pytest-xdist

Jan 12, 2023

Contents:

The *pytest-xdist* plugin extends pytest with new test execution modes, the most used being distributing tests across multiple CPUs to speed up test execution:

```
pytest -n auto
```

With this call, pytest will spawn a number of workers processes equal to the number of available CPUs, and distribute the tests randomly across them.

Note: Due to how pytest-xdist is implemented, the `-s/--capture=no` option does not work.

CHAPTER 1

Installation

Install the plugin with:

```
pip install pytest-xdist
```

To use `psutil` for detection of the number of CPUs available, install the `psutil` extra:

```
pip install pytest-xdist[psutil]
```

CHAPTER 2

Features

- Test run *Running tests across multiple CPUs*: tests can be executed across multiple CPUs or hosts. This allows to speed up development or to use special resources of *Sending tests to remote SSH accounts*.
- `--looponfail`: run your tests repeatedly in a subprocess. After each run pytest waits until a file in your project changes and then re-runs the previously failing tests. This is repeated until all tests pass after which again a full run is performed (DEPRECATED).
- *Sending tests to remote SSH accounts* coverage: you can specify different Python interpreters or different platforms and run tests in parallel on all of them.

  Before running tests remotely, `pytest` efficiently "rsyncs" your program source code to the remote place. You may specify different Python versions and interpreters. It does not installs/synchronize dependencies however.

  Note: this mode exists mostly for backward compatibility, as modern development relies on continuous integration for multi-platform testing.

2.1 Running tests across multiple CPUs

To send tests to multiple CPUs, use the `-n` (or `--numprocesses`) option:

```
pytest -n auto
```

This can lead to considerable speed ups, especially if your test suite takes a noticeable amount of time.

With `-n auto`, pytest-xdist will use as many processes as your computer has CPU cores.

Use `-n logical` to use the number of *logical* CPU cores rather than physical ones. This currently requires the `psutil` package to be installed; if it is not, pytest-xdist will fall back to `-n auto` behavior.

Pass a number, e.g. `-n 8`, to specify the number of processes explicitly.

To specify a different meaning for `-n auto` and `-n logical` for your tests, you can:

- Set the environment variable `PYTEST_XDIST_AUTO_NUM_WORKERS` to the desired number of processes.

- Implement the `pytest_xdist_auto_num_workers` pytest hook (a `pytest_xdist_auto_num_workers(config)` function in e.g. `conftest.py`) that returns the number of processes to use. The hook can use `config.option.numprocesses` to determine if the user asked for `"auto"` or `"logical"`, and it can return `None` to fall back to the default.

If both the hook and environment variable are specified, the hook takes priority.

Parallelization can be configured further with these options:

- `--maxprocesses=maxprocesses`: limit the maximum number of workers to process the tests.
- `--max-worker-restart`: maximum number of workers that can be restarted when crashed (set to zero to disable this feature).

The test distribution algorithm is configured with the `--dist` command-line option:

- `--dist load` **(default)**: Sends pending tests to any worker that is available, without any guaranteed order.
- `--dist loadscope`: Tests are grouped by **module** for *test functions* and by **class** for *test methods*. Groups are distributed to available workers as whole units. This guarantees that all tests in a group run in the same process. This can be useful if you have expensive module-level or class-level fixtures. Grouping by class takes priority over grouping by module.
- `--dist loadfile`: Tests are grouped by their containing file. Groups are distributed to available workers as whole units. This guarantees that all tests in a file run in the same worker.
- `--dist loadgroup`: Tests are grouped by the `xdist_group` mark. Groups are distributed to available workers as whole units. This guarantees that all tests with same `xdist_group` name run in the same worker.

```
@pytest.mark.xdist_group(name="group1")
def test1():
    pass


class TestA:
    @pytest.mark.xdist_group("group1")
    def test2():
        pass
```

  This will make sure `test1` and `TestA::test2` will run in the same worker. Tests without the `xdist_group` mark are distributed normally as in the `--dist=load` mode.

- `--dist worksteal`: Initially, tests are distributed evenly among all available workers. When a worker completes most of its assigned tests and doesn't have enough tests to continue (currently, every worker needs at least two tests in its queue), an attempt is made to reassign ("steal") a portion of tests from some other worker's queue. The results should be similar to the `load` method, but `worksteal` should handle tests with significantly differing duration better, and, at the same time, it should provide similar or better reuse of fixtures.
- `--dist no`: The normal pytest execution mode, runs one test at a time (no distribution at all).

2.2 Running tests in a Python subprocess

To instantiate a `python3.9` subprocess and send tests to it, you may type:

```
pytest -d --tx popen//python=python3.9
```

This will start a subprocess which is run with the `python3.9` Python interpreter, found in your system binary lookup path.

If you prefix the –tx option value like this:

```
--tx 3*popen//python=python3.9
```

then three subprocesses would be created and tests will be load-balanced across these three processes.

2.3 Sending tests to remote SSH accounts

Deprecated since version 3.0.

