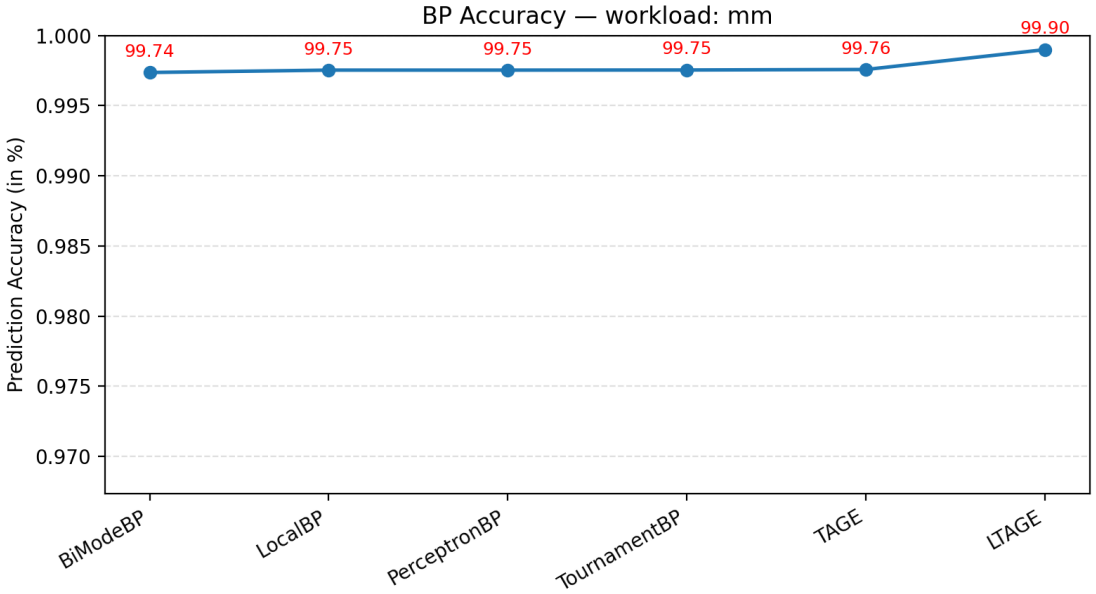

Figure 10: `mm`: prediction accuracy by predictor.

All predictors achieve near-perfect accuracy (≈99.75–99.90%), reflecting trivial and predictable branches in this kernel; as a result, accuracy differences are practically irrelevant to performance.

