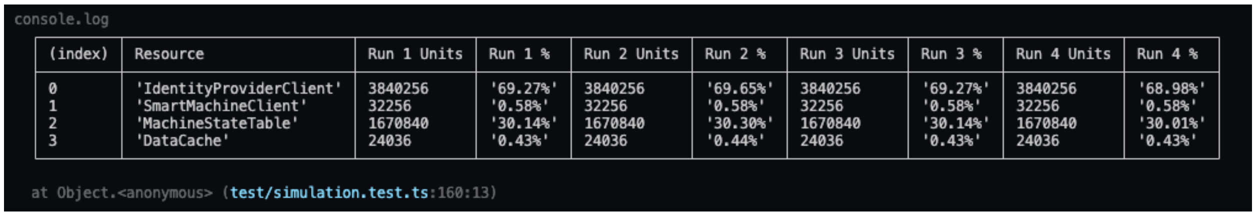
System Design Component Implementation

Report

All tests completed successfully. The test suite ran 12 tests across 2 test suites, with all tests passing. Total execution time was 0.89 seconds. The testing included both API functionality tests and system simulation tests.

Resource Usage Analysis

The simulation ran four separate iterations to measure resource consumption patterns. Resource usage remained consistent across all runs, which indicates stable behavior under the test conditions.



```
console.log
  ┌─────────┬───────────────────────────┬─────────────┬──────────┬─────────────┬──────────┬─────────────┬──────────┬─────────────┬──────────┐
  │ (index) │ Resource                  │ Run 1 Units │ Run 1 %  │ Run 2 Units │ Run 2 %  │ Run 3 Units │ Run 3 %  │ Run 4 Units │ Run 4 %  │
  ├─────────┼───────────────────────────┼─────────────┼──────────┼─────────────┼──────────┼─────────────┼──────────┼─────────────┼──────────┤
  │ 0       │ 'IdentityProviderClient'  │ 3840256     │ '69.27%' │ 3840256     │ '69.65%' │ 3840256     │ '69.27%' │ 3840256     │ '68.98%' │
  │ 1       │ 'SmartMachineClient'      │ 32256       │ '0.58%'  │ 32256       │ '0.58%'  │ 32256       │ '0.58%'  │ 32256       │ '0.58%'  │
  │ 2       │ 'MachineStateTable'       │ 1670840     │ '30.14%' │ 1670840     │ '30.30%' │ 1670840     │ '30.14%' │ 1670840     │ '30.01%' │
  │ 3       │ 'DataCache'               │ 24036       │ '0.43%'  │ 24036       │ '0.44%'  │ 24036       │ '0.43%'  │ 24036       │ '0.43%'  │
  └─────────┴───────────────────────────┴─────────────┴──────────┴─────────────┴──────────┴─────────────┴──────────┴─────────────┴──────────┘

    at Object.<anonymous> (test/simulation.test.ts:160:13)
```

IdentityProviderClient consumed the most resources, accounting for approximately 69% of total units across all four runs. The values ranged from 68.98% to 69.65%, showing minimal variation between runs. This component used 3,840,256 units in each test iteration.

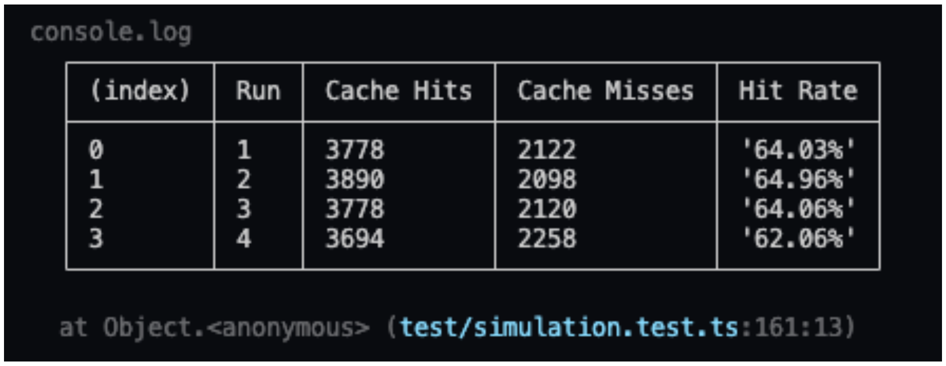
MachineStateTable was the second largest consumer at roughly 30% of total resources. Usage ranged from 30.01% to 30.30% across the four runs, with each run consuming 1,670,840 units. This consistency suggests predictable resource requirements.

SmartMachineClient and DataCache both showed minimal resource usage. SmartMachineClient maintained 0.58% usage (32,256 units) across all runs. DataCache consumed between 0.43% and 0.44% of resources (24,036 units per run).

