A STEP-BY-STEP
GUIDE

StorPool
DISTRIBUTED STORAGE

Storage Performance and Resilience Testing:

How to select a storage system for your cloud

TABLE OF CONTENTS

INTRODUCTION

When choosing a new storage system, it is crucial to test different storage alternatives properly. The storage tests should combine a number of tests, meant to assess the functionality, performance, and resilience of the system.

However, it is the preparation and research phases that are the foundation of your project. Especially the first step is often skipped or vaguely defined. Moreover, it is crucial to start with an internal focus - do an audit of your current state - use case, application(s), features used, resources consumed, speed and capacity requirements. Then outline an estimate of what you will need in the future.

List your "must have" features and "nice to have" ones, plus other relevant requirements. Only then proceed to look for solutions, which match your needs. Reading reviews and testing systems will not be optimal beforehand.

In this guide, we are describing a step-by-step process for the next technical step - testing different types of storage solutions and getting accurate, real-world results. This systematic process will help you to compare alternatives and select the one that matches your needs best.

GENERAL ADVICES ON TESTING ANY KIND OF STORAGE - THE BASICS YOU NEED TO KNOW

- Start with writing down the requirements for your storage system. This will help you to navigate between the many alternatives and shortlist a few that should be a good fit. For example, an entirely in-memory storage system would be the fastest possible, but it loses data in case of a power outage. If you need persistence, you'll disqualify a set of solutions, which are not a fit to your case and look for the right ratio of speed vs. data persistence.

- If possible, do the tests with the services/applications you'll be running in production. If not possible run a copy of the actual workload, besides the standard synthetic tests.

- Create a testing environment which is as close to the production environment as possible. Ideally, set-up an identical environment which would use the same network type and topology, hardware configurations and software versions.

- If evaluating different storage systems, change only the storage system between tests, not to introduce any more variables. If this is not possible, change the test setups as little as possible.

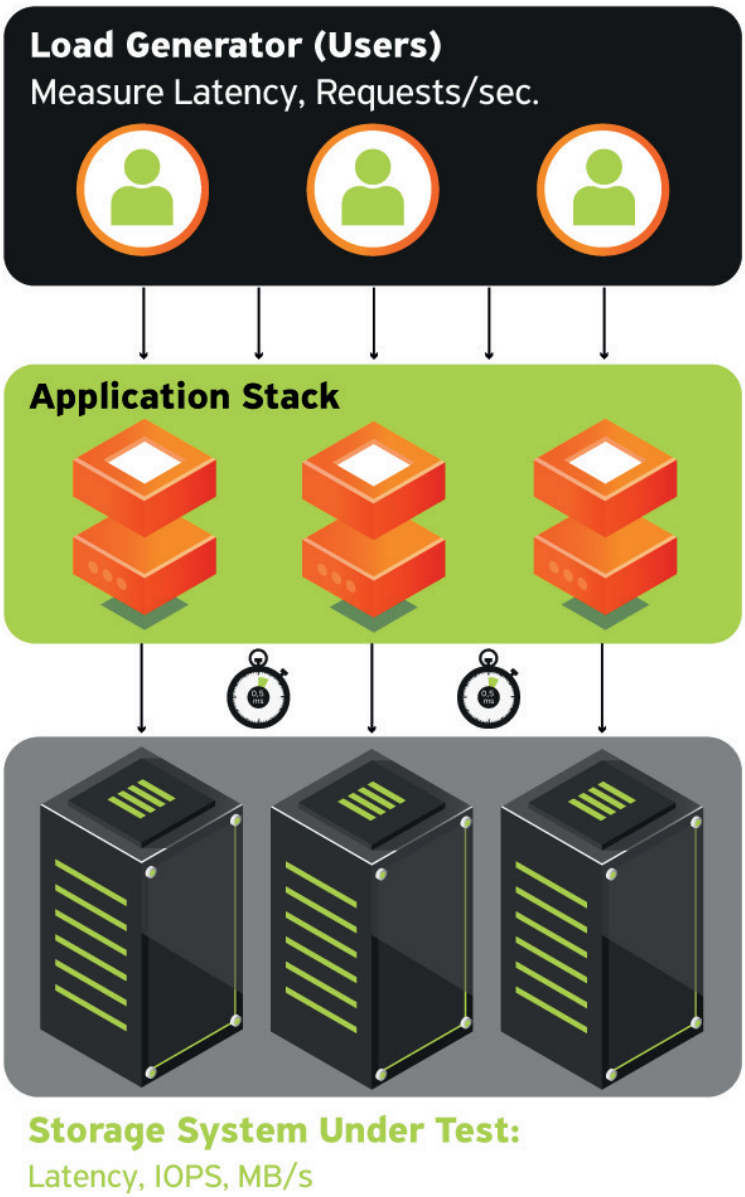
- Keep written track of the measured results, in a unified form/format. Also list the specifications of the environment, in which they were obtained. A spreadsheet can be of immense help. At the end of the document we have reference to a template.