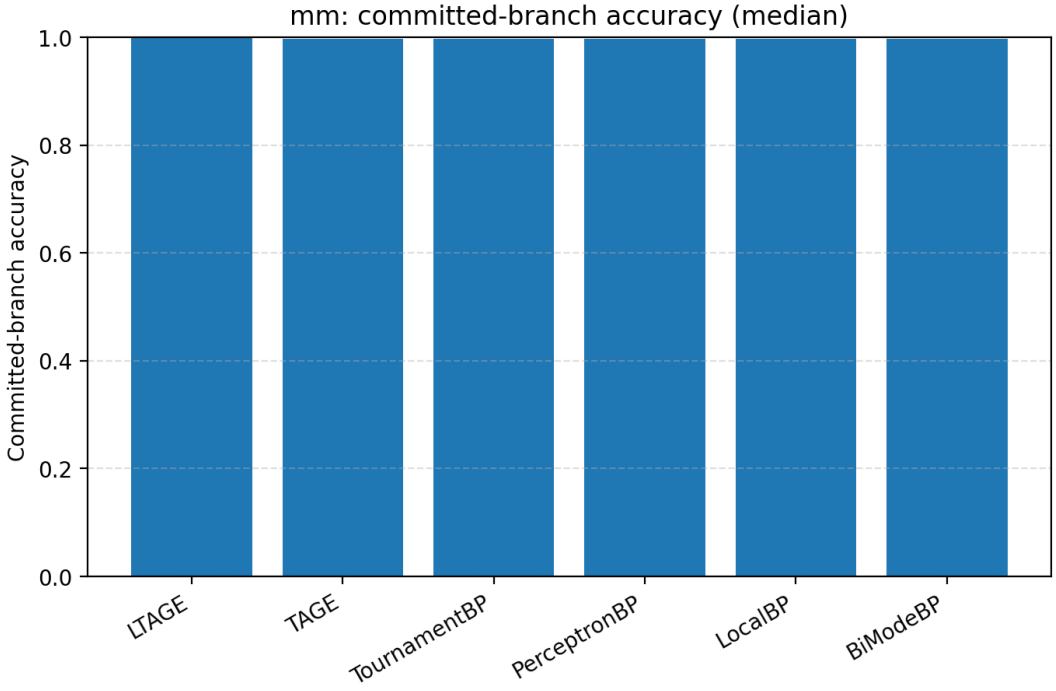

Figure 11: `mm`: committed-branch accuracy (bar).

The bars confirm the tight clustering in accuracy, indicating that branch prediction is not a performance discriminator for this workload.

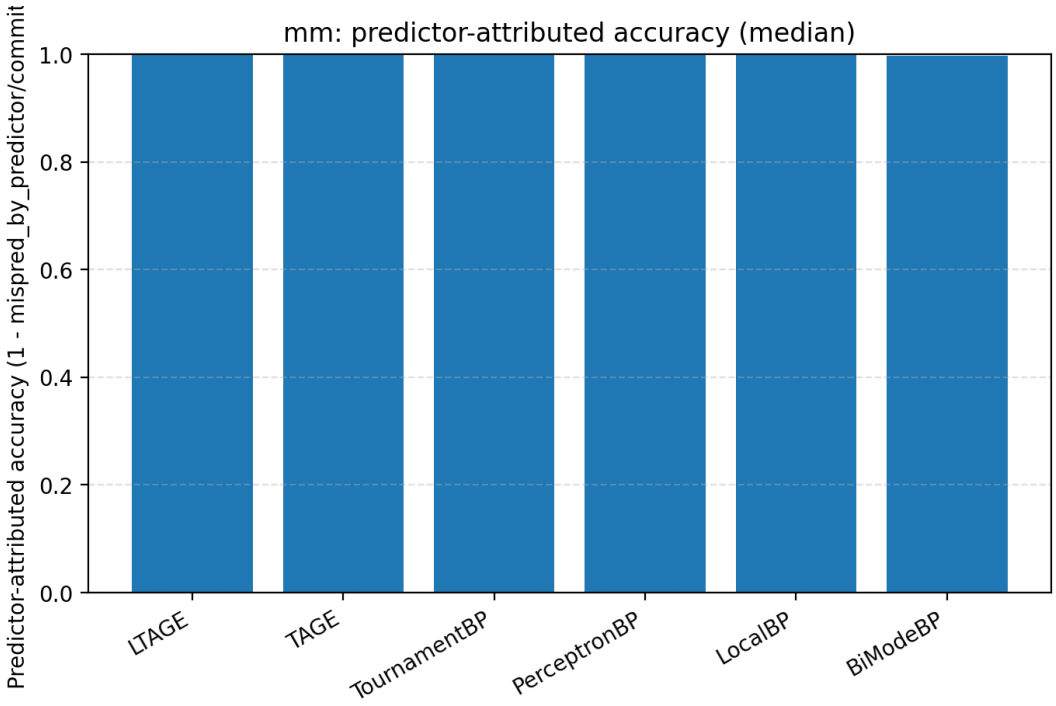

Figure 12: `mm`: predictor-attributed accuracy (bar).

Predictor-attributed accuracy is similarly saturated, reinforcing that any IPC limits arise from the compute/memory back end rather than from front-end control-flow redirection.