Cache Performance Metrics

Cache hit rates varied slightly across the four simulation runs. Run 2 achieved the highest hit rate at 64.96% (3,890 hits, 2,098 misses). Run 4 showed the lowest hit rate at 62.06% (3,694 hits, 2,258 misses). Runs 1 and 3 performed similarly with hit rates of 64.03% and 64.06% respectively.

The variation in cache performance between runs was relatively small, spanning just under 3 percentage points. This suggests that while cache behavior is somewhat variable, it remains within a consistent range under similar conditions.



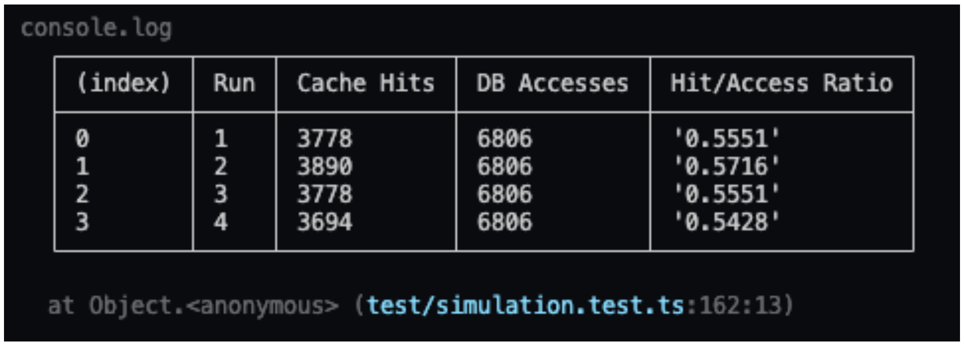
```
console.log
┌─────────┬─────┬────────────┬──────────────┬──────────┐
│ (index) │ Run │ Cache Hits │ Cache Misses │ Hit Rate │
├─────────┼─────┼────────────┼──────────────┼──────────┤
│ 0       │ 1   │ 3778       │ 2122         │ '64.03%' │
│ 1       │ 2   │ 3890       │ 2098         │ '64.96%' │
│ 2       │ 3   │ 3778       │ 2120         │ '64.06%' │
│ 3       │ 4   │ 3694       │ 2258         │ '62.06%' │
└─────────┴─────┴────────────┴──────────────┴──────────┘

  at Object.<anonymous> (test/simulation.test.ts:161:13)
```

Database Access Patterns

Database access count remained constant at 6,806 operations across all four runs. This consistency indicates that the test workload was identical in each iteration.

The ratio of cache hits to database accesses ranged from 0.5428 to 0.5716. Run 2 showed the best ratio at 0.5716, meaning that for every database access, there were approximately 0.57 cache hits. Run 4 had the lowest ratio at 0.5428.

These ratios indicate that cache hits occur at roughly half the rate of database accesses. The relatively tight clustering of these values across runs suggests stable system behavior.



```
console.log
┌─────────┬─────┬────────────┬─────────────┬──────────────────┐
│ (index) │ Run │ Cache Hits │ DB Accesses │ Hit/Access Ratio │
├─────────┼─────┼────────────┼─────────────┼──────────────────┤
│ 0       │ 1   │ 3778       │ 6806        │ '0.5551'         │
│ 1       │ 2   │ 3890       │ 6806        │ '0.5716'         │
│ 2       │ 3   │ 3778       │ 6806        │ '0.5551'         │
│ 3       │ 4   │ 3694       │ 6806        │ '0.5428'         │
└─────────┴─────┴────────────┴─────────────┴──────────────────┘

  at Object.<anonymous> (test/simulation.test.ts:162:13)
```

Observations

The test results show consistent system behavior across multiple runs. Resource allocation patterns were stable, with the same components consuming similar percentages of resources in each iteration. The two dominant consumers (IdentityProviderClient and MachineStateTable) together accounted for roughly 99% of all resource usage.

Cache performance showed some variability but remained within a narrow band. The difference between the best and worst cache hit rates was less than 3%, which falls within normal variation for cached systems. The fact that database access counts stayed constant while cache hits varied slightly suggests that the caching layer is working as intended but responds to minor differences in request patterns or timing.

The relationship between cache hits and database accesses remained proportional across all runs. This proportionality, combined with the consistent database access count, indicates that the test workload was uniform and that the system handled it in a repeatable manner.

Conclusion

All tests passed successfully. The system demonstrated consistent performance characteristics across multiple test runs. Resource usage patterns were stable, and cache behavior fell within expected variation ranges. The test results provide a baseline for system performance under the simulated workload conditions.