

WTB Architecture Diagrams

This document provides visual diagrams showing how the Workflow Test Bench (WTB) works for each action.

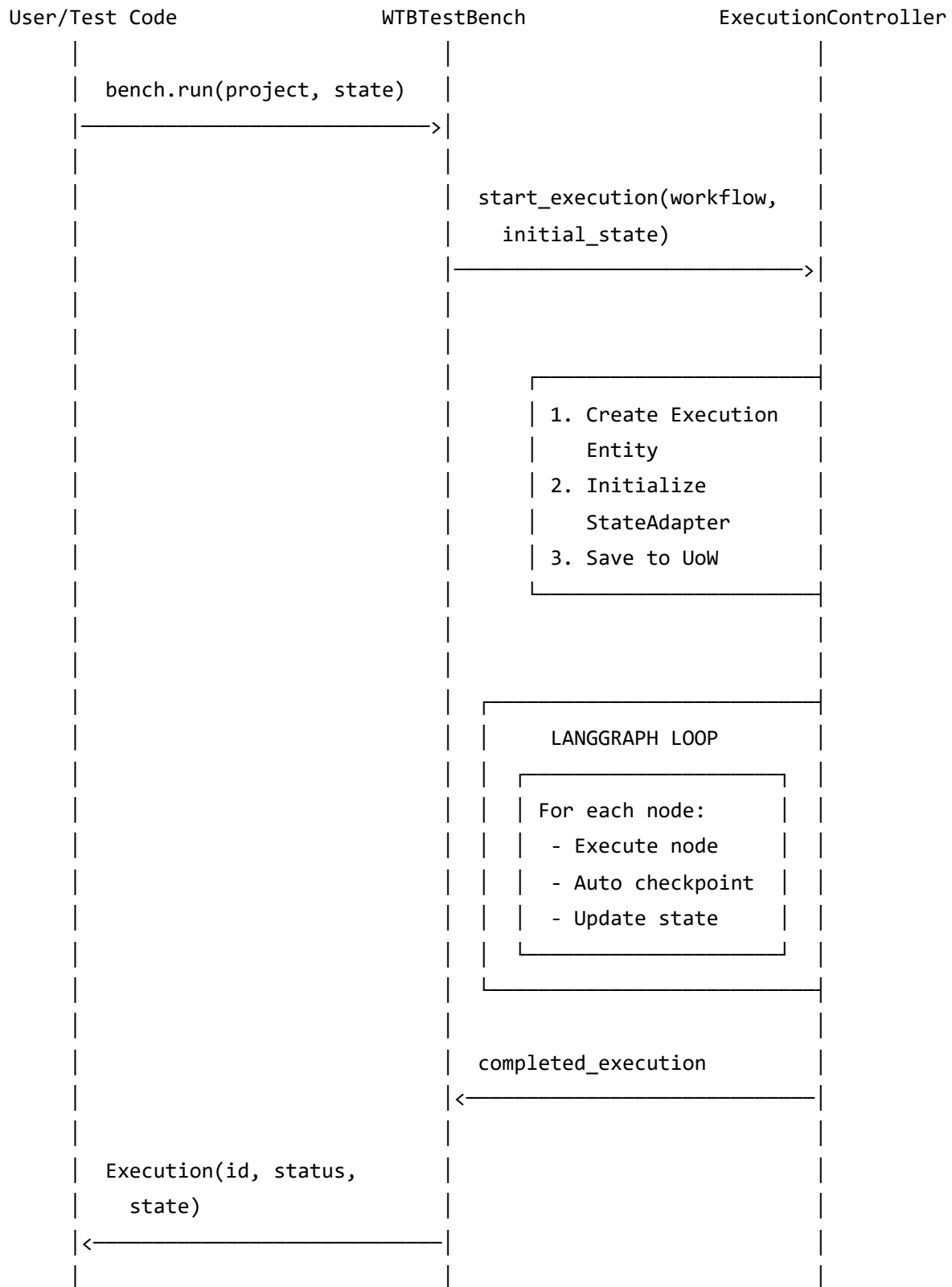
Table of Contents

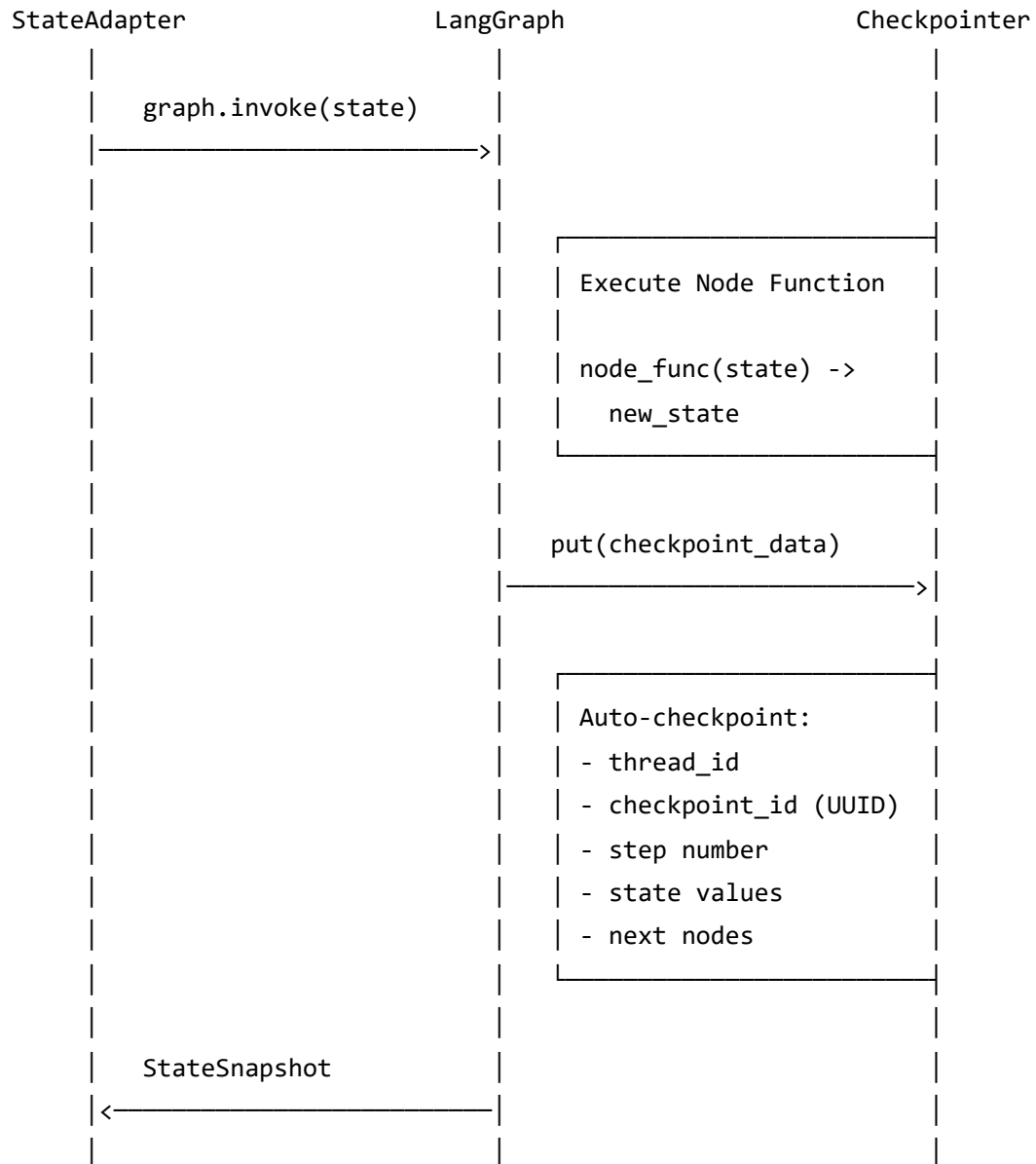
1. [Basic Execution Flow](#)
2. [Checkpointing Flow](#)
3. [Rollback Flow](#)
4. [Fork/Branch Flow](#)
5. [File Tracking Flow](#)
6. [Ray Batch Execution Flow](#)
7. [Component Overview](#)

1. Basic Execution Flow

How a workflow is executed from start to finish.