Warning: This feature is deprecated because the support for `rsync` is faulty in terms of reproducing the development environment in the remote worker, and there is no clear solution moving forward.

This feature is scheduled to be removed in release 4.0, to let the team focus on a smaller set of features.

Suppose you have a package `mypkg` which contains some tests that you can successfully run locally. And you have a ssh-reachable machine `myhost`. Then you can ad-hoc distribute your tests by typing:

```
pytest -d  --rsyncdir mypkg --tx ssh=myhostpopen mypkg/tests/unit/test_something.py
```

This will synchronize your `mypkg` package directory to a remote ssh account and then locally collect tests and send them to remote places for execution.

You can specify multiple `--rsyncdir` directories to be sent to the remote side.

Note: For pytest to collect and send tests correctly you not only need to make sure all code and tests directories are rsynced, but that any test (sub) directory also has an `__init__.py` file because internally pytest references tests as a fully qualified python module path. **You will otherwise get strange errors** during setup of the remote side.

You can specify multiple `--rsyncignore` glob patterns to be ignored when file are sent to the remote side. There are also internal ignores: `.*, *.pyc, *.pyo, *~` Those you cannot override using rsyncignore command-line or ini-file option(s).

2.3.1 Sending tests to remote Socket Servers

Download the single-module socketserver.py Python program and run it like this:

```
python socketserver.py
```

It will tell you that it starts listening on the default port. You can now on your home machine specify this new socket host with something like this:

```
pytest -d --tx socket=192.168.1.102:8888 --rsyncdir mypkg
```

2.3.2 Running tests on many platforms at once

The basic command to run tests on multiple platforms is:

```
pytest --dist=each --tx=spec1 --tx=spec2
```

If you specify a windows host, an OSX host and a Linux environment this command will send each tests to all platforms - and report back failures from all platforms at once. The specifications strings use the xspec syntax.

2.4 When tests crash

If a test crashes a worker, pytest-xdist will automatically restart that worker and report the test's failure. You can use the `--max-worker-restart` option to limit the number of worker restarts that are allowed, or disable restarting altogether using `--max-worker-restart 0`.

2.5 How-tos

This section show cases how to accomplish some specialized tasks with `pytest-xdist`.

2.5.1 Identifying the worker process during a test

New in version 1.15.

If you need to determine the identity of a worker process in a test or fixture, you may use the `worker_id` fixture to do so:

```
@pytest.fixture()
def user_account(worker_id):
    """ use a different account in each xdist worker """
    return "account_%s" % worker_id
```

When `xdist` is disabled (running with `-n0` for example), then `worker_id` will return `"master"`.

Worker processes also have the following environment variables defined:

`PYTEST_XDIST_WORKER`

The name of the worker, e.g., `"gw2"`.

`PYTEST_XDIST_WORKER_COUNT`

The total number of workers in this session, e.g., `"4"` when `-n 4` is given in the command-line.

The information about the worker_id in a test is stored in the `TestReport` as well, under the `worker_id` attribute.

Since version 2.0, the following functions are also available in the `xdist` module:

2.5.2 Identifying workers from the system environment

New in version 2.4

If the setproctitle package is installed, `pytest-xdist` will use it to update the process title (command line) on its workers to show their current state. The titles used are `[pytest-xdist running] file.py/node::id` and `[pytest-xdist idle]`, visible in standard tools like `ps` and `top` on Linux, Mac OS X and BSD systems. For Windows, please follow setproctitle's pointer regarding the Process Explorer tool.

This is intended purely as an UX enhancement, e.g. to track down issues with long-running or CPU intensive tests. Errors in changing the title are ignored silently. Please try not to rely on the title format or title changes in external scripts.

2.5.3 Uniquely identifying the current test run

New in version 1.32.

If you need to globally distinguish one test run from others in your workers, you can use the `testrun_uid` fixture. For instance, let's say you wanted to create a separate database for each test run:

```
import pytest
from posix_ipc import Semaphore, O_CREAT


@pytest.fixture(scope="session", autouse=True)
def create_unique_database(testrun_uid):
    """ create a unique database for this particular test run """
    database_url = f"psql://myapp-{testrun_uid}"

    with Semaphore(f"/{testrun_uid}-lock", flags=O_CREAT, initial_value=1):
        if not database_exists(database_url):
            create_database(database_url)


@pytest.fixture()
def db(testrun_uid):
    """ retrieve unique database """
    database_url = f"psql://myapp-{testrun_uid}"
    return database_get_instance(database_url)
```

Additionally, during a test run, the following environment variable is defined:

PYTEST_XDIST_TESTRUNUID

The unique id of the test run.

2.5.4 Accessing `sys.argv` from the controller node in workers

To access the `sys.argv` passed to the command-line of the controller node, use `request.config.workerinput["mainargv"]`.

2.5.5 Specifying test exec environments in an ini file

You can use pytest's ini file configuration to avoid typing common options. You can for example make running with three subprocesses your default like this:

```
[pytest]
addopts = -n3
```

You can also add default environments like this:

```
[pytest]
addopts = --tx ssh=myhost//python=python3.9 --tx ssh=myhost//python=python3.6
```

and then just type:

```
pytest --dist=each
```

to run tests in each of the environments.

