LOGGING, CHECKPOINTS, AND RECOVERY

Eccentric Loggers // Haibin Lin, Matt Perron, Abhishek Joshi // 5/6/2016

OUTLINE

- Proposal Review
- Current State
- Benchmark Result
- Test Coverage
- Future Work

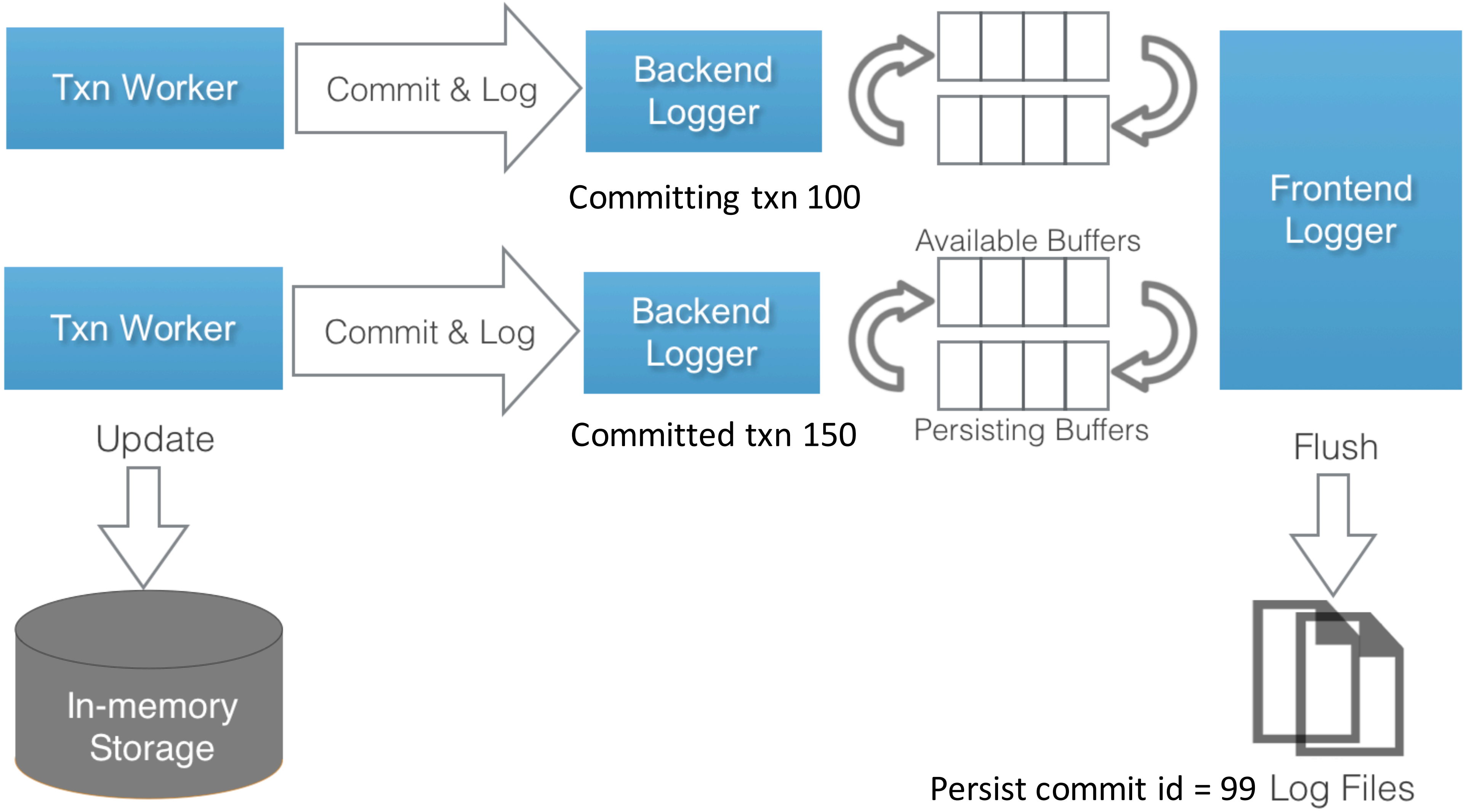
PROBLEMS WITH PREVIOUS DESIGN

- Single log file which grows without bound
  - Overuse of disk space
  - Difficult to truncate efficiently.
- Only one front-end logger for all back-end loggers
- No checkpoints
- Sequential Recovery
- **Cannot handle concurrent txns**
  - **Did not preserve txn order**
  - **Premature release of workers**