BASIC EXECUTION FLOW



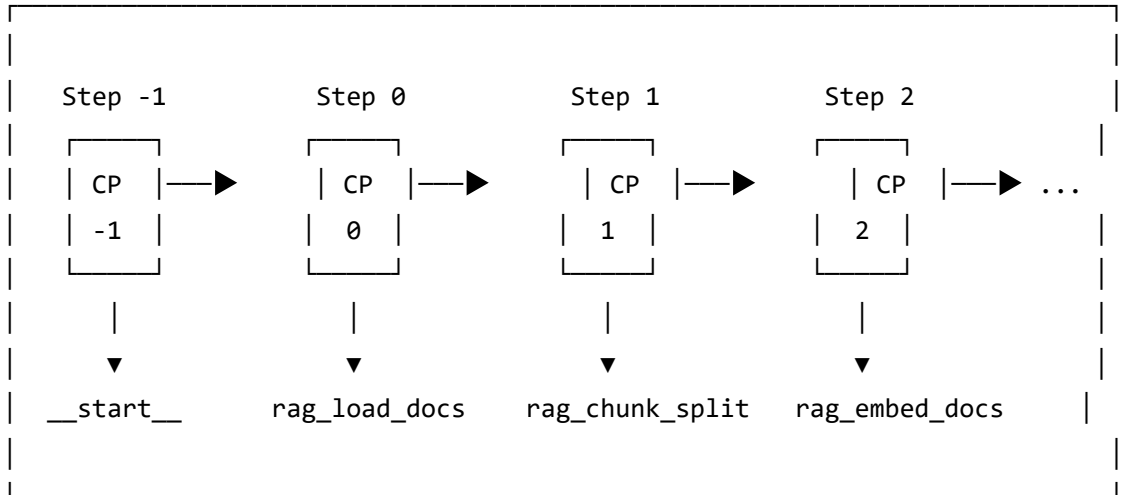


2. Checkpointing Flow

How checkpoints are created and stored at each node boundary.

CHECKPOINTING FLOW

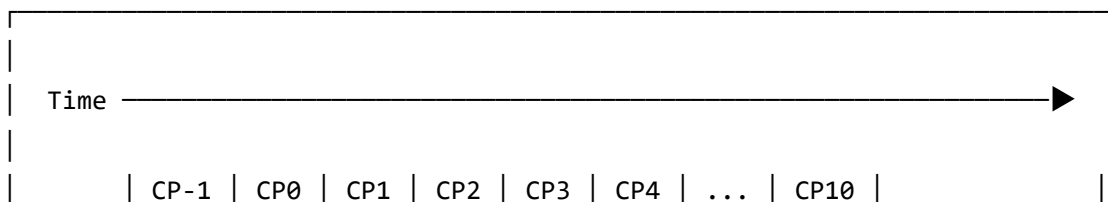
LangGraph Super-Step

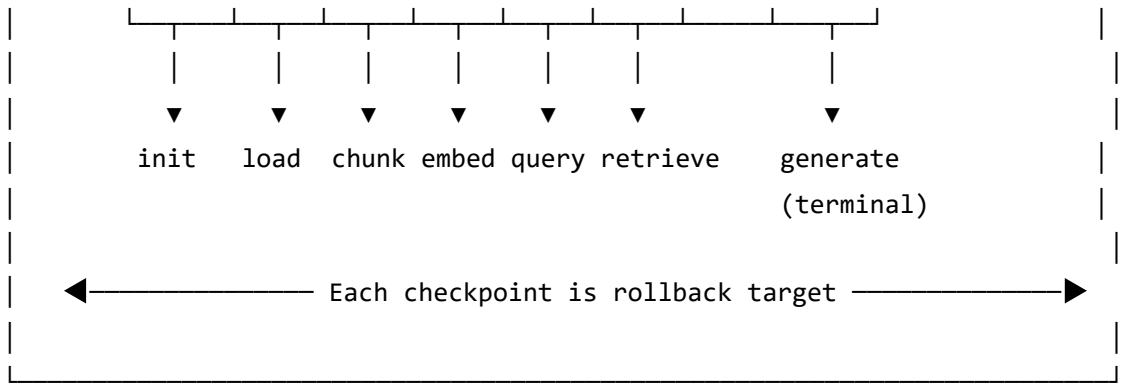


Checkpoint Structure (SQLite: wtb_checkpoints.db)

```
checkpoint_id: 1f0f34e8-164f-6fe7-800a-... (UUID)
thread_id:    wtb-{execution_id}
step:        10
metadata: {
  "source": "loop",
  "writes": {...},
  "step": 10
}
values: {
  "query": "What is revenue?",
  "answer": "Revenue is...",
  "_output_files": {"result.txt": "..."}
}
```