2.5.6 Specifying "rsync" dirs in an ini-file

In a `tox.ini` or `setup.cfg` file in your root project directory you may specify directories to include or to exclude in synchronisation:

```
[pytest]
rsyncdirs = . mypkg helperpkg
rsyncignore = .hg
```

These directory specifications are relative to the directory where the configuration file was found.

2.5.7 Making session-scoped fixtures execute only once

`pytest-xdist` is designed so that each worker process will perform its own collection and execute a subset of all tests. This means that tests in different processes requesting a high-level scoped fixture (for example `session`) will execute the fixture code more than once, which breaks expectations and might be undesired in certain situations.

While `pytest-xdist` does not have a builtin support for ensuring a session-scoped fixture is executed exactly once, this can be achieved by using a lock file for inter-process communication.

The example below needs to execute the fixture `session_data` only once (because it is resource intensive, or needs to execute only once to define configuration options, etc), so it makes use of a FileLock to produce the fixture data only once when the first process requests the fixture, while the other processes will then read the data from a file.

Here is the code:

```
import json

import pytest
from filelock import FileLock


@pytest.fixture(scope="session")
def session_data(tmp_path_factory, worker_id):
    if worker_id == "master":
        # not executing in with multiple workers, just produce the data and let
        # pytest's fixture caching do its job
        return produce_expensive_data()

    # get the temp directory shared by all workers
    root_tmp_dir = tmp_path_factory.getbasetemp().parent

    fn = root_tmp_dir / "data.json"
    with FileLock(str(fn) + ".lock"):
        if fn.is_file():
            data = json.loads(fn.read_text())
        else:
            data = produce_expensive_data()
            fn.write_text(json.dumps(data))
    return data
```

The example above can also be use in cases a fixture needs to execute exactly once per test session, like initializing a database service and populating initial tables.

This technique might not work for every case, but should be a starting point for many situations where executing a high-scope fixture exactly once is important.

2.5.8 Creating one log file for each worker

To create one log file for each worker with `pytest-xdist`, you can leverage *`PYTEST_XDIST_WORKER`* to generate a unique filename for each worker.

Example:

```
# content of conftest.py
def pytest_configure(config):
    worker_id = os.environ.get("PYTEST_XDIST_WORKER")
    if worker_id is not None:
        log_file = config.getini("worker_log_file")
        logging.basicConfig(
            format=config.getini("log_file_format"),
            filename=f"tests_{worker_id}.log",
            level=config.getini("log_file_level"),
        )
```

When running the tests with `-n3`, for example, three files will be created in the current directory: `tests_gw0.log`, `tests_gw1.log` and `tests_gw2.log`.

2.6 How it works?

`xdist` works by spawning one or more **workers**, which are controlled by the **controller**. Each **worker** is responsible for performing a full test collection and afterwards running tests as dictated by the **controller**.

The execution flow is:

1. **controller** spawns one or more **workers** at the beginning of the test session. The communication between **controller** and **worker** nodes makes use of execnet and its gateways. The actual interpreters executing the code for the **workers** might be remote or local.
2. Each **worker** itself is a mini pytest runner. **workers** at this point perform a full test collection, sending back the collected test-ids back to the **controller** which does not perform any collection itself.
3. The **controller** receives the result of the collection from all nodes. At this point the **controller** performs some sanity check to ensure that all **workers** collected the same tests (including order), bailing out otherwise. If all is well, it converts the list of test-ids into a list of simple indexes, where each index corresponds to the position of that test in the original collection list. This works because all nodes have the same collection list, and saves bandwidth because the **controller** can now tell one of the workers to just *execute test index 3* index of passing the full test id.
4. If **dist-mode** is **each**: the **controller** just sends the full list of test indexes to each node at this moment.
5. If **dist-mode** is **load**: the **controller** takes around 25% of the tests and sends them one by one to each **worker** in a round robin fashion. The rest of the tests will be distributed later as **workers** finish tests (see below).
6. Note that `pytest_xdist_make_scheduler` hook can be used to implement custom tests distribution logic.
7. **workers** re-implement `pytest_runtestloop`: pytest's default implementation basically loops over all collected items in the `session` object and executes the `pytest_runtest_protocol` for each test item, but in xdist **workers** sit idly waiting for **controller** to send tests for execution. As tests are received by **workers**, `pytest_runtest_protocol` is executed for each test. Here it worth noting an implementation detail: **workers** always must keep at least one test item on their queue due to how the `pytest_runtest_protocol(item, nextitem)` hook is defined: in order to pass the `nextitem` to the hook, the worker must wait for more instructions from controller before executing that remaining test. If it receives more tests, then it can safely call `pytest_runtest_protocol` because it knows what the

`nextitem` parameter will be. If it receives a "shutdown" signal, then it can execute the hook passing `nextitem` as `None`.

