

Entries	WritebackCache	CachedCommittedData	AuthorityStore
main crates/sui-node/src/main.rs		UncommittedData	
match sui_node::SuiNode::start_async		crates/sui-core/src/execution_cache/writeback_cache.rs	
start_csync Crates/sui-node/src/lib.rs let committee_store = Arc::new(CommitteeStore::new(pruner_db = Some(Arc::new(AuthorityPrunerTables::open(let perpetual_tables = Arc::new(AuthorityPrunerTables::open(let checkpoint_store = CheckpointStore::new(&config.db_path().join["checkpoints"])-AuthorityStore::open(perpetual_tables, &genesis, &config. &prunetheus_registry).await?; let cache_traits = build_execution_cache(let state = AuthorityStore:new(WritebackCache crates/sul-core/src/execution_cache/writeback_cache.rs dirty: UncommittedData, cached: CachedCommittedData, packages: MokaCache <objectid, packageobject="">, object_locks: ObjectLocks, executed_effects_digests_notify_read:</objectid,>	objects: DashMap <objectid, cachedversionmap<objectentry="">>, markers: DashMap<markerkey, cachedversionmap<markervalue="">>, transaction_effects: DashMap<fransactioneffectsdigest, dashmap<fransactiondigest,="" transaction_events:="" transactioneffectss,="" transactionevents="">, executed_effects, digests: DashMap<fransactiondigest, transactionevents="">, pending_transaction_writes: DashMap<fransactiondigest, arc<fransactionoutputs="">>, total_transaction_inserts: AtomicU64, total_transaction_inserts: AtomicU64, CachedCommittedData crates/Sui-core/src/execution_cache/writeback_cache.rs</fransactiondigest,></fransactiondigest,></fransactioneffectsdigest,></markerkey,></objectid,>	
build_execution_cache crates/sui-core/src/execution_cache.rs		object_cache: MokaCache <objectid, arc<mutex<cachedversionmap<objectentry="">>>>, object_by_id_cache: MonotonicCache<objectid,< td=""><td></td></objectid,<></objectid,>	
let execution_cache_metrics = Arc::new(ExecutionCacheMetrics::new(prometheus_registry)); ExecutionCacheTraitPointers::new(LatestObjectCacheEntry>, marker_cache: MokaCache <markerkey, arc<multex<cachedversionmap<markervalue="">>>>, transactions: MonotonicCache<fransactiondigest, pointcacheltem<arc<verifiedtransaction="">>>, transaction effects:MonotonicCache<transactioneffectsdiaest,< td=""><td></td></transactioneffectsdiaest,<></fransactiondigest,></markerkey,>	
ExecutionCacheTraitPointers Crates/sul-core/src/execution_cache.rs		Itanisacijon terrecis, indrodnic cache i masacijon terrecis, indrodnic cache i masacijon terrecis, i masacijon	
pub object_cache_reader: Arc <dyn objectcacheread="">, pub transaction_cache_reader: Arc<dyn transactioncacheread="">, pub cache_writer: Arc<dyn executioncachewrite="">,</dyn></dyn></dyn>		executed_effects_digests:MonotonicCache <transactiondigest, pointcacheltem<transactioneffectsdigest="">>, _transaction_objects: MokaCache<transactiondigest, vec<object="">>,</transactiondigest,></transactiondigest,>	
pub backing_store: Arc <dyn +="" backingstore="" send="" sync="">, pub backing_package_store: Arc<dyn +="" backingpackagestore="" send="" sync="">, pub backing_package_store: Arc<dyn +="" objectstore="" send="" sync="">, pub reconfig_api: Arc<dyn executioncachereconfigapi="">, pub accumulator store: Arc<dyn accumulatorstore="">, pub checkpoint_cache: Arc<dyn checkpointcache="">, pub cache: Arc<dyn checkpointcache="">, pub cache: Arc<dyn executioncachecommit="">, pub testing_api: Arc<dyn testingapi="">,</dyn></dyn></dyn></dyn></dyn></dyn></dyn></dyn></dyn>	store: Arc <authoritystore>, backpressure_threshold: u64, backpressure_manager: Arc<backpressuremanager>, metrics: Arc<executioncachemetrics>,</executioncachemetrics></backpressuremanager></authoritystore>		AuthorityStore crates/sui-core/src/authority/authority_store.rs mutex_table: MutexTable <objectdigest>, pub[crate] perpetual_tables: Arc>AuthorityPerpetualTables>, root_state_notify_read: NotifyRead<epochld. (checkpointsequencenumber.="" accumulator)="">, enable_epoch_sui_conservation_check: bool, metrics: AuthorityStoreMetrics,</epochld.></objectdigest>
revert_state_update crates/sul-core/src/execution_cache.rs	revert_state_update_impl crates/sui-core/src/execution_cache/writeback_cache.rs let Some([_ outputs]) = self.dirty.pending_transaction_writes.remove(tx) else { for (object_id. object) in outputs.written.iter() { self.packages.invalidate(object_id); self.cached.object_by_id_cache.invalidate(object_id); self.cached.object_cache.invalidate(object_id); for ObjectKey(object_id) in outputs.deleted.iter().chain(outputs.wrapped.iter(self.cached.object_by_id_cache.invalidate(object_id); self.cached.object_cache.invalidate(object_id); self.cached.object_cache.invalidate(object_id);		AuthorityPerpetualTables
self.revert_state_update_impl(digest)		70) {	pub(crate) objects: DBMap <objectkey, storeobjectwrapper="">, pub(crate) transactions: DBMap<transactiondigest, dbmap<transactioneffectsdigest,="" effects:="" event="" pub(crate)="" transactioneffectsdigest,="" trustedfransactionpices,="" usize),="">, pub(crate) events: DBMap<(TransactionEventsDigest, usize), Event>, pub(crate) expected_network_sul_amount: DBMap<(), u64>,</transactiondigest,></objectkey,>