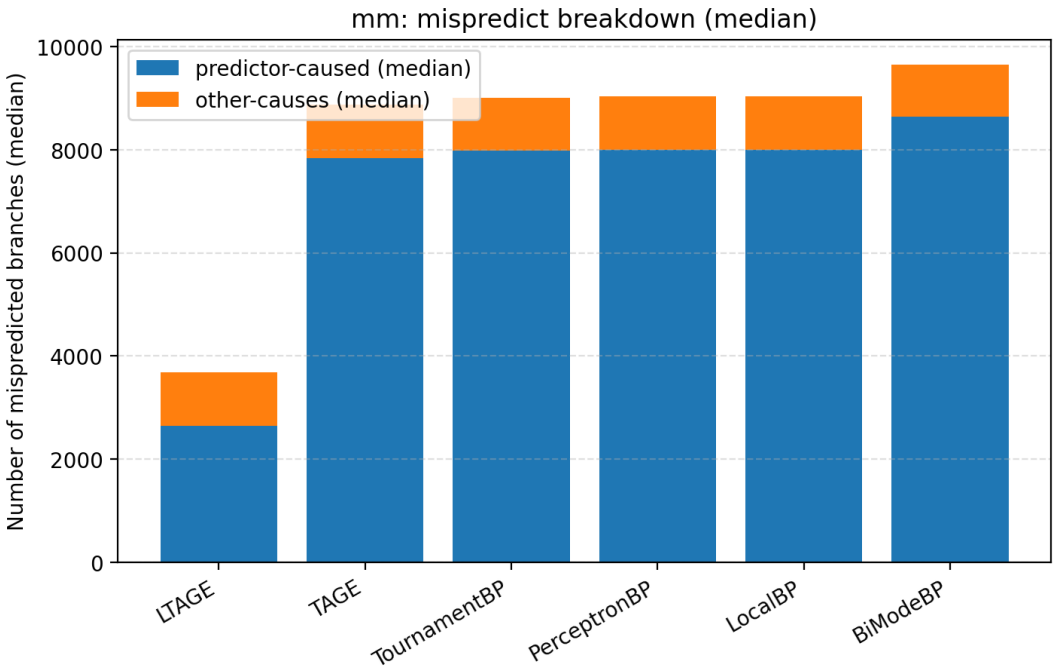

Figure 13: `mm`: misprediction breakdown (median).

The breakdown shows very low counts for predictor-caused mispredictions across the board, consistent with the nearly identical IPCs reported for all predictors.

Discussion

Across the three workloads, predictor rankings align with mechanism: tagged geometric histories (TAGE/LTAGE) yield the most consistent accuracy wins, with Tournament competitive due to chooser selection between local and global signals. These accuracy improvements translate into measurable IPC gains where control flow is on the critical path (branchy and loop-regular codes), but have negligible impact where the back end or memory dominates (compute-heavy `mm`). Overall, upgrading from local/bimodal to tournament or TAGE-class predictors provides the highest benefit on control-intensive applications, while compute-bound kernels are indifferent to predictor sophistication.

Conclusion and Limitations

TAGE and LTAGE consistently provide the best accuracy and top IPC on branch-sensitive and loop-regular workloads, with Tournament close behind, while BiMode/Local/Perceptron trail on difficult branches; on compute-heavy `mm`, predictor choice has negligible IPC impact. Using a fixed 100M-instruction ROI and a uniform O3 configuration isolates the predictor's effect and clarifies when accuracy translates to throughput versus when the performance bottleneck lies elsewhere.

Author Contributions

Conceptualization: Manish Patel, Adithya R Narayan.
Methodology/Experimental Design: Manish Patel (predictor matrix, run plan), Adithya R Narayan (benchmark parameters, ROI definition); jointly finalized.
Implementation (code/config): Manish Patel (gem5 configuration, predictor selection flags, run scripting); Adithya R Narayan (benchmark C programs: `mm.c`, `fft.c`, `branchy_test.c`; build system).
Data Collection: Manish Patel (automation and per-run directories), Adithya R Narayan (stats parsing and CSV verification).
Analysis/Visualization: Adithya R Narayan (accuracy and misprediction-breakdown plots), Manish Patel (tables for IPC/mispred, figure captions); interpretation written jointly.
Writing – Original Draft: Manish Patel (Background, Discussion), Adithya R Narayan (Methodology, Results).
Writing – Review & Editing: Manish Patel, Adithya R Narayan.
Presentation: Manish Patel (slide assembly and layout), Adithya R Narayan (slide narratives and figure selection).