8. As tests are started and completed at the **workers**, the results are sent back to the **controller**, which then just forwards the results to the appropriate pytest hooks: `pytest_runtest_logstart` and `pytest_runtest_logreport`. This way other plugins (for example `junitxml`) can work normally. The **controller** (when in dist-mode **load**) decides to send more tests to a node when a test completes, using some heuristics such as test durations and how many tests each **worker** still has to run.
9. When the **controller** has no more pending tests it will send a "shutdown" signal to all **workers**, which will then run their remaining tests to completion and shut down. At this point the **controller** will sit waiting for **workers** to shut down, still processing events such as `pytest_runtest_logreport`.

2.6.1 FAQ

Question: Why does each worker do its own collection, as opposed to having the controller collect once and distribute from that collection to the workers?

If collection was performed by controller then it would have to serialize collected items to send them through the wire, as workers live in another process. The problem is that test items are not easily (impossible?) to serialize, as they contain references to the test functions, fixture managers, config objects, etc. Even if one manages to serialize it, it seems it would be very hard to get it right and easy to break by any small change in pytest.

2.7 Known limitations

pytest-xdist has some limitations that may be supported in pytest but can't be supported in pytest-xdist.

2.7.1 Order and amount of test must be consistent

It is not possible to have tests that differ in order or their amount across workers.

This is especially true with `pytest.mark.parametrize`, when values are produced with sets or other unordered iterables/generators.

Example:

```
import pytest


@pytest.mark.parametrize("param", {"a", "b"})
def test_pytest_parametrize_unordered(param):
    pass
```

In the example above, the fact that `set` are not necessarily ordered can cause different workers to collect tests in different order, which will throw an error.

Workarounds

A solution to this is to guarantee that the parametrized values have the same order.

Some solutions:

- Convert your sequence to a `list`.

```
import pytest


@pytest.mark.parametrize("param", ["a", "b"])
def test_pytest_parametrize_unordered(param):
    pass
```

- Sort your sequence, guaranteeing order.

```
import pytest


@pytest.mark.parametrize("param", sorted({"a", "b"}))
def test_pytest_parametrize_unordered(param):
    pass
```

2.7.2 Output (stdout and stderr) from workers

The `-s`/`--capture=no` option is meant to disable pytest capture, so users can then see stdout and stderr output in the terminal from tests and application code in real time.

However, this option does not work with `pytest-xdist` because execnet the underlying library used for communication between master and workers, does not support transferring stdout/stderr from workers.

Currently, there are no plans to support this in `pytest-xdist`.

Debugging

This also means that debugging using PDB (or any other debugger that wants to use standard I/O) will not work. The `--pdb` option is disabled when distributing tests with `pytest-xdist` for this reason.

It is generally likely best to use `pytest-xdist` to find failing tests and then debug them without distribution; however, if you need to debug from within a worker process (for example, to address failures that only happen when running tests concurrently), remote debuggers (for example, python-remote-pdb or python-web-pdb) have been reported to work for this purpose.

2.8 Changelog

2.8.1 pytest-xdist 3.1.0 (2022-12-01)

Features

- #789: Users can now set a default distribution mode in their configuration file:

```
[pytest]
addopts = --dist loadscope
```

- #842: Python 3.11 is now officially supported.

Removals

- #842: Python 3.6 is no longer supported.

2.8.2 pytest-xdist 3.0.2 (2022-10-25)

Bug Fixes

- #813: Cancel shutdown when a crashed worker is restarted.

Deprecations

- #825: The `--rsyncdir` command line argument and `rsyncdirs` config variable are deprecated.

  The rsync feature will be removed in pytest-xdist 4.0.

- #826: The `--looponfail` command line argument and `looponfailroots` config variable are deprecated.

  The loop-on-fail feature will be removed in pytest-xdist 4.0.

Improved Documentation

- #791: Document the `pytest_xdist_auto_num_workers` hook.
- #796: Added known limitations section to documentation.
- #829: Document the `-n logical` option.

Features

- #792: The environment variable `PYTEST_XDIST_AUTO_NUM_WORKERS` can now be used to specify the default for `-n auto` and `-n logical`.
- #812: Partially restore old initial batch distribution algorithm in `LoadScheduling`.

  pytest orders tests for optimal sequential execution - i. e. avoiding unnecessary setup and teardown of fixtures. So executing tests in consecutive chunks is important for optimal performance.

  In v1.14, initial test distribution in `LoadScheduling` was changed to round-robin, optimized for the corner case, when the number of tests is less than `2 * number of nodes`. At the same time, it became worse for all other cases.

  For example: if some tests use some "heavy" fixture, and these tests fit into the initial batch, with round-robin distribution the fixture will be created `min(n_tests, n_workers)` times, no matter how many other tests there are.

  With the old algorithm (before v1.14), if there are enough tests not using the fixture, the fixture was created only once.

  So restore the old behavior for typical cases where the number of tests is much greater than the number of workers (or, strictly speaking, when there are at least 2 tests for every node).