Checkpoint Timeline View

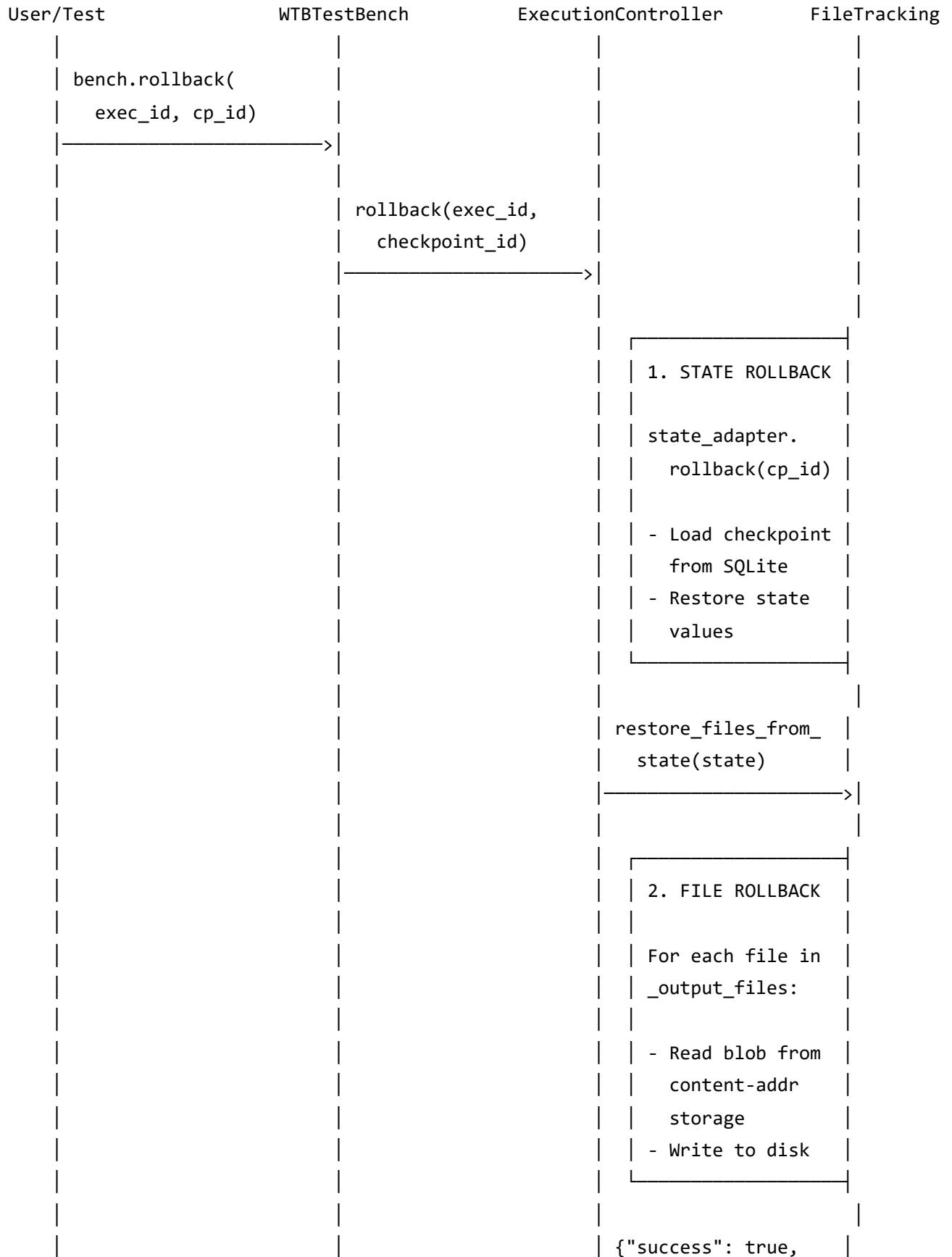


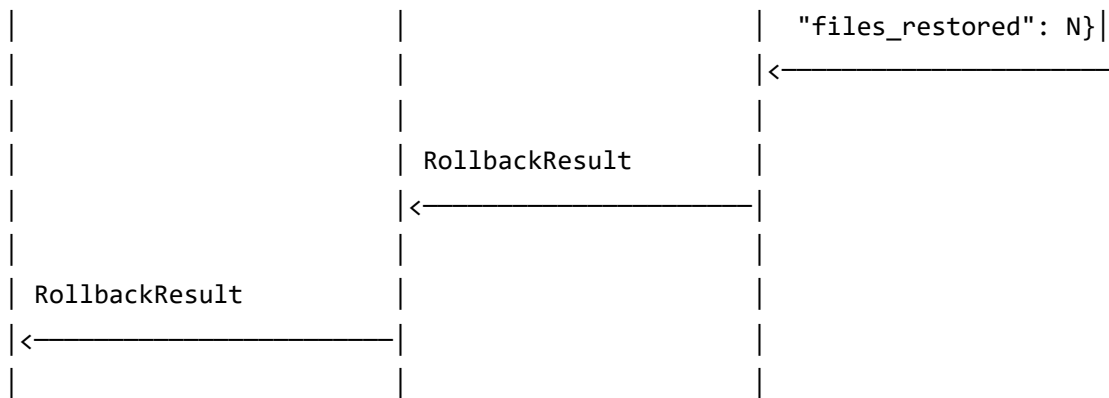


3. Rollback Flow

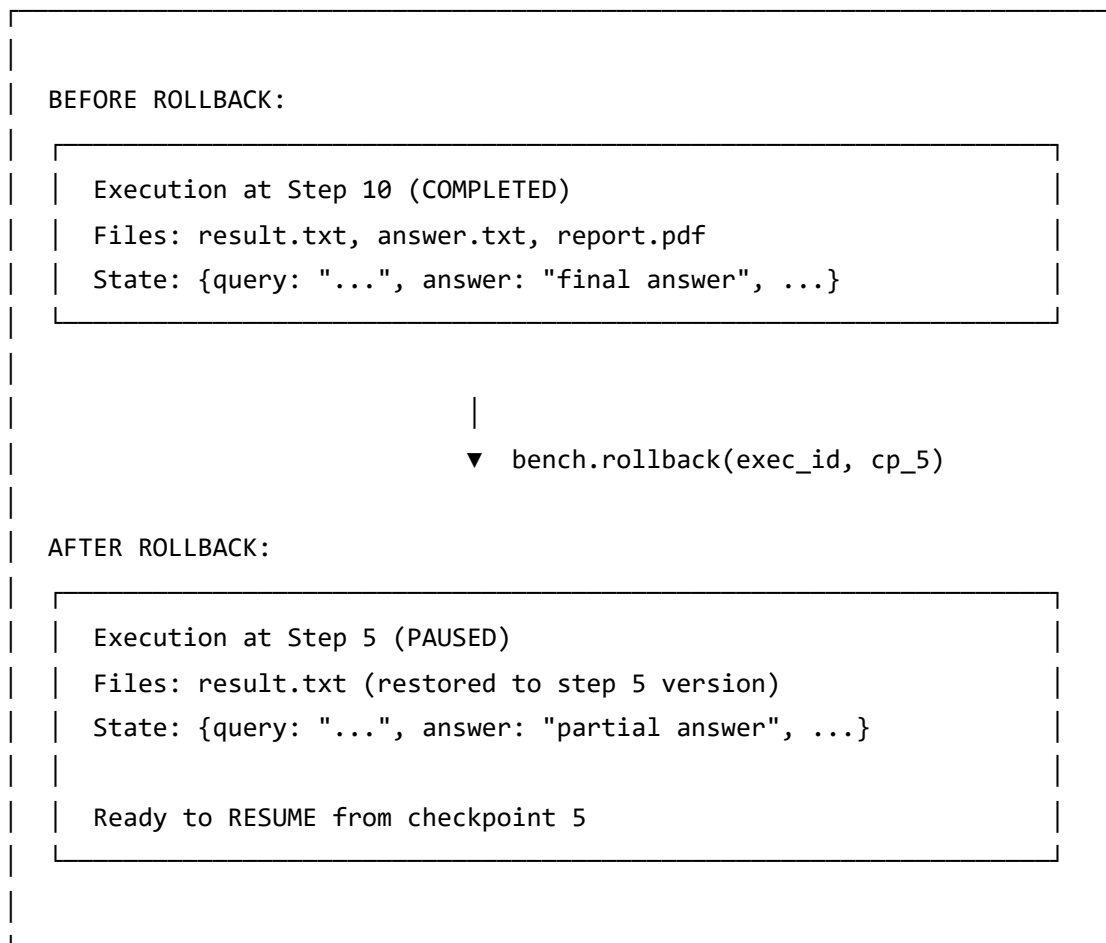
How rollback restores both workflow state AND file system state.

ROLLBACK FLOW





Rollback State Transition

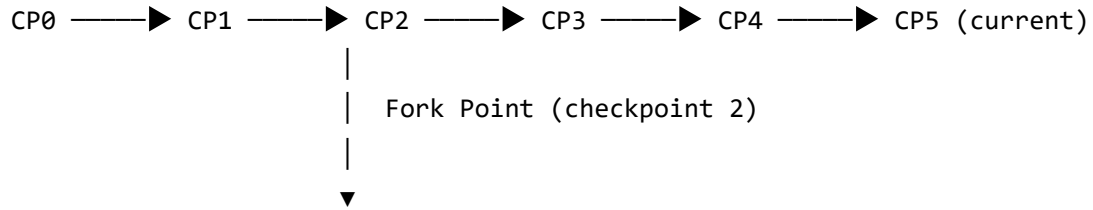


4. Fork/Branch Flow

How branching creates independent execution copies for A/B testing.