- Run the tests multiple times so that you can sift-out any one-offs. Record measurements, as per the point above and strive for reproducibility of the tests. Some systems have much variability in their results. Take the time to understand what causes the variability and what is its effect on the overall work of the system.

- Be very careful with extrapolating results from small-sized tests. Twice the hardware doesn't always translate to twice the performance, automatically. You have to know where the bottlenecks of the system are, as well as whether the solution you are testing is a scale-out solution in the first place.

THE STORAGE TESTING ENVIRONMENT

Any testing environment has (at least) three components: storage, application and load generator. Of course, there is the networking in between, etc., but here we will focus on the storage aspect of things.



The storage and application are self-describing. A load generator is a tool which initiates requests to the application with pre-configured parallelism and rate, to make it possible to observe the application's behavior under load, which in turn shows the metrics of the storage subsystem. Load generators come in different types - from tools which can generate a single request, to ones with programmable scenarios, which simulate complex real end-users behavior.

If you can deploy the application or service, which you will be running in production, on the test system - this is great. For everyone who isn't able to use the real workloads, there are some options you can look into described below.

Performance Testing

MICROBENCHMARKS/SYNTHETIC TESTS

It is possible to find out the performance limits of a storage system with just three sets of synthetic tests:

- ☑ LATENCY TESTS
- ☑ IOPS TESTS
- ☑ BANDWIDTH TESTS

LATENCY TESTS

A latency test is issuing one storage operation, just after the previous one completes, and measuring how long does it take to complete the operations, on average. This shows what is the minimum time that the system takes to react to any request. Most workloads are quite sensitive to latency, as they make a request, wait for its result, and then issue another request. Database transactions are one example and since most applications have a database, latency becomes a fundamental performance metric of a storage system. We would say that, latency is the #1 metric of your cloud.

IOPS TESTS

An IOPS test shows the amount of I/O operations, which the system can process as a whole, at different levels of parallel requests. It shows the upper bound of operations that can be performed by applications on the storage, i.e. for example it may help estimate a hard limit of how many database transactions can be performed on the storage. IOPS and latency are interconnected - for example "hero" IOPS numbers, typically quoted by most storage vendors are measured at points where the tested system is so overloaded, that the IOPS are peak, but latency is dismal (say >10ms). A latency this high is unfit for running production workloads, even if the IOPS numbers seem impressive.

BANDWIDTH TESTS

The bandwidth tests show the amount of data that can be read/written in a specific unit of time, which is a measurement of the throughput of the system. If you need to move large amounts of data to/from the storage, this measurement sets the upper bound of the possibilities.