PROPOSAL REVIEW

- Correct single-threaded logging implementation
- Backpressure mechanism to prevent backup of logs on workers
- Making a multi-file log for truncation after taking a checkpoint
- Single-threaded checkpoints
- Correct single threaded recovery
- Multi-threaded logging and recovery from log

DESIGN OVERVIEW



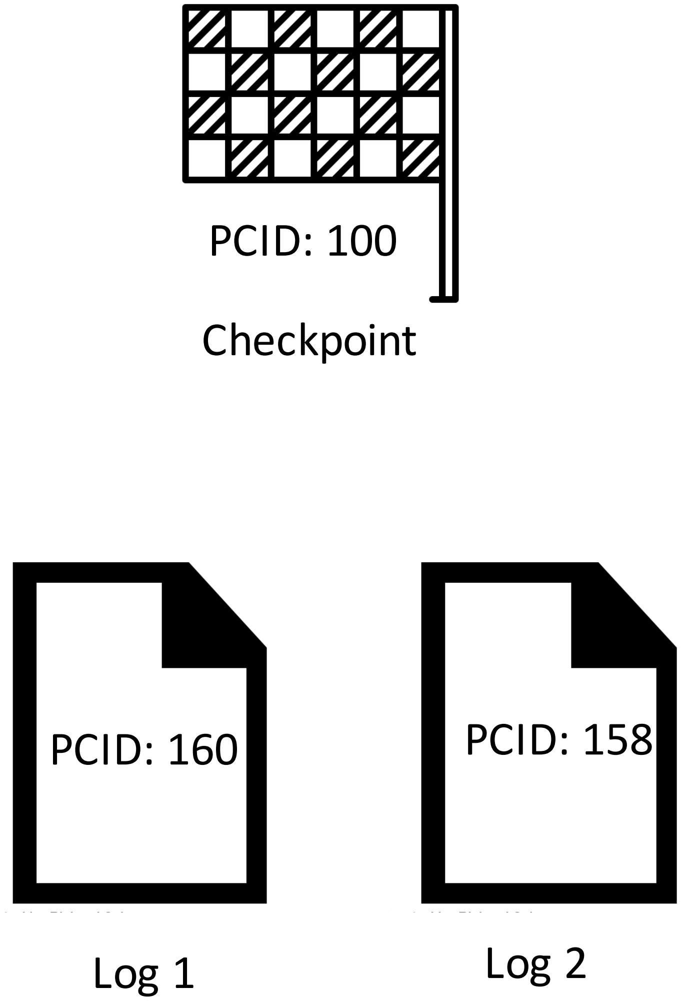
Committing txn 100

Committed txn 150

Persist commit id = 99

RECOVERY PROCESS

1. Recover Checkpoint
2. Find min PCID of logs
3. Recover transactions in log between Checkpoint id and persistent commit id
4. Rebuild Indexes