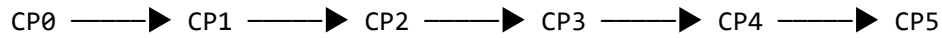
FORK/BRANCH FLOW

Original Execution Timeline

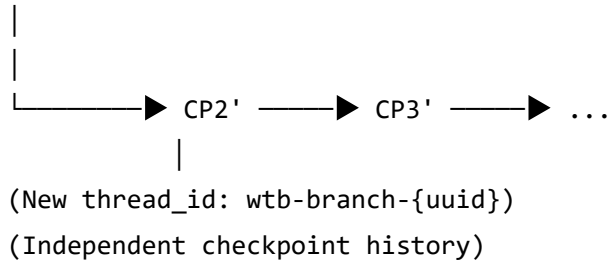


After `bench.fork(execution_id, checkpoint_id="CP2")`

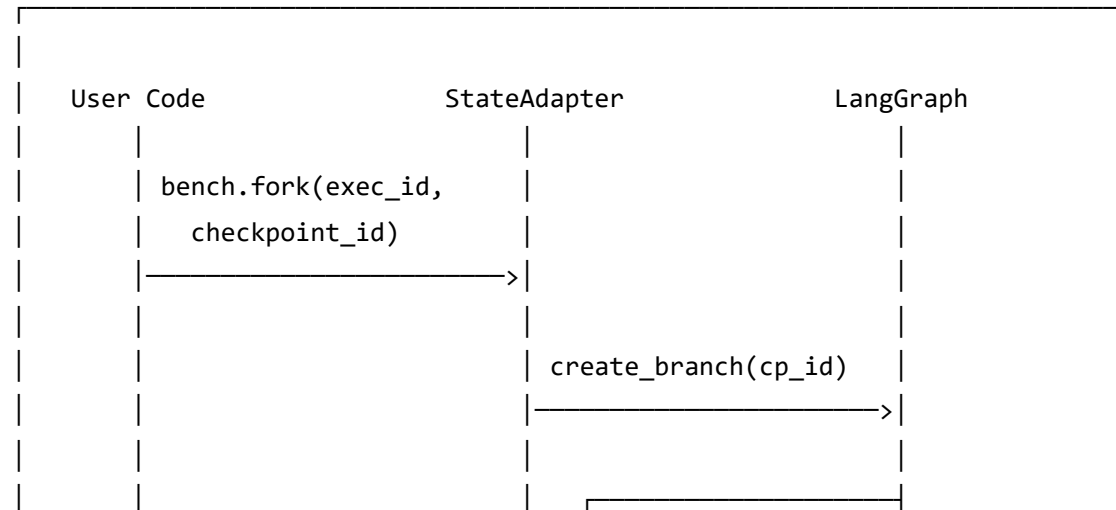
Original (exec-A):

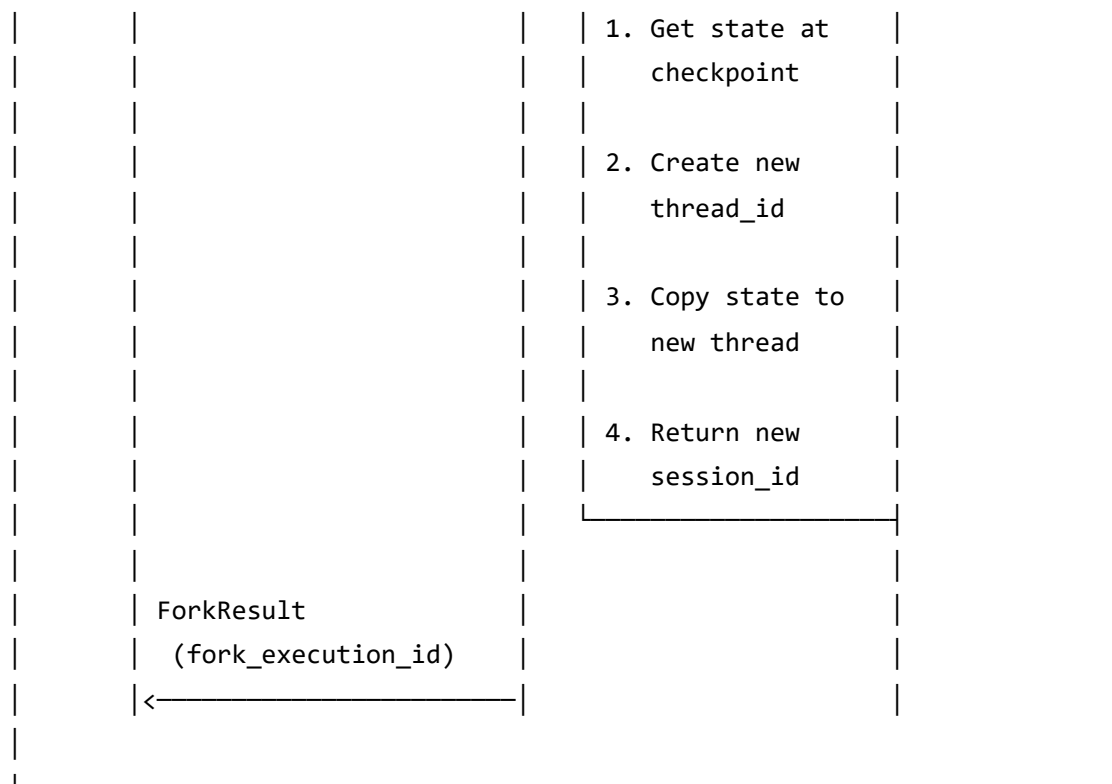


Fork (exec-B):

