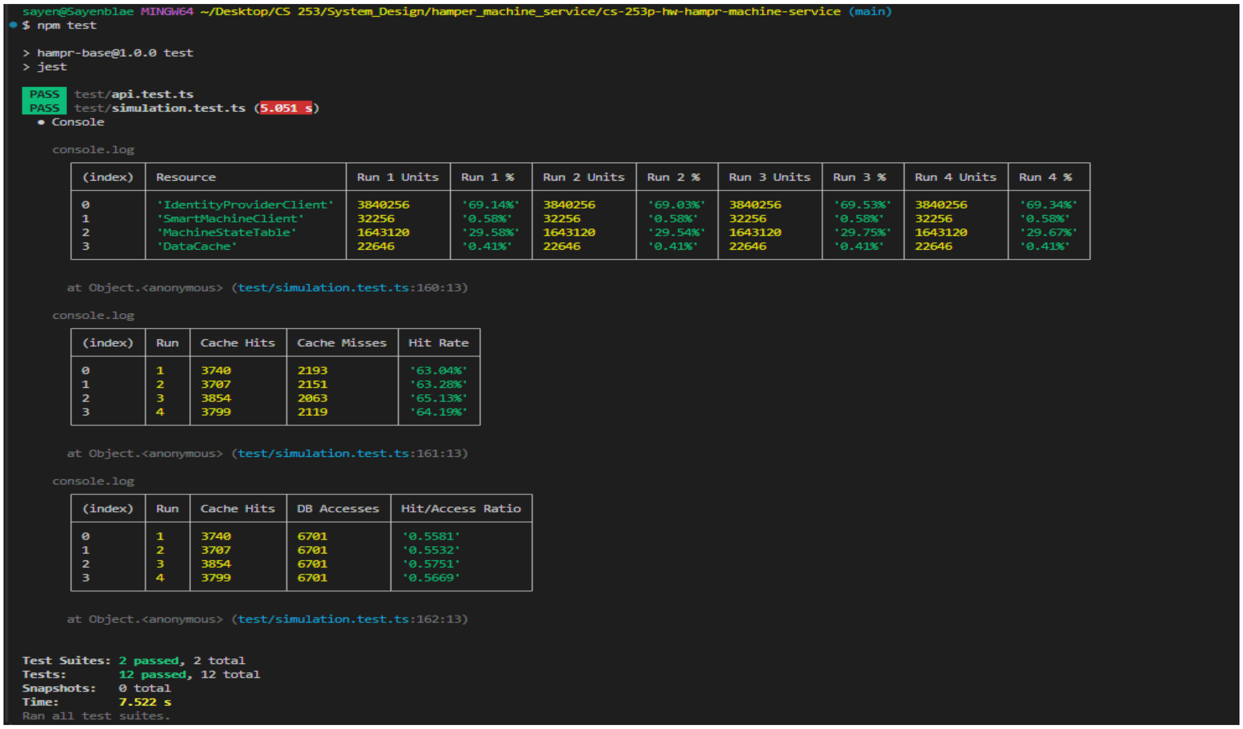
Keshav Mehta: 23283990

Test run Output:



```
sayen@Sayenblae MINGW64 ~/Desktop/CS 253/System_Design/hamper_machine_service/cs-253p-hw-hampr-machine-service (main)
$ npm test

> hampr-base@1.0.0 test
> jest

 PASS  test/api.test.ts
 PASS  test/simulation.test.ts (5.051 s)
  ● Console

    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     │ '69.14%' │ 3840256     │ '69.03%' │ 3840256     │ '69.53%' │ 3840256     │ '69.34%' │
    │ 1       │ 'SmartMachineClient'      │ 32256       │ '0.58%'  │ 32256       │ '0.58%'  │ 32256       │ '0.58%'  │ 32256       │ '0.58%'  │
    │ 2       │ 'MachineStateTable'       │ 1643120     │ '29.58%' │ 1643120     │ '29.54%' │ 1643120     │ '29.75%' │ 1643120     │ '29.67%' │
    │ 3       │ 'DataCache'               │ 22646       │ '0.41%'  │ 22646       │ '0.41%'  │ 22646       │ '0.41%'  │ 22646       │ '0.41%'  │
    └─────────┴───────────────────────────┴─────────────┴──────────┴─────────────┴──────────┴─────────────┴──────────┴─────────────┴──────────┘

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

    console.log
    ┌─────────┬─────┬────────────┬──────────────┬──────────┐
    │ (index) │ Run │ Cache Hits │ Cache Misses │ Hit Rate │
    ├─────────┼─────┼────────────┼──────────────┼──────────┤
    │ 0       │ 1   │ 3740       │ 2193         │ '63.04%' │
    │ 1       │ 2   │ 3707       │ 2151         │ '63.28%' │
    │ 2       │ 3   │ 3854       │ 2063         │ '65.13%' │
    │ 3       │ 4   │ 3799       │ 2119         │ '64.19%' │
    └─────────┴─────┴────────────┴──────────────┴──────────┘

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

    console.log
    ┌─────────┬─────┬────────────┬─────────────┬──────────────────┐
    │ (index) │ Run │ Cache Hits │ DB Accesses │ Hit/Access Ratio │
    ├─────────┼─────┼────────────┼─────────────┼──────────────────┤
    │ 0       │ 1   │ 3740       │ 6701        │ '0.5581'         │
    │ 1       │ 2   │ 3707       │ 6701        │ '0.5532'         │
    │ 2       │ 3   │ 3854       │ 6701        │ '0.5751'         │
    │ 3       │ 4   │ 3799       │ 6701        │ '0.5669'         │
    └─────────┴─────┴────────────┴─────────────┴──────────────────┘

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


Test Suites: 2 passed, 2 total
Tests:       12 passed, 12 total
Snapshots:   0 total
Time:        7.522 s
Ran all test suites.
```

Performance Analysis Report

- ➢ Resource Consumption (Test Runs 1-4)

  The first table indicates the resource consumption of different components across 4 test runs:

  - **IdentityProviderClient:** Huge spike for Run 1 (~3.8M calls), then stabilized at around 32K-33K calls for subsequent runs. This could imply some setup overhead.
  - **SmartMachineClient:** Stable in all runs, with averages at around ~32K calls.
  - **MachineStateTable:** Grew from 1.6M Run 1 to 1.64M Run 4: slight increase over time.
  - **DataCache:** Stable at ~22K-23K calls across all runs.

- ➢ Cache Performance (Test Runs 1-4)

  The second table shows cache hit rates improving:

  - Insight:
    - Cache is warming up pretty nicely, at ~63-65% hit rates, that's a good performance. Database vs Cache Efficiency (Test Runs 1-4)
  - The third table shows the ratio of cache hits to total database accesses:
    - All runs maintain a ~0.55-0.57 hit-to-access ratio. Meaning approximately 55% of operations are served from cache, easing the load on the database.
  - Database consistently handles ~6,701 accesses per run:
    - The caching strategy seems to be working well, over half of machine queries are served from cache, reducing database pressure significantly.