Page icon

Created by Richard Schumann
from Noun Project

Checkpoint icon

Created by hunotika
from Noun Project

BENCHMARK RESULTS

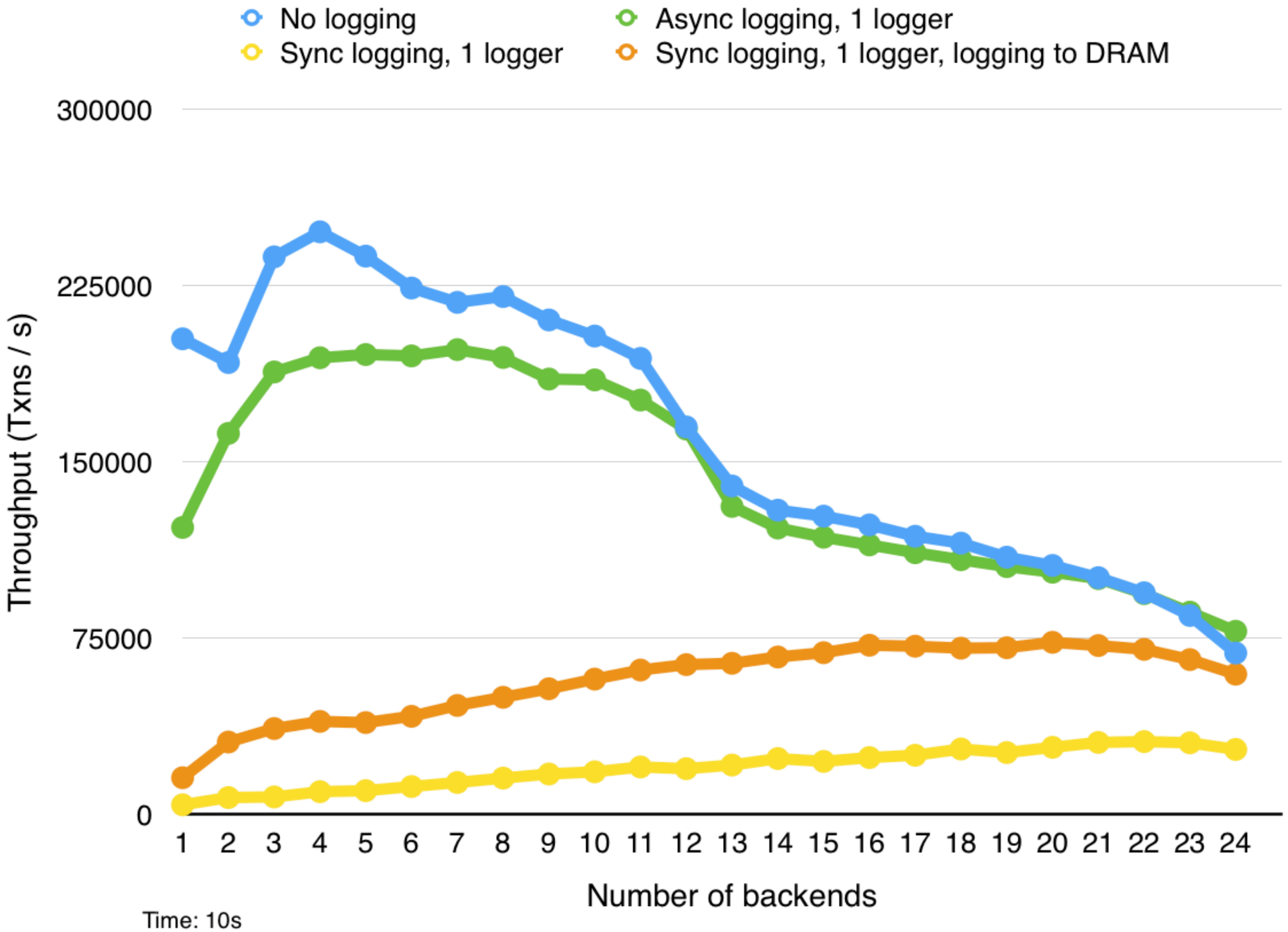
EXPERIMENT SETUP

- Dual-socket Xeon E5-2620