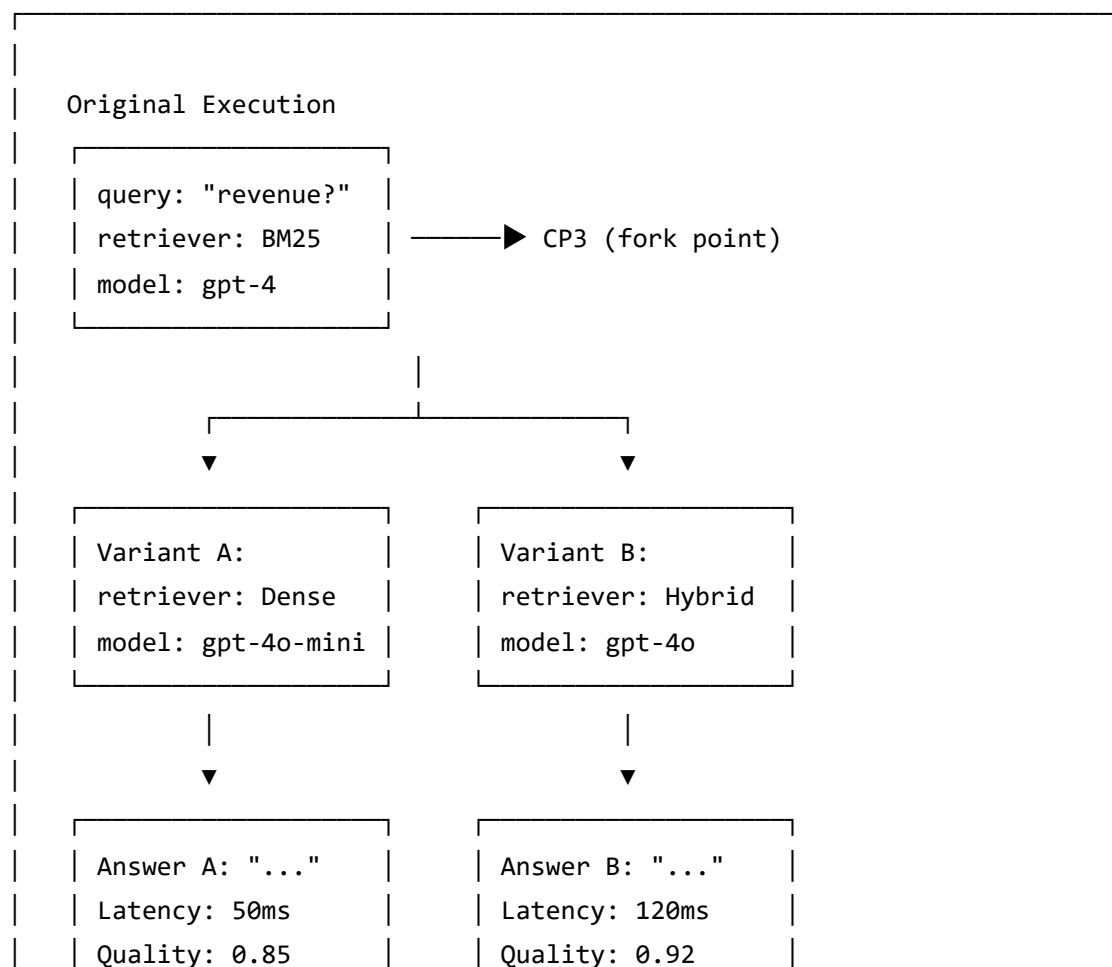


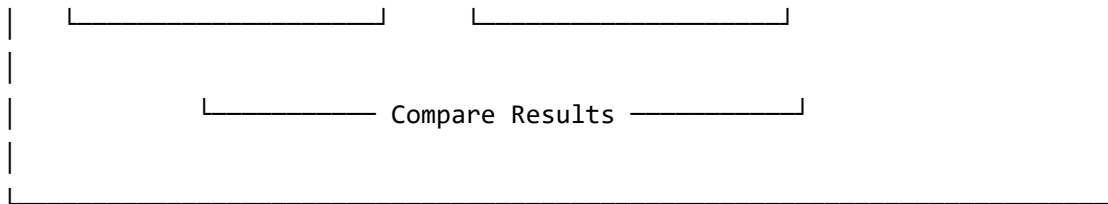
Fork Implementation Detail





A/B Testing Use Case



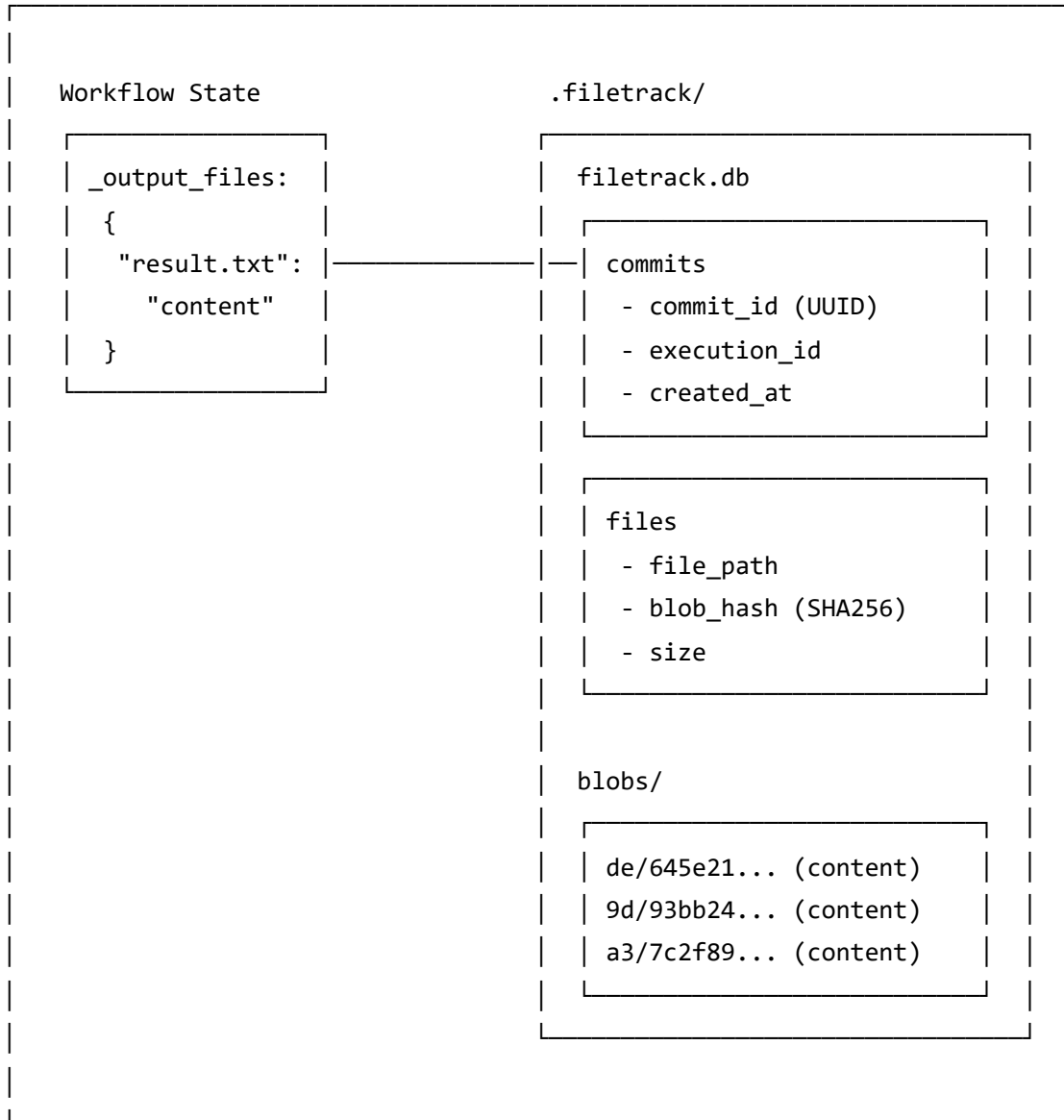


5. File Tracking Flow

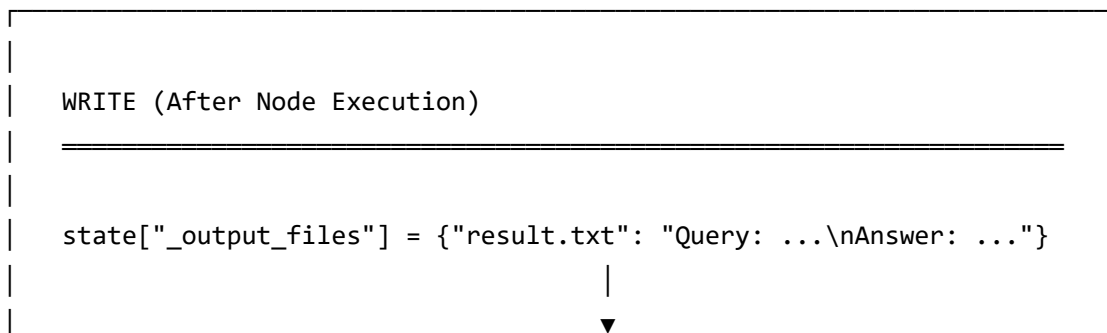
How file tracking enables atomic file system rollback using content-addressable storage.