STORAGE TEST SUITE

At StorPool we have developed a test suite based on the `fio` tool, which we use when testing storage systems - be it new systems we deliver or existing customer systems, as part of the capacity and performance planning. We made our test suite available here:

https://github.com/storpool/fio-tests

You can use the test suite to check the performance of any of the storage clusters you want to put into production. It's straightforward to use and shows all metrics described above (latency, IOPS, and bandwidth) for any device it's given to test.

Please note that these tests do reads and writes on the entire device. I.e. they will overwrite existing data on the device, so make sure you are using a scratch device.

PGBENCHTESTING

Synthetic tests give you reference numbers, but those don't translate directly to real-world loads. They just illustrate the performance envelope of the storage system (minimum latency, maximum IOPS, etc).
For better comparison between different systems, one may use the standardized pgbench test. It is a tool developed by the PostgreSQL team to test their own database. It's based on the industry standard TPC-B benchmark (http://www.tpc.org/tpcb/default.asp) and simulates a load where users execute database transactions that consist ot multiple SELECT, UPDATE and INSERT statements.

The meaningful tests that can be done with pgbench for a storage system are two kinds:

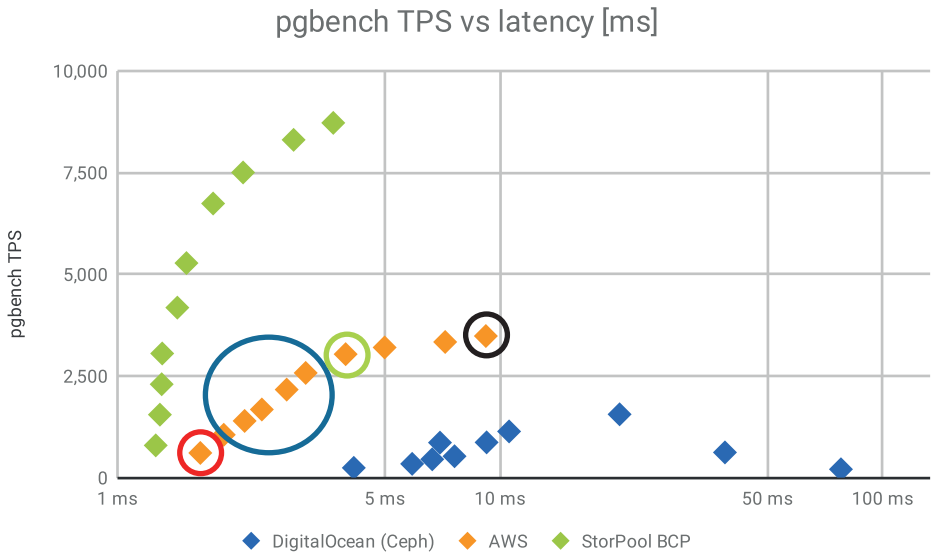
- ☑ the dataset being 0.9 times the memory of the node, which tests mostly the write part of the database
- ☑ the dataset being 4 times the memory of the node, which also tests the read part

Which test's results are more meaningful to you depends a lot on the read/write profile of your workloads. Some things haven't changed in the last 20-30 years: for example, any database performs better if its dataset fits into memory, which is something that everyone strives to do, even though it's not always possible. We've also seen that with a fast enough storage, the gap between dataset-in-memory and dataset-not-in-memory can be less frightening and even hard to notice.

More information on testing with pgbench can be found at:

https://wiki.postgresql.org/wiki/Pgbenchtesting

For every type of test (either 0.9x or 4x) there needs to be run with different levels of parallelism, to ascertain the abilities of the system. The single client/thread test shows the overall latency of the system, and the tests with more clients/threads can show the abilities of the system and its behaviour under load.