- 6 cores / 12 threads (24 hyper-threads)
- 3 SSD's

IMPACT OF LOGGING ON THROUGHPUT

YCSB Micro-benchmark

- 100% Insert

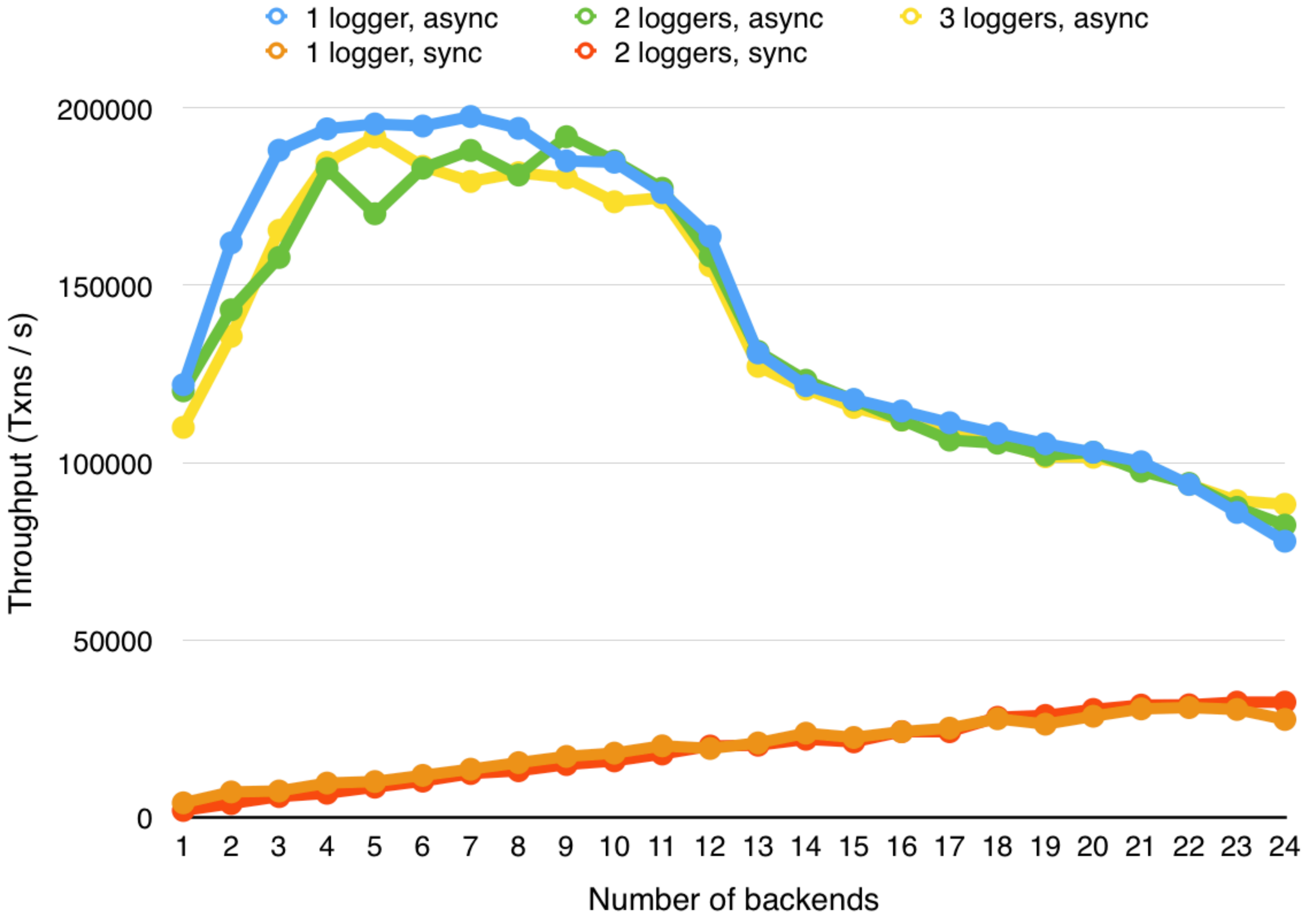


Time: 10s
Inserts Only

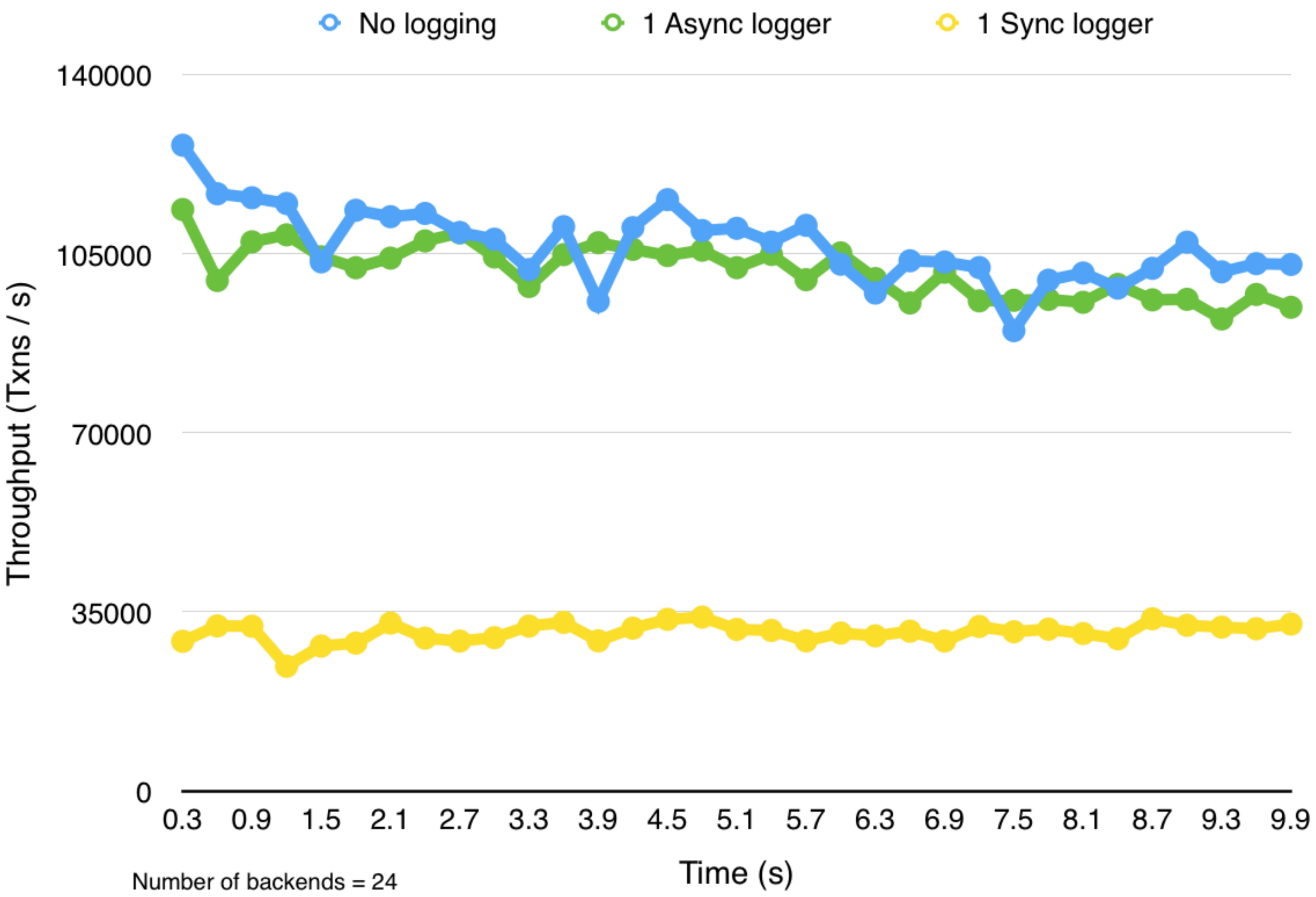