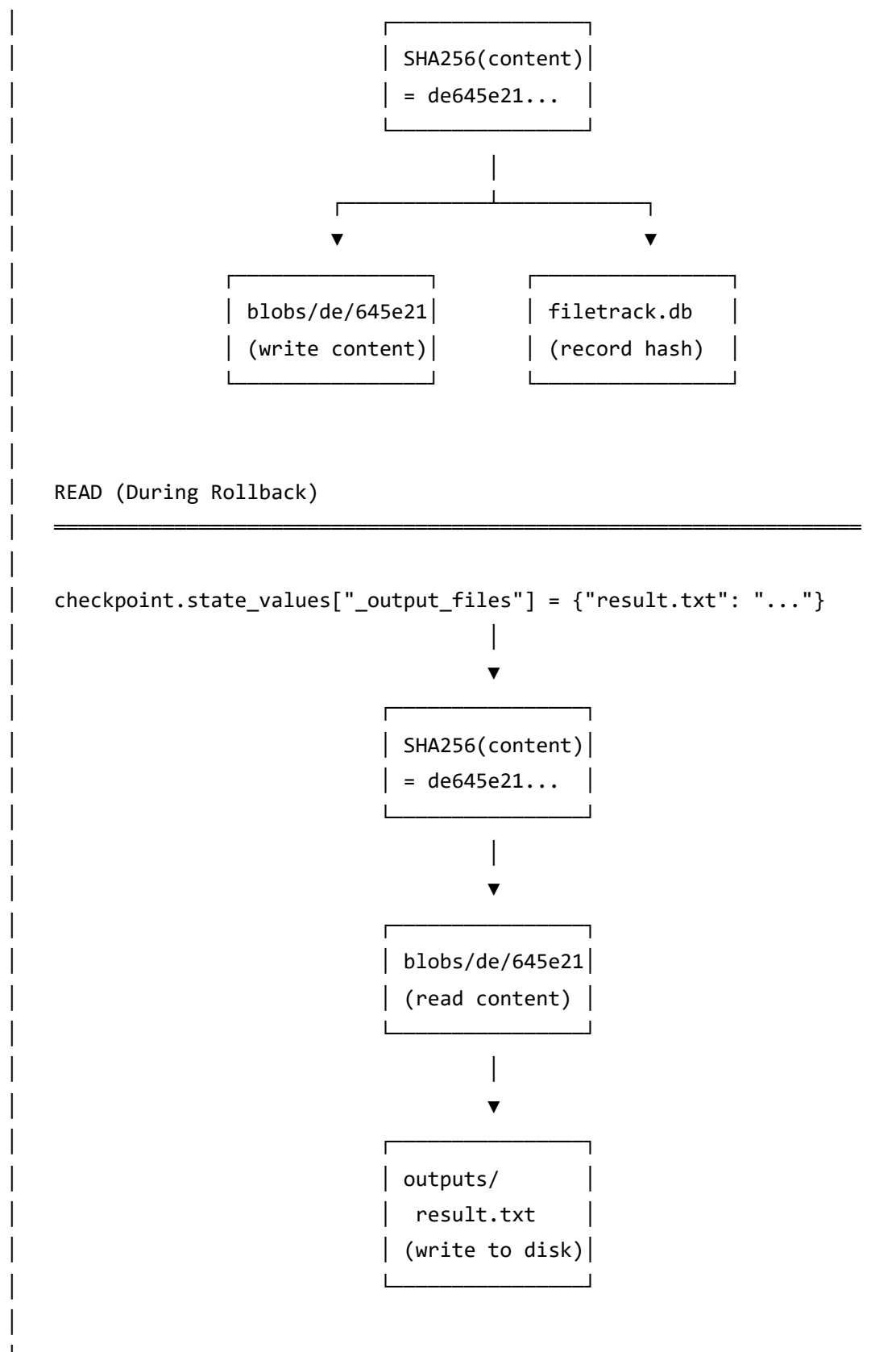
FILE TRACKING FLOW

File Tracking Architecture



Content-Addressable Storage (CAS) Flow





Deduplication Benefit

Execution 1: Creates "result.txt" with hash=abc123
Execution 2: Creates "result.txt" with hash=abc123 (same content)
Execution 3: Creates "result.txt" with hash=def456 (different)

Blob Storage:

blobs/ab/c123... (1 copy)	◀ Exec 1 & 2 share this
blobs/de/f456... (1 copy)	◀ Exec 3 unique

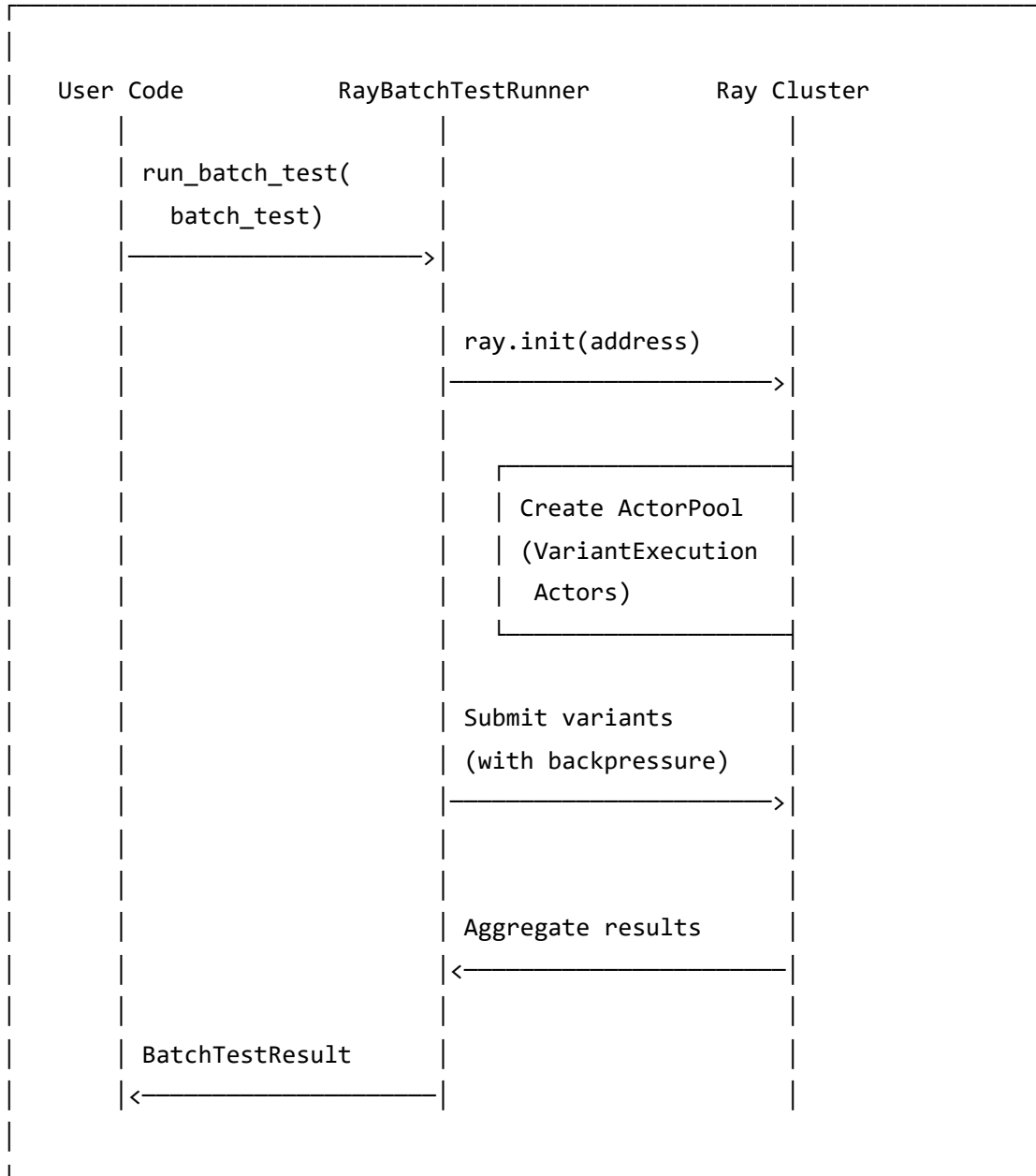
Storage: 2 blobs instead of 3 = 33% savings