The above graph shows the performance of three different storage systems, indicated by different color. For each of the storage systems the number of parallel requests is varied and average latency and Transactions Per Second (TPS) are recorded. We have marked the following distinct points:

- Single thread, lowest latency (circled in red)
- The place where you want your system to usually be - low latency and space to grow (circled in blue)
- Saturation point - where raising the parallelism doesn't give you more throughput (cicled in green)
- What most benchmark results show and the situation you don't want to be in (circled in black)

For any real-world workload, the preferable load would be the red and blue parts of the curve - low latency under moderate load (not to affect the end users' experience). Everything after the saturation point in green (the "knee") point results in unacceptable delays, overload of the system and generally bad user experience. So several things are essential here:

1. How low is the starting latency of the system (the closer to zero, the better)?
2. Where the saturation point of the system is (the knee)?
3. And of course, what IOPS can the system deliver before the "knee," which shows the productive range of the storage system?

Knowing these 3 things is used in capacity planning to keep the real workload below the saturation point and guarantee that the performance requirements are met.

It is worth stressing once again, that latency (and not IOPS) is the most critical metric for a storage system. Real production workloads typically have low queue depth, i.e., they are latency sensitive, while they also experience occasional bursts (lots of operations submitted at the same time - a boot storm can be an example). Therefore, for real workloads, both latency under light load and burst handling capability are essential.

SYSBENCH TEST

Sysbench (https://github.com/akopytov/sysbench) is a scriptable multi-threaded benchmarking tool, used for database benchmarks. It's mostly used to test MySQL/MariaDB databases and in some ways is similar to the PGBench tests above, but its load is different - it tests in each transaction with indexed and unindexed updates, and does more queries (~50) in each transaction.

The databases' sizes for the different tests are set in numbers of rows. Our tests have shown that for one table, 20 million rows take around 4.16GiB for data and 298MiB for indexes, and as both the number of tables and number of rows is configurable, you can design a test which is IO-bound only for data lookups, or IO-bound for both index and data lookups. Using a similar approach as pgbench with 0.9x memory and 4x memory also makes sense with sysbench.

RSYNC TEST

One simple test we wrote to simulate file system access is our so-called "rsync test". Its dataset is 50 unpacked linux kernel 4.17.13 sources. The test consists of three runs:

- An initial copy of the whole directory, without any previous cache (the filesystem cache gets flushed first).
- A sync of the data, again with dropping the cache beforehand.
- A sync of the data without dropping the cache.

The test is very simple and is useful to measure the effects of the storage on large batched filesystem operations, unpacking archives and installing packages.

Resilience Testing

RESILIENCE TESTING

Here you do controlled experiments with the test system, trying to break it in various ways and observing how the system reacts. This includes things like restarting servers, clients, targets, unplugging network cables, drives, power cables, etc, etc.
The performance tests can also help as an element of resilience testing, as they stress the system and can expose flaws, which may otherwise not cause issues under low system load.

It is usually required to test the behaviour of the system under different failure conditions. Some types of failure conditions depend on the environment, but most can be summarized as follow:

1. Single component failure
2. Failures of unrelated single components
3. Multiple related component failures
4. Total system failures

When evaluating systems, for any failure scenario you should know:

1. What is the probability for this scenario to occur?
2. What is the effect on the system?
3. What is the cost/loss from this effect?

Based on answers, you should gain a good idea if a system is a good match for your needs. Or at least you will get a deeper understanding of your data and how you think about it.
In our experience this is a somewhat elusive matter, as statistics fall short to measure or try to predict it.
Let's illustrate this with a simple example that uses a large RAID10 system of 100 drives.
For example if you want to know what is the potential failure risk, you can try to calculate it.
Taking into account that a single drive's probability to fail within one year is, say 1.25% (as measured by BackBlaze), this means that in a 100-drive system one drive will likely fail.
This one failure will not result in data loss, assuming several things:

1. The mirror disk of the failed drive will not fail.
2. You change the failed drive fast enough.
3. All other components of the storage system work well and will not fail during reduced redundancy or the following rebuild (hot-spot alert).
4. External events will not interfere - datacenter power outage, human error, etc.