Removals

- #468: The `--boxed` command-line option has been removed. If you still need this functionality, install pytest-forked separately.

Trivial Changes

- #468: The `py` dependency has been dropped.
- #822: Replace internal usage of `py.log` with a custom solution (but with the same interface).
- #823: Remove usage of `py._pydir` as an rsync candidate.
- #824: Replace internal usages of `py.path.local` by `pathlib.Path`.

2.8.3 pytest-xdist 2.5.0 (2021-12-10)

Deprecations and Removals

- #468: The `--boxed` command line argument is deprecated. Install pytest-forked and use `--forked` instead. pytest-xdist 3.0.0 will remove the `--boxed` argument and `pytest-forked` dependency.

Features

- #722: Full compatibility with pytest 7 - no deprecation warnings or use of legacy features.
- #733: New `--dist=loadgroup` option, which ensures all tests marked with `@pytest.mark.xdist_group` run in the same session/worker. Other tests run distributed as in `--dist=load`.

Trivial Changes

- #708: Use `@pytest.hookspec` decorator to declare hook options in `newhooks.py` to avoid warnings in `pytest 7.0`.
- #719: Use up-to-date `setup.cfg`/`pyproject.toml` packaging setup.
- #720: Require pytest>=6.2.0.
- #721: Started using type annotations and mypy checking internally. The types are incomplete and not published.

2.8.4 pytest-xdist 2.4.0 (2021-09-20)

Features

- #696: On Linux, the process title now changes to indicate the current worker state (running/idle).

  Depends on the setproctitle package, which can be installed with `pip install pytest-xdist[setproctitle]`.
- #704: Add support for Python 3.10.

2.8.5 pytest-xdist 2.3.0 (2021-06-16)

Deprecations and Removals

- #654: Python 3.5 is no longer supported.

Features

- #646: Add `--numprocesses=logical` flag, which automatically uses the number of logical CPUs available, instead of physical CPUs with `auto`.

  This is very useful for test suites which are not CPU-bound.

- #650: Added new `pytest_handlecrashitem` hook to allow handling and rescheduling crashed items.

Bug Fixes

- #421: Copy the parent process sys.path into local workers, to work around execnet's python -c adding the current directory to sys.path.
- #638: Fix issue caused by changing the branch name of the pytest repository.

Trivial Changes

- #592: Replace master with controller where ever possible.
- #643: Use 'main' to refer to pytest default branch in tox env names.

2.8.6 pytest-xdist 2.2.1 (2021-02-09)

Bug Fixes

- #623: Gracefully handle the pending deprecation of Node.fspath by using config.rootpath for topdir.

2.8.7 pytest-xdist 2.2.0 (2020-12-14)

Features

- #608: Internal errors in workers are now propagated to the master node.

2.8.8 pytest-xdist 2.1.0 (2020-08-25)

Features

- #585: New `pytest_xdist_auto_num_workers` hook can be implemented by plugins or `conftest.py` files to control the number of workers when `--numprocesses=auto` is given in the command-line.

Trivial Changes

- #585: `psutil` has proven to make `pytest-xdist` installation in certain platforms and containers problematic, so to use it for automatic number of CPUs detection users need to install the `psutil` extra:

```
pip install pytest-xdist[psutil]
```

2.8.9 pytest-xdist 2.0.0 (2020-08-12)

Deprecations and Removals

- #541: Drop backward-compatibility "slave" aliases related to worker nodes. We deliberately moved away from this terminology years ago, and it seems like the right time to finish the deprecation and removal process.
- #569: `pytest-xdist` no longer supports Python 2.7.

Features

- #504: New functions `xdist.is_xdist_worker`, `xdist.is_xdist_master`, `xdist.get_xdist_worker_id`, to easily identify the current node.

Bug Fixes

- #471: Fix issue with Rsync reporting in quiet mode.
- #553: When using `-n auto`, count the number of physical CPU cores instead of logical ones.

Trivial Changes

- #541: `pytest-xdist` now requires `pytest>=6.0`.

2.8.10 pytest-xdist 1.34.0 (2020-07-27)

Features

- #549: Make `--pdb` imply `--dist no`, as the two options cannot really work together at the moment.

Bug Fixes

- #478: Fix regression with duplicated arguments via $PYTEST_ADDOPTS in 1.30.0.
- #558: Fix `rsyncdirs` usage with pytest 6.0.
- #562: Do not trigger the deprecated `pytest_warning_captured` in pytest 6.0+.

2.8.11 pytest-xdist 1.33.0 (2020-07-09)

Features

- #554: Fix warnings support for upcoming pytest 6.0.

Trivial Changes

- #548: SCM and CI files are no longer included in the source distribution.

2.8.12 pytest-xdist 1.32.0 (2020-05-03)

Deprecations and Removals

- #475: Drop support for EOL Python 3.4.

Features

- #524: Add *testrun_uid* fixture. This is a shared value that uniquely identifies a test run among all workers. This also adds a *PYTEST_XDIST_TESTRUNUID* environment variable that is accessible within a test as well as a command line option *–testrunuid* to manually set the value from outside.