MULTIPLE LOGGERS

YCSB Micro-benchmark

- 100% Insert
- Multiple SSD's

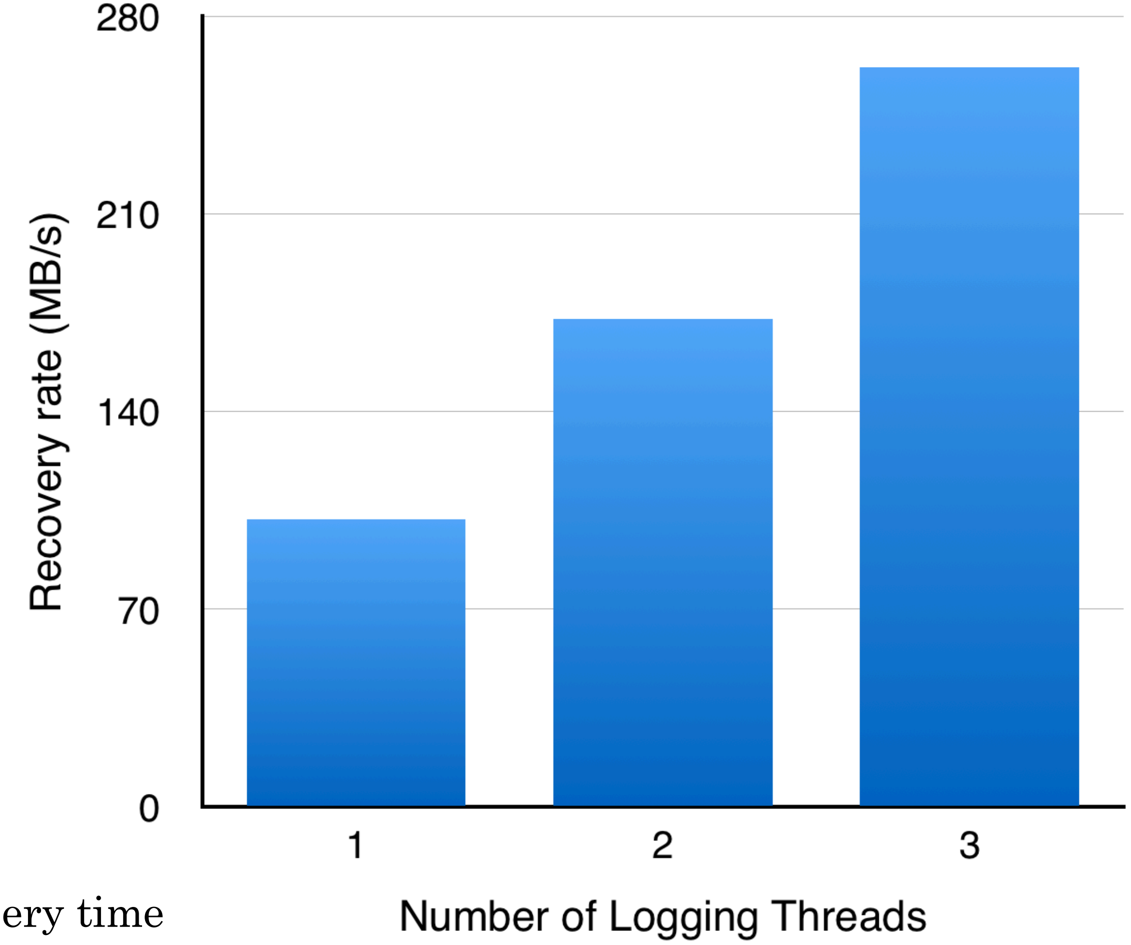


LOGGING THROUGHPUT



Number of backends = 24
Scale factor = 24
Insert Only

LOGGING RECOVERY SCALABILITY



