PI-23 Review

Performance Characterisation: Progress Assessment

Jan 2024

Workstream Update

Workstream Name:	Performance Characterisation
Roadmap Pillar:	Foundation: Quality Product
Lead:	James Bush
Workstream Objectives:	See next slides
Progress Against Objectives:	● Position Batching on fulfils with a recharacterization. ● Quotes refactor. ● SDK characterisation with performance improvements. ● Configuring performance dashboards in IaC. ● Setting up performance test framework for IaC.
Anticipated Progress by PI End:	All three transfer stages tested and quick fixes complete.
Roadblocks:	Resource limitations: Committed resources from community and adopters
Support Needed:	Engineers with infra and/or mojaloop core experience. Bare metal testing support.

Workstream Objectives

Q: What problem(s) are we trying to solve?
A: Understand how various components of the system perform individually and collectively on hardware representative of near-term upcoming deployments. What are the system sensitivities to hardware characteristics and scheme architecture (number of transacting participants etc…).

Strategy: Show the system is capable of 1000 tps.
Plan: Discover what the actual number is and push beyond current limits.

1. Priority 1: Find out how fast the system is.
   a. If it doesn't meet the baseline, then take immediate action to rectify.
      i. Target is 1000 tps sustained for 1 hour, < 1% taking > 1 second @ 0% unexpected errors, and durable for 1 hour.
         1. A "transaction" is: an address lookup followed by a quote followed by a transfer.
         2. Consider ramp-up.

Workstream Objectives

Cont…

a. Nation scale deployment is the target customer right now (adoption pipeline); target activities at understanding performance characteristics for this type of deployment.
   - i. Design appropriate scheme architecture(s) to simulate.
     1. Firstly, target national scale switch situations with many DFSPs. (? how many, some big, some small ?)
   - ii. Design appropriate REUSABLE test architecture and tools e.g. simulators, load generators etc… (keep loads separate i.e. sims on separate infra to switch components)
     1. Run this in an environment where we can take underlying platform considerations out of the equation e.g. bare metal; that accurately represents the targeted deployment environment. (simplify)
     2. Tests should be end-to-end:
        - a. What does this mean? ALS? Quote? Transfer?
          - i. It means all three stages! BUT - we have to account for user interface delays, e.g. latency at each end due to test harness latency etc…
   - iii. Consider looking at the same toolchain as used previously for generating load (jMeter clusters).
     1. Try to figure out how to handle callbacks in jMeter to keep the amount of “stuff” in the test harnesses to a minimum.
   - iv. Design appropriate loads/transaction scenario mix (+ve, -ve cases etc…) to test the edges of the system. Include scheme level issues such as liquidity problems etc…
     1. Include some scenarios which test failure modes for certain situations e.g. DFSP goes down, slows down etc… capture the impact of such events on the system as a whole.
     2. How about running settlements during periods of high load?
   - v. Push beyond stable limits to discover failure modes and plan any remediating actions needed.
   - vi. Decide/Design appropriate metrics and points of observation to capture and monitor data for analysis.
     1. Not just application layer metrics, also look at underlying infra metrics, CPU, I/O etc…
     2. Consider how to best inform calculations of cost per transfer, TCO vs procurement costs, maintenance etc... Provide data to support those calculations.

Workstream Objectives

Cont…

1. Priority 2: Introduce performance tests at a lower (?component) level in CI pipelines
   a. Run these tests in CI runners on merge/commit/PR?
      i. CI runners are on shared infra, this might not be good for comparison.
      ii. Possibly run these on foundation infra, possibly have some private CI runners in foundation AWS account?
   b. Generate metrics, reports, charts etc… so all interested parties can see the impact of code changes on performance.
2. Priority 3: Develop capacity to manually run repeatable full system end-to-end performance tests (as per P1 above):
   a. Spin up / down an entire env and test infrastructure, run tests, capture metrics, generate reports.
   b. Analyse the cost of doing this.

Progress so far this PI…

Workstream artefacts in github: https://github.com/mojaloop/ml-perf-characterization

Highlights:

- Position Batching on fulfils with a recharacterization.
- Quotes refactor.
  - Switch to async kafka message based approach as per central ledger
- SDK characterisation with performance improvements.
- Configuring performance dashboards in IaC.
- Setting up performance test framework for IaC.

Reach out to participate

- Email: james.bush@mojaloop.io
- Mojaloop Slack: #perf-scale-benchmarking