2.8.13 pytest-xdist 1.31.0 (2019-12-19)

Features

- #486: Add support for Python 3.8.

Bug Fixes

- #491: Fix regression that caused custom plugin command-line arguments to be discarded when using `--tx` mode.

2.8.14 pytest-xdist 1.30.0 (2019-10-01)

Features

- #448: Initialization between workers and master nodes is now more consistent, which fixes a number of long-standing issues related to startup with the `-c` option.

  Issues:

  - #6: Poor interaction between `-n#` and `-c X.cfg`
  - #445: pytest-xdist is not reporting the same nodeid as pytest does

  This however only works with **pytest 5.1 or later**, as it required changes in pytest itself.

Bug Fixes

- #467: Fix crash issues related to running xdist with the terminal plugin disabled.

2.8.15 pytest-xdist 1.29.0 (2019-06-14)

Features

- #226: `--max-worker-restart` now assumes a more reasonable value (4 times the number of nodes) when not given explicitly. This prevents test suites from running forever when the suite crashes during collection.
- #435: When the test session is interrupted due to running out of workers, the reason is shown in the test summary for easier viewing.

- #442: Compatibility fix for upcoming pytest 5.0: `session.exitstatus` is now an `IntEnum` object.

Bug Fixes

- #435: No longer show an internal error when we run out of workers due to crashes.

2.8.16 pytest-xdist 1.28.0 (2019-04-02)

Features

- #426: `pytest-xdist` now uses the new `pytest_report_to_serializable` and `pytest_report_from_serializable` hooks from `pytest 4.4` (still experimental). This will make report serialization more reliable and extensible.

  This also means that `pytest-xdist` now requires `pytest>=4.4`.

2.8.17 pytest-xdist 1.27.0 (2019-02-15)

Features

- #374: The new `pytest_xdist_getremotemodule` hook allows overriding the module run on remote nodes.
- #415: Improve behavior of `--numprocesses=auto` to work well with `--pdb` option.

2.8.18 pytest-xdist 1.26.1 (2019-01-28)

Bug Fixes

- #406: Do not implement deprecated `pytest_logwarning` hook in pytest versions where it is deprecated.

2.8.19 pytest-xdist 1.26.0 (2019-01-11)

Features

- #376: The current directory is no longer added `sys.path` for local workers, only for remote connections.

  This behavior is surprising because it makes xdist runs and non-xdist runs to potentially behave differently.

Bug Fixes

- #379: Warning attributes are checked to make sure they can be dumped prior to serializing the warning for submission to the master node.

2.8.20 pytest-xdist 1.25.0 (2018-12-12)

Deprecations and Removals

- #372: Pytest versions older than 3.6 are no longer supported.

Features

- #373: Node setup information is hidden when pytest is run in quiet mode to reduce noise on many-core machines.
- #388: `mainargv` is made available in `workerinput` from the host's `sys.argv`.

  This can be used via `request.config.workerinput["mainargv"]`.

Bug Fixes

- #332: Fix report of module-level skips (`pytest.skip(reason, allow_module_level=True)`).
- #378: Fix support for gevent monkeypatching
- #384: pytest 4.1 support: `ExceptionInfo` API changes.
- #390: pytest 4.1 support: `pytest_logwarning` hook removed.

2.8.21 pytest-xdist 1.24.1 (2018-11-09)

Bug Fixes

- #349: Correctly handle warnings created with arguments that can't be serialized during the transfer from workers to master node.

2.8.22 pytest-xdist 1.24.0 (2018-10-18)

Features

- #337: New `--maxprocesses` command-line option that limits the maximum number of workers when using `--numprocesses=auto`.

Bug Fixes

- #351: Fix scheduling deadlock in case of inter-test locking.

2.8.23 pytest-xdist 1.23.2 (2018-09-28)

Bug Fixes

- #344: Fix issue where Warnings could cause pytest to fail if they do not set the args attribute correctly.

2.8.24 pytest-xdist 1.23.1 (2018-09-25)

Bug Fixes

- #341: Fix warnings transfer between workers and master node with pytest >= 3.8.

2.8.25 pytest-xdist 1.23.0 (2018-08-23)

Features

- #330: Improve collection performance by reducing the number of events sent to `master` node.

2.8.26 pytest-xdist 1.22.5 (2018-07-27)

Bug Fixes

- #321: Revert change that dropped support for `pytest<3.4` and require `six`.

  This change caused problems in some installations, and was a mistaken in the first place as we should not change version requirements in bug-fix releases unless they fix an actual bug.

2.8.27 pytest-xdist 1.22.4 (2018-07-27)

Bug Fixes

- #305: Remove last references to obsolete `py.code`.

  Remove some unnecessary references to `py.builtin`.

- #316: Workaround cpu detection on Travis CI.

