

Entries	block_stm	aptos_vm	move-vm
execute_and_update_state execution/executor/src/block executor/mod.rs		execute_block aptos-move/aptos-vm/src/aptos_vm.rs	execute_user_transaction aptos-move/aptos-vm/src/aptos_mv.rs
self.inner.execute_and_update_state(		self.execute_block_with_config(txn_provider, state_view, config, transaction_slice_metadata)	match self.execute_user_transaction_with_custom_gas_meter(
execute_and_update_state execution/executor/src/block_executor/mod.rs		execute_block_with_config aptos-move/aptos-vm/src/aptos_vm.rs	execute_user_transaction_with_custom_gas_meter aptos-move/aptos-vm/src/aptos_mv.rs
DoGetExecutionOutput::by_transaction_execution(		let result = AptosVMBlockExecutorWrapper::execute_block::<	let (status, output) = self.execute_user_transaction_impl(
by_transaction_execution execution/executor/src/workflow/do_get_execution_output rs		execute_block aptos-move/aptos-vm/src/block_executor/mod.rs	execute_user_transaction_impl aptos-move/aptos-vm/src/aptos mv.rs
let out = match transactions {     ExecutableTransactions::Unsharded(txns) => {         Self::by_transaction_execution_unsharded:: <v>{         ExecutableTransactions::Sharded(txns) =&gt;          Self::by_transaction.execution_sharded::<v>{</v></v>		Self::execute_block_on_thread_pool:: <s, l,="" tp="">(</s,>	payload @ TransactionPayload::EntryFunction(_) => self
		execute_block_on_thread_pool aptos-move/aptos-vm/src/block_executor/mod.rs	execute_script_or_entry_function
by_transaction_execution_unsharded		let executor = BlockExecutor:: <signatureverifiedtransaction, e,="" l,="" s,="" tp="">::new let ret = executor.execute_block(</signatureverifiedtransaction,>	Pantos-move/antos-vm/src/antos my rs
execution/executor/src/workflow/do_get_execution_output rs let block_output = Self::execute_block:: <v>{</v>	execute_block aptos-move/block-executor/src/executor.rs		validate_and_execute_entry_function aptos-move/aptos-vm/src/aptos mv.rs
py_transaction_execution_sharded execution/executor/src/workflow/do_get_execution_output_rs	if self.config.local.concurrency_level > 1 {     let parallel_result = self.execute_transactions_parallel(		session.execute_entry_function(
et transaction_outputs = Self::execute_block_sharded:: <v>(</v>	let sequential_result = self.execute_transactions_sequential(		execute_entry_function aptos-move/aptos-vm/src/move_vm_ext/session/mod.r.
execute_block execution/executor/src/workflow/do_get_execution_output.rs	execute_transactions_parallel aptos-move/block-executor/src/executor.rs	execute	MoveVM::execute_loaded_function(
Dk(executor.execute_block(	self.executor_thread_pool.scope( s  { for _ in 0num_workers {	aptos-move/block-executor/src/executor.rs	execute_loaded_function third_party/move/move-vm/runtime/src/move_vm.rs
execute_block_sharded execution/executor/src/workflow/do_get_execution_output_rs	if let Err(err) = self.worker_loop(	txn, idx_to_execute);	Interpreter::entrypoint(
Dk(executor.execute_block(	worker_loop aptos-move/block-executor/src/executor.rs	execute_transaction aptos-move/aptos-vm/src/block_executor/vm_wraper.rs	entrypoint third_party/move/move-vm/runtime/src/interpreter.rs
	scheduler_task = match scheduler_task { SchedulerTask:\ValidationTask(tkn_idx, incarnation, wave) => { let valid = Self:\validate( Self::\update_on_validation( SchedulerTask:\update_cutionTask() let needs_suffix_validation = Self::\update_cute( SchedulerTask:\update_cutionTask() ExecutionTask() ExecutionTask()_pe:\update_cute(condvar)) => { scheduler.next_task() SchedulerTask:\update_cute(condvar), \update_cute() SchedulerTask:\update_cute() SchedulerTask:\update_cute() SchedulerTask:\update_cute() SchedulerTask:\update() Sc	.vm.execute_single_transaction(txn, &resolver, view, &log_context)	InterpreterImpl::entrypoint(
		execute_single_transaction aptos-move/aptos-vm/src/aptos_mv.rs	entrypoint third_party/move/move-vm/runtime/src/interpreter.rs
		self.execute_user_transaction(resolver, code_storage, txn, log_context);	interpreter.dispatch_execute_main:: <noruntimetypecheck>(</noruntimetypecheck>
			execute_main third_party/move/move-vm/runtime/src/interpreter.rs
			self.call_native:: <rttcheck, rtcaches="">(</rttcheck,>