∗ Includes single-thread index recovery time

CHECKPOINT STORAGE SAVINGS

YCSB Micro-benchmark

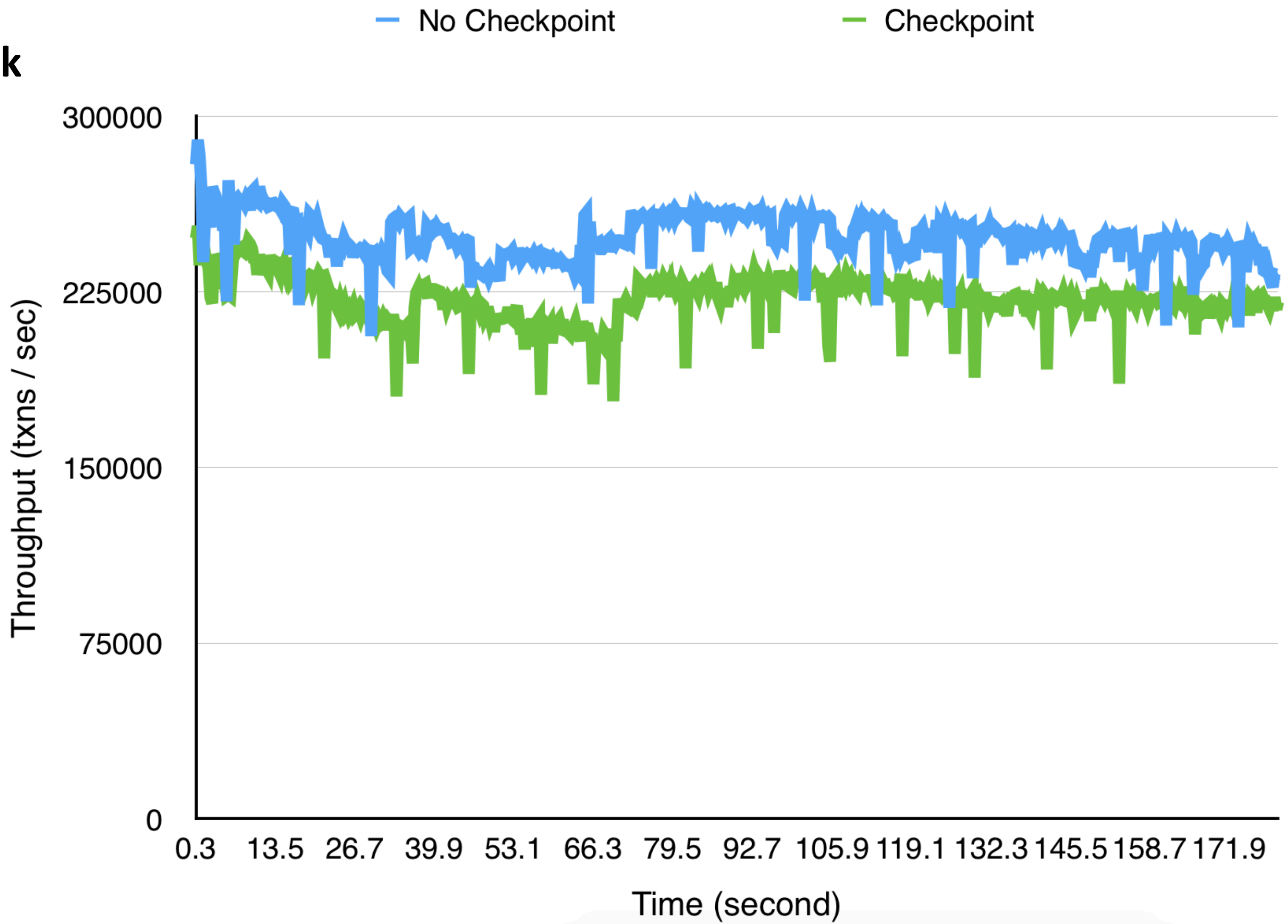
- 100% Update
- 1000 seconds, 1M tuples, result in 60GB in memory