2.8.28 pytest-xdist 1.22.3 (2018-07-23)

Bug Fixes

- Fix issue of virtualized or containerized environments not reporting the number of CPUs correctly. (#9)

Trivial Changes

- Make all classes subclass from `object` and fix `super()` call in `LoadFileScheduling`; (#297)

2.8.29 pytest-xdist 1.22.2 (2018-02-26)

Bug Fixes

- Add backward compatibility for `slaveoutput` attribute to `WorkerController` instances. (#285)

2.8.30 pytest-xdist 1.22.1 (2018-02-19)

Bug Fixes

- Fix issue when using `loadscope` or `loadfile` where tests would fail to start if the first scope had only one test. (#257)

Trivial Changes

- Change terminology used by `pytest-xdist` to *master* and *worker* in arguments and messages (for example `--max-worker-reset`). (#234)

2.8.31 pytest-xdist 1.22.0 (2018-01-11)

Features

- Add support for the `pytest_runtest_logfinish` hook which will be released in pytest 3.4. (#266)

2.8.32 pytest-xdist 1.21.0 (2017-12-22)

Deprecations and Removals

- Drop support for EOL Python 2.6. (#259)

Features

- New `--dist=loadfile` option which load-distributes test to workers grouped by the file the tests live in. (#242)

Bug Fixes

- Fix accidental mutation of test report during serialization causing longrepr string-ification to break. (#241)

2.8.33 pytest-xdist 1.20.1 (2017-10-05)

Bug Fixes

- Fix hang when all worker nodes crash and restart limit is reached (#45)
- Fix issue where the -n option would still run distributed tests when pytest was run with the –collect-only option (#5)

2.8.34 pytest-xdist 1.20.0 (2017-08-17)

Features

- `xdist` now supports tests to log results multiple times, improving integration with plugins which require it like pytest-rerunfailures and flaky. (#206)

Bug Fixes

- Fix issue where tests were being incorrectly identified if a worker crashed during the `teardown` stage of the test. (#124)

2.8.35 pytest-xdist 1.19.1 (2017-08-10)

Bug Fixes

- Fix crash when transferring internal pytest warnings from workers to the master node. (#214)

2.8.36 pytest-xdist 1.19.0 (2017-08-09)

Deprecations and Removals

- `--boxed` functionality has been moved to a separate plugin, pytest-forked. This release now depends on ‘‘ pytest-forked‘‘ and provides `--boxed` as a backward compatibility option. (#1)

Features

- New `--dist=loadscope` option: sends group of related tests to the same worker. Tests are grouped by module for test functions and by class for test methods. See `README.rst` for more information. (#191)
- Warnings are now properly transferred from workers to the master node. (#92)

Bug Fixes

- Fix serialization of native tracebacks (`--tb=native`). (#196)

2.8.37 pytest-xdist 1.18.2 (2017-07-28)

Bug Fixes

- Removal of unnecessary dependency on incorrect version of py. (#105)
- Fix bug in internal event-loop error handler in the master node. This bug would shadow the original errors making extremely hard/impossible for users to diagnose the problem properly. (#175)

2.8.38 pytest-xdist 1.18.1 (2017-07-05)

Bug Fixes

- Fixed serialization of `longrepr.sections` during error reporting from workers. (#171)
- Fix `ReprLocal` not being unserialized breaking –showlocals usages. (#176)

2.8.39 pytest-xdist 1.18.0 (2017-06-26)

- `pytest-xdist` now requires `pytest>=3.0.0`.

Features

- Add long option *–numprocesses* as alternative for *-n*. (#168)

Bug Fixes

- Fix serialization and deserialization dropping longrepr details. (#133)

2.8.40 pytest-xdist 1.17.1 (2017-06-10)

Bug Fixes

- Hot fix release reverting the change introduced by #124, unfortunately it broke a number of test suites so we are reversing this change while we investigate the problem. (#157)

Improved Documentation

- Introduced `towncrier` for `CHANGELOG` management. (#154)
- Added `HOWTORELEASE` documentation. (#155)

1.17.0

- fix #124: xdist would mark test as complete after 'call' step. As a result, xdist could identify the wrong test as failing when test crashes at teardown. To address this issue, xdist now marks test as complete at teardown.

1.16.0

- `pytest-xdist` now requires pytest 2.7 or later.
- Add `worker_id` attribute in the TestReport
- new hook: `pytest_xdist_make_scheduler(config, log)`, can return custom tests items distribution logic implementation. You can take a look at built-in `LoadScheduling` and `EachScheduling` implementations. Note that required scheduler class public API may change in next `pytest-xdist` versions.

1.15.0

- new `worker_id` fixture, returns the id of the worker in a test or fixture. Thanks Jared Hellman for the PR.
- display progress during collection only when in a terminal, similar to pytest #1397 issue. Thanks Bruno Oliveira for the PR.
- fix internal error message when `--maxfail` is used (#62, #65). Thanks Collin RM Stocks and Bryan A. Jones for reports and Bruno Oliveira for the PR.

