Test parallelism

Tom Rochette <tom.rochette@coreteks.org>

August 13, 2025 — 863ffbd2

- Start the longest tests first. If you have for example 2 tests, a short one and a long one, the long one will be the sequential bottleneck. If you have many additional short tests, they will run in parallel to the longer one. If you have 2 workers, 1 will be dedicated to the long task while the other will process many of the short tests. If the total execution time of the short tasks is greater than the longer task then the total execution time will be at least the duration of the long task.
  - If we were not to schedule the longest tests first, we run the risk of running many short tests on a worker and then running the longest test, which may be suboptimal.
  - As you accumulate many short tests this issue is mitigated. However you have to be careful of distributing your longest tasks on separate workers as much as possible.
- If you know the exact duration of a test (or an approximation of it, given a desired quantum), determine if you can solve the scheduling problem in a reasonable amount of time such that solving the scheduling and running the scheduled tests takes less time than running the tests.
- From the standpoint of execution, we will prefer to have the best response time possible (i.e., run the shortest tasks first to get most of the tests executed as soon as possible).
- As you have many short and long tests, it may make sense to run all the short tests first and keep all the long tests for the end, ignoring the heuristics that suggests running the longest tests first.
- Other important consideration when running tests are shared environments/resources between tests. If those take a non-neglectable amount of time to setup/teardown and could be reused, it may be beneficial to group such tests on the same worker in order to avoid paying the cost of setup/teardown.
  - Setup/teardown duration should be recorded and be associated to each test.
  - Some system have different types of setup/teardown: global, per class/module/scope. Being able to be aware of whether the test system will need to re-run a set of setup/teardown functions may help with scheduling the tests.
- While we may be collecting test duration data, it is important to be aware that this data may end up being tainted if used in a continuous integration system. What this means is that we may sometimes record durations that are out of the ordinary due to a bug in the code, or because the tests execute on different hardware.
  - To mitigate this issue we may use different approaches such as using the median of the recorded values (which is more robust to outliers than the mean), use a percentile (such as 95%) to determine the expected duration, record the duration of successful tests separately from failed ones, record tests in separate buckets defined by the user (e.g., defined on the CPU spec used to run the test), etc.
- If the number/list of tests hasn't changed since the last run, it would also be beneficial to store the computed schedule so that it is only computed once and reused many times, which is a common use case. The only time you may not want to do this is if computing the schedule is very cheap and represent a neglectable amount of time in the overall testing process.

1 Example use case - pytest-xdist

- By default pytest does not store any prior test duration so we would have to estimate that all tests are of equal duration.
  - As we run tests, we may start to collect information about the duration of parameterized tests.

  This may serve us to determine whether the parameterized test has the same duration over different set of parameters and serve as a base to start providing estimates for the remaining parameters combination of this test.
  - A system similar to the cache provided by pytest may be used to record the duration of tests

2 Concepts

- Scheduler: responsible for taking a set of tests with meta data and scheduling them (i.e., determine in which order they will be executed).
- test: the unit of code to execute.
- test duration: the duration of a test. Test duration may be represented as a distribution since tests generally do not complete in an exact and fixed amount of time.
- test unit: a collection of tests that we can assume as being the same test (e.g., parameterized tests).

3 References

- https://en.wikipedia.org/wiki/Scheduling_(computing)
- https://en.wikipedia.org/wiki/Queueing_theory
- https://en.wikipedia.org/wiki/Queuing_Rule_of_Thumb
- https://docs.pytest.org/en/latest/cache.html