	Log File Size	Checkpoint Size
No Checkpoint	30GB	
Has Checkpoint	32MB	1GB

LOW RUNTIME IMPACT OF CHECKPOINTS

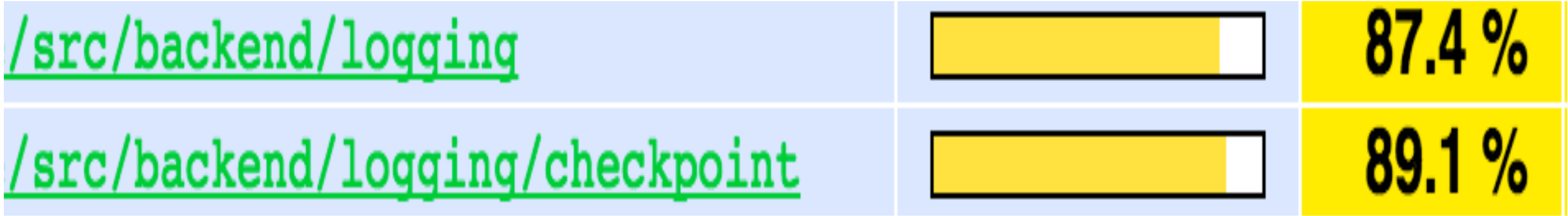
YCSB Micro-benchmark

- 100% Insert



TEST COVERAGE

- Unit Tests
- Scheduled tests for different ordering of worker logging operations (adapted from CC team)
- Coarse grained tests (insert/update, crash, test with ycsb)



/src/backend/logging	87.4 %
/src/backend/logging/checkpoint	89.1 %

CODE QUALITY

- Correctness of single threaded logging
- Backpressure mechanism (log buffers)
- Recovery from both checkpoint and logging
- Easy to extend
- File management

FUTURE WORK

- Further reducing recovery time
- Single threaded Checkpoint performance
- SiloR-style multithreaded checkpoints and multithreaded checkpoint recovery
- Preserve tile layout information in checkpoint
- Compressing log and checkpoint
- Data integrity checks of logs and checkpoint
- Performance investigation (as yet unobserved)
- Core-pinning workers and frontend loggers

PROPOSAL REVIEW

- 75% - Basic Checkpoint and Logging
  - Multi-file Log
  - Single-thread checkpoint and log creation
  - Single-thread recovery from checkpoint and log

- 100% - SiloR-style Logging
  - Multi-thread SiloR-style logging and recovery
  - Single-thread checkpoint

- 125% - SiloR-style Checkpoint
  - Multi-thread checkpoint creation and recovery
  - Reduce log size by compression