1.14

- new hook: `pytest_xdist_node_collection_finished(node, ids)`, called when a worker has finished collection. Thanks Omer Katz for the request and Bruno Oliveira for the PR.
- fix README display on pypi
- fix #22: xdist now works if the internal tmpdir plugin is disabled. Thanks Bruno Oliveira for the PR.
- fix #32: xdist now works if looponfail or boxed are disabled. Thanks Bruno Oliveira for the PR.

1.13.1

- fix a regression -n 0 now disables xdist again

1.13

- extended the tox matrix with the supported py.test versions
- split up the plugin into 3 plugin's to prepare the departure of boxed and looponfail.

  looponfail will be a part of core and forked boxed will be replaced with a more reliable primitive based on xdist
- conforming with new pytest-2.8 behavior of returning non-zero when all tests were skipped or deselected.
- new "–max-slave-restart" option that can be used to control maximum number of times pytest-xdist can restart slaves due to crashes. Thanks to Anatoly Bubenkov for the report and Bruno Oliveira for the PR.
- release as wheel
- "-n" option now can be set to "auto" for automatic detection of number of cpus in the host system. Thanks Suloev Dmitry for the PR.

1.12

- fix issue594: properly report errors when the test collection is random. Thanks Bruno Oliveira.
- some internal test suite adaptation (to become forward compatible with the upcoming pytest-2.8)

1.11

- fix pytest/xdist issue485 (also depends on py-1.4.22): attach stdout/stderr on –boxed processes that die.
- fix pytest/xdist issue503: make sure that a node has usually two items to execute to avoid scoped fixtures to be torn down pre-maturely (fixture teardown/setup is "nextitem" sensitive). Thanks to Andreas Pelme for bug analysis and failing test.
- restart crashed nodes by internally refactoring setup handling of nodes. Also includes better code documentation. Many thanks to Floris Bruynooghe for the complete PR.

1.10

- add glob support for rsyncignores, add command line option to pass additional rsyncignores. Thanks Anatoly Bubenkov.
- fix pytest issue382 - produce "pytest_runtest_logstart" event again in master. Thanks Aron Curzon.
- fix pytest issue419 by sending/receiving indices into the test collection instead of node ids (which are not necessarily unique for functions parametrized with duplicate values)
- send multiple "to test" indices in one network message to a slave and improve heuristics for sending chunks where the chunksize depends on the number of remaining tests rather than fixed numbers. This reduces the number of master -> node messages (but not the reverse direction)

1.9

- changed LICENSE to MIT
- fix duplicate reported test ids with –looponfailing (thanks Jeremy Thurgood)
- fix pytest issue41: re-run tests on all file changes, not just randomly select ones like .py/.c.
- fix pytest issue347: slaves running on top of Python3.2 will set PYTHONDONTWRITEYBTECODE to 1 to avoid import concurrency bugs.

1.8

- fix pytest-issue93 - use the refined pytest-2.2.1 runtestprotocol interface to perform eager teardowns for test items.

1.7

- fix incompatibilities with pytest-2.2.0 (allow multiple pytest_runtest_logreport reports for a test item)

1.6

- terser collection reporting
- fix issue34 - distributed testing with -p plugin now works correctly
- fix race condition in looponfail mode where a concurrent file removal could cause a crash

1.5

- adapt to and require pytest-2.0 changes, rsyncdirs and rsyncignore can now only be specified in [pytest] sections of ini files, see "py.test -h" for details.
- major internal refactoring to match the pytest-2.0 event refactoring - perform test collection always at slave side instead of at the master - make python2/python3 bridging work, remove usage of pickling
- improve initial reporting by using line-rewriting
- remove all trailing whitespace from source

1.4

- perform distributed testing related reporting in the plugin rather than having dist-related code in the generic py.test distribution
- depend on execnet-1.0.7 which adds "env1:NAME=value" keys to gateway specification strings.
- show detailed gateway setup and platform information only when "-v" or "–verbose" is specified.

1.3

- fix –looponfailing - it would not actually run against the fully changed source tree when initial conftest files load application state.
- adapt for py-1.3.1's new –maxfailure option

1.2

- fix issue79: sessionfinish/teardown hooks are now called systematically on the slave side
- introduce a new data input/output mechanism to allow the master side to send and receive data from a slave.
- fix race condition in underlying pickling/unpickling handling
- use and require new register hooks facility of py.test>=1.3.0
- require improved execnet>=1.0.6 because of various race conditions that can arise in xdist testing modes.
- fix some python3 related pickling related race conditions
- fix PyPI description

1.1

- fix an indefinite hang which would wait for events although no events are pending - this happened if items arrive very quickly while the "reschedule-event" tried unconditionally avoiding a busy-loop and not schedule new work.

1.0

- moved code out of py-1.1.1 into its own plugin
- use a new, faster and more sensible model to do load-balancing of tests - now no magic "MAXITEMSPERHOST" is needed and load-testing works effectively even with very few tests.
- cleaned up termination handling
- make -x cause hard killing of test nodes to decrease wait time until the traceback shows up on first failure

Index