So in this case, you see the need for a procedure for rapid replacement of failed drives, either manually or automatically (hot spares) so you can protect against data loss. With this in mind, you can stock on some spare drives and be reasonably sure that you will not be affected by these failure scenarios. This is what people usually do. However, with raising amounts of data it may not be enough.

Statistically, the chance of the exact mirror drive failing is low. In practice though, drives fail in batches. Since usually the drives in the system are from the same production batch, if a drive fails, this in practice means that the likelihood of the other drive to fail is higher. Especially, since you are now making a hot spot on this drive as you rebuild, increasing its load and thus likelihood to fail.
Further it can be the disk controller that is actually causing the issue, thus corrupting the data on all drives attached to it (real case).

This is why it's hard and elusive to calculate failure risk and to protect against it.
In summary - you need 2 things here:

1. Get a system which is as reliable as possible. And 3 copies in 3 different servers are more reliable than 2 copies in the same box, this is why it's a distributed storage system.
2. Make sure to have clear written procedures for taking action, for when a system incident occurs. Train your staff and be prepared.

Different storage systems handle data redundancy in different ways, and the 100-drive RAID10 system is given just as an example of how to consider availability and availability-improving measures.
If you want to go down the rabbit hole on this, we recommend reading the chapter on Disaster Recovery and Data integrity in the great book "The practice of System and Network Administration". The book is a bit dated, still it is a fundamental work in the field.

A point we touched upon above - you should have understanding of all the components of the storage system, to be sure that you can check the failure scenarios for all of them. Some systems, like a single drive have relatively well-known failure modes and few components (for all intents and purposes, a drive is a single component), but most storage systems have many components:

- **DRIVES**
- **INTERNAL BUSES/BACKPLANES**
- **INTERNAL CONTROLLERS**
- **FIRMWARE**
- **STORAGE SOFTWARE**
- **CONTROL/MANAGEMENT**
- **STORAGE NETWORK**
- **CLIENT ACCESS**
- **PEOPLE (HUMAN ERRORS)**
- **ETC.**

These exist for both traditional SAN systems and software-defined storages. For example, the network(s) are an integral part of most distributed software-defined storages and their failure modes have to be taken into account.

And finally, for every failure mode of the system you need to have an expectation of the overall effect on the system, which the system should be able to satisfy.
Here are some examples:

- The most basic expectation is that if a single drive fails, the system would continue working.
- Similar to the previous one, if a drive gets corrupted, it will not cause corruption in the user data (by using data redundancy as well as a data integrity mechanism to detect corrupted data).
- If the storage system reports a write operation as persisted/written to the storage that the data would be there after a power outage or some component failure. This is also known as the property "Durability" in the ACID-compliant databases.
- If the whole system goes down because of, total network outage, it would continue working as soon as the network is restored.
- And, something that's rarely tested: if you introduce a lot of random problems in the system, when would it fail and how hard would it be to return to working state, and how much operator involvement would be required (as Netflix does with Chaos Monkey, https://netflix.github.io/chaosmonkey/).
- As a side note, any code that StorPool deploys in production passes a set of tests with semi-random killing of multiple components, to check the system's behavior and to ensure there are no unexpected issues.

Storage System Components: Failure Scenarios

Specially for the needs of IT experts and engineers, testing different types of storage systems, StorPool has prepared a simple table with different failures and the expected result. This table is a good starting point for designing your own set of tests.

When choosing a storage system, you need to be sure it will provide you with all the features and functionalities you business requires. On the other hand in the process of testing, you need to discover how the system will behave in different failure scenarios.

All this information will help you to make the perfect choice of a storage system for your public or private cloud.

COMPARING DIFFERENT STORAGE SYSTEMS?

REQUEST YOUR

Storage Comparison Sheet:
Failure Scenarios and Expected Results

Just write us at info@storpool.com
and we will send you the comparison sheet!

StorPool

DISTRIBUTED STORAGE

www.storpool.com
info@storpool.com

+1 415 670 9320
+44 (0) 20 7097 8536