6. Ray Batch Execution Flow

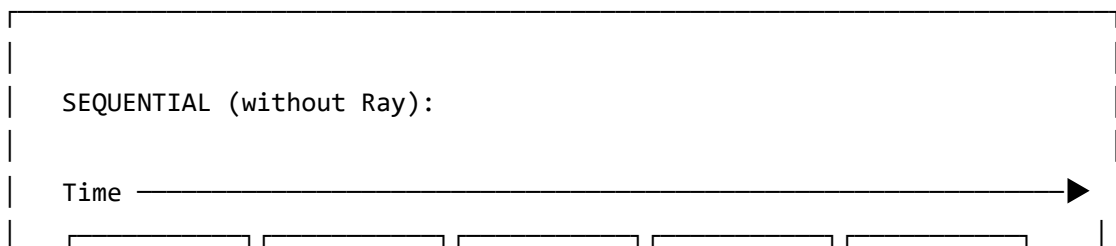
How Ray enables parallel execution of multiple workflow variants.

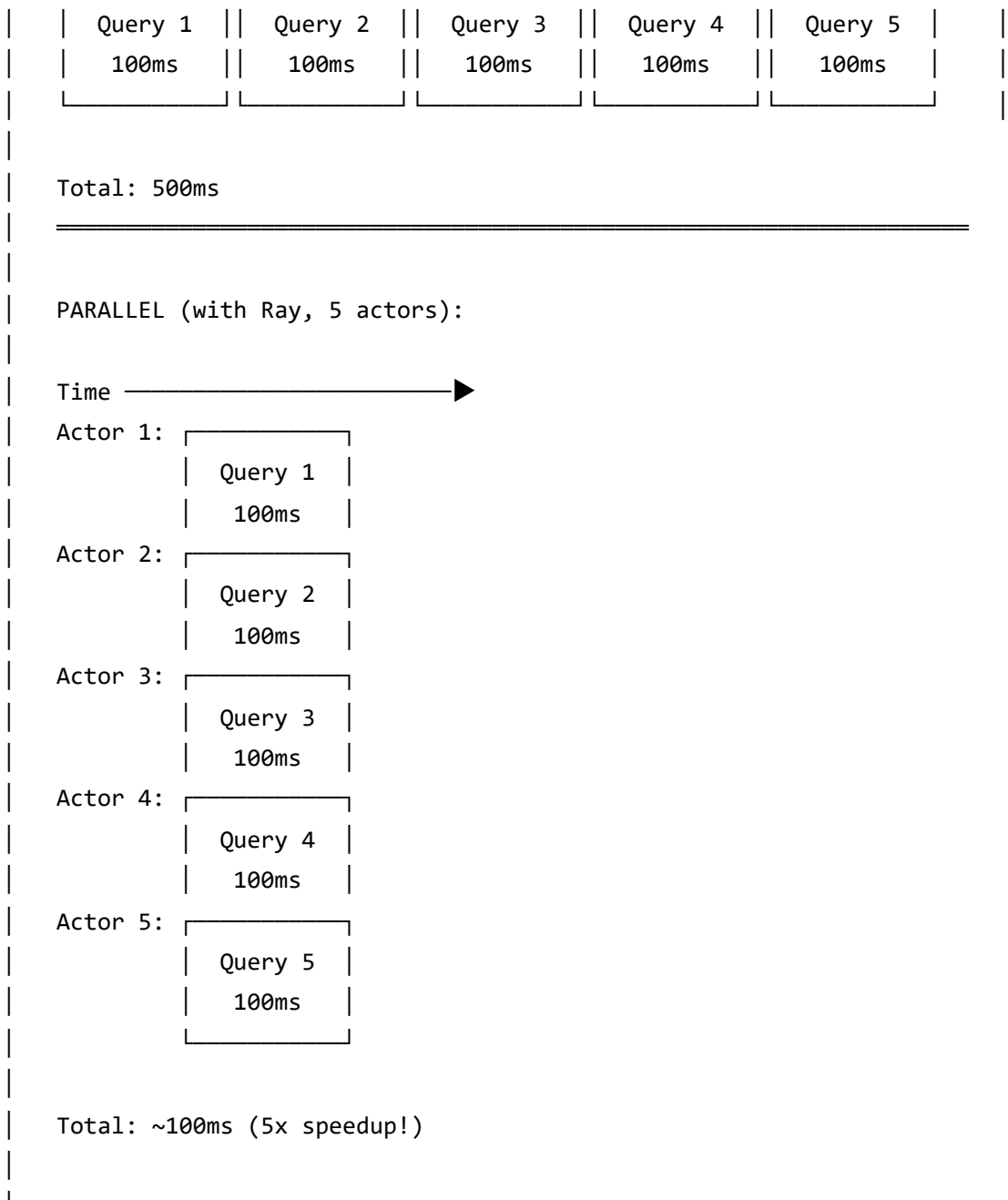
RAY BATCH EXECUTION FLOW

Batch Test Submission

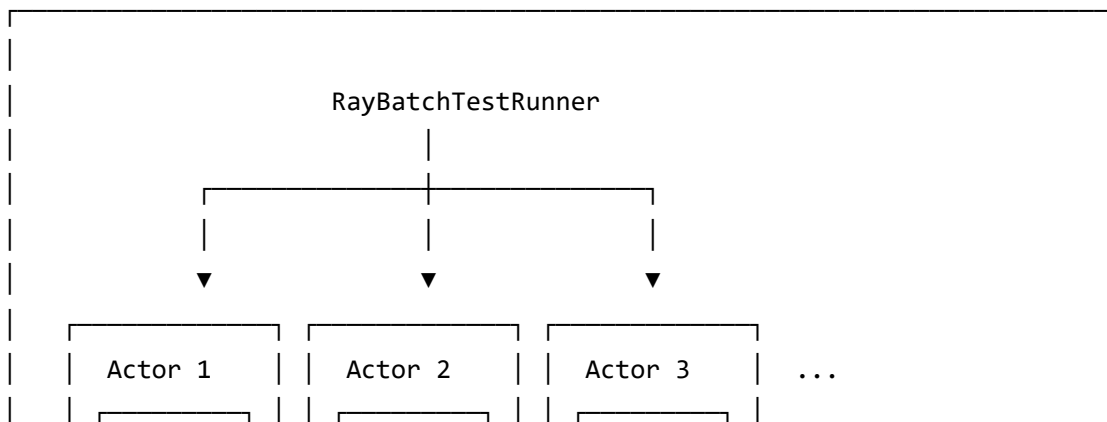


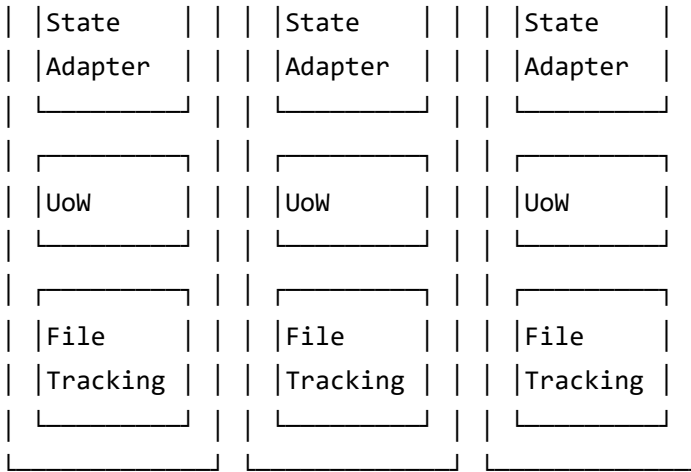
Parallel Execution Timeline





Actor Pool Architecture





Each actor has isolated:

- Database connections (connection pooling per actor)
- State adapter instances
- File tracking service
- Workspace isolation (optional)

Resource Allocation per Node

```
node_resources = {
    "rag_embed_docs": NodeResourceConfig(num_cpus=2, memory="2GB"),
    "rag_retrieve": NodeResourceConfig(num_cpus=2, memory="1GB"),
    "rag_generate": NodeResourceConfig(num_cpus=1, memory="1GB"),
}
```

Ray Scheduler:

Available: 32 CPUs, 64GB RAM

embed_1	embed_2	retrieve_1	...
2CPU/2GB	2CPU/2GB	2CPU/1GB	

Ray automatically schedules tasks based on:

- Resource requirements
- Available cluster capacity

	- Locality preferences	

7. Component Overview

High-level WTB architecture showing all components and their relationships.

WTB COMPONENT ARCHITECTURE

USER / TEST CODE

```
bench = WTBTestBench.create(mode="development", data_dir="./data")
result = bench.run(project="my_workflow", initial_state={...})
bench.rollback(execution_id, checkpoint_id)
bench.fork(execution_id, checkpoint_id)
```



SDK LAYER

WTBTestBench

- run()
- rollback()
- fork()
- get_checkpoints()
- get_state()

WorkflowProject

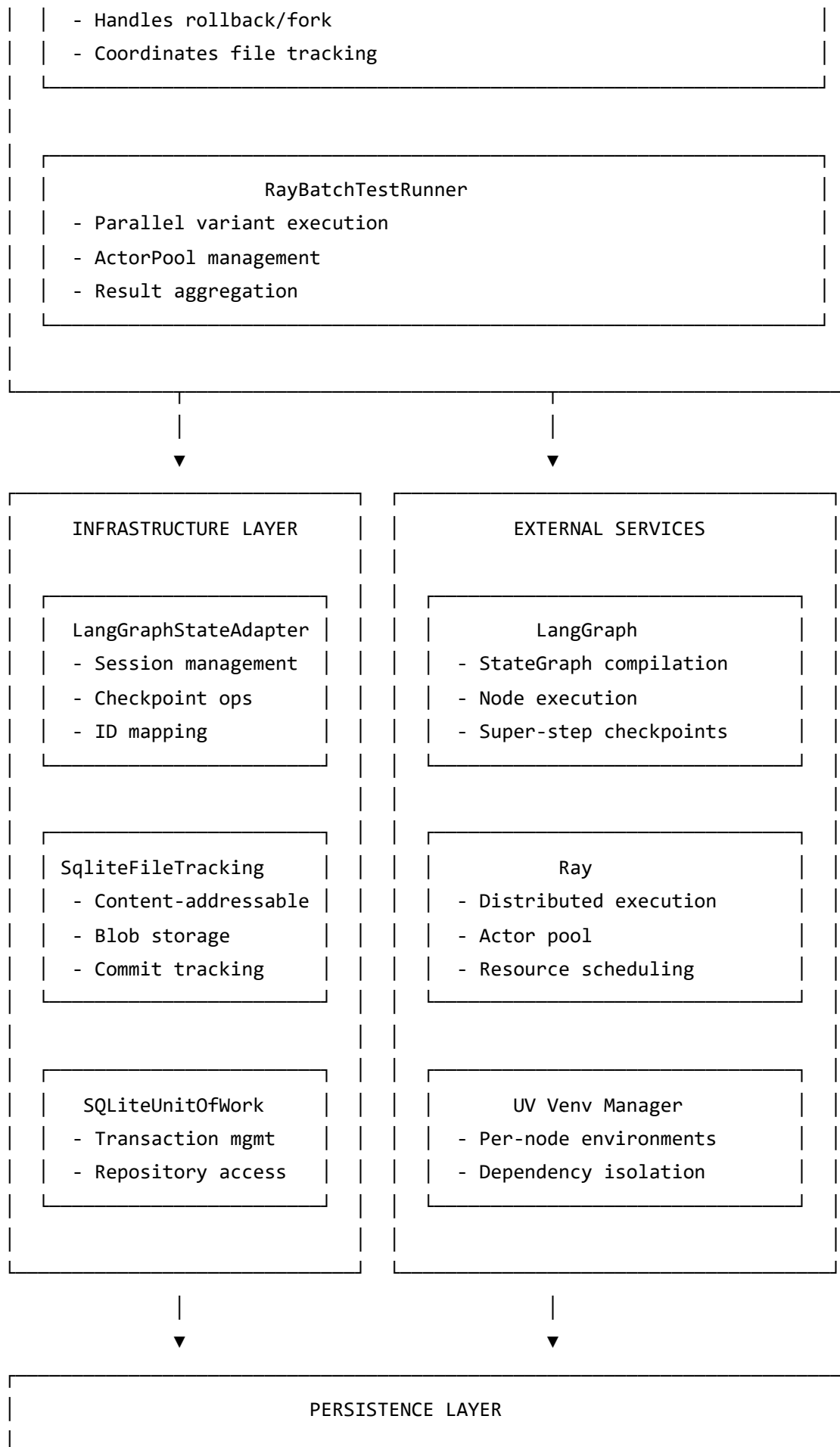
- FileTrackingConfig
- EnvironmentConfig (venv per node)
- ExecutionConfig (Ray, resources)
- PauseStrategyConfig



APPLICATION LAYER

ExecutionController

- Orchestrates workflow execution
- Manages checkpoints



wtb.db	wtb_checkpoints	.filetrack/
	.db	filetrack.db
- executions		blobs/
- workflows	- checkpoints	de/645e21...
- batch_tests	- thread_id	9d/93bb24...
- outbox	- step	
	- values	

Summary

Action	Key Components	Data Flow
Execute	WTBTestBench → ExecutionController → LangGraph	State flows through nodes, auto-checkpoints
Checkpoint	LangGraph → Checkpointer → SQLite	Thread-scoped, step-indexed snapshots
Rollback	StateAdapter → FileTracking	Restore state + files atomically
Fork	StateAdapter.create_branch	New thread_id with copied state
File Track	SqliteFileTrackingService	Content-addressed dedup storage
Ray Batch	RayBatchTestRunner → ActorPool	Parallel isolated executions

Generated: 2026-01-17

WTB Version: 0.1.0