Advantages of Distributed Computing

Mariana Ávalos Arce
Distributed Computing

In this following essay the main advantages of a distributed system that focuses on Consistency & Replication, Fault Tolerance, Security, and overall Testing/Release will be presented in general terms.

1 Consistency and Replication

According to the textbook *Distributed Systems*[2], in a distributed system, in order for it to be reliable, data is usually replicated. Then, an important issue arises, which is keeping these replicas consistent, or in other words, equally updated. This data replication increases the complexity of a distributed system implementation, because now all replicas would need to be modified exactly the same, which becomes an important issue. So what would be the advantages that makes this complex task worth doing?

Fist of all, replicating data increases the **reliability** of a system, because when a system duplicates data that is distributed, if a replica crashes or fails, information is not lost as we simply need to move to another replica to get the information needed. Having data equally updated also protects the system from **data corruption** due to failed actions, because if one action failed on a file, we still have the other replica's actions that may have been successful, making the data not corrupt.

Another advantage is that the system increases its **scalability**, either in terms of size or actual physical coverage of such system in the future. If a system uses data replication, when a large number of clients or processes needs access to a certain portion of stored data, these processes increase its **performance** if the servers are duplicated, because workload that otherwise would be dedicated for one server is now distributed in others, giving multiple and simultaneous points of access for a large number of processes instead of waiting for one to accept and serve all requests. This was mainly about a system that grows in size, but also performance is optimized by data duplication in the case of a system that grows geographically. If data is duplicated and placed in a server close to the requester, the time taken to access data is reduced in comparison to a single server in a single location containing the data in question.

The *relaxation* of the consistency rules in order to update all replicas is directly proportional to the **increased performance** the requester client will get. So, if these rules are strict such as requiring atomic operations for updating each replica, then the performance gain will be reduced. Thus, The **advantage of performance and scalability** implemented by replication is greater when the rules of updating are relaxed. Just like Van Steen wrote in his book, 'only if we loosen consistency can there be hope for attaining efficient solutions'.

2 Fault Tolerance

Before I move on to mention the advantages of Fault Tolerance, it is crucial to understand what it means: **Fault Tolerance** is a key characteristic of distributed systems, which consists in that part of the system failing while the remaining parts function properly and meanwhile repair those failures. A **fault** causes an

error, and an error leads to a **failure**. These systems provide their promised even when there are failures.

Now, the main advantage of a system that is fault tolerant is that such a system is a **dependable** system. A dependable system entails **other advantages of fault tolerance**, which are generally four:

- **System's availability**: a fault tolerant system is one that inherently will most likely be working at any given instant in time, because partial failures are handled without the client noticing by other internal parts that work properly.
- **System's Reliability**: a fault tolerant system also will most likely continue to work without interruption during a (preferably) long period of time, because fault tolerance means to avoid **crashing or fully shutting down** due to the automatic repair of partial failures.
- **System's Safety**: a system that is fault tolerant will also be safe, meaning that no catastrophic consequences will happen due to a failure, and this is logically because a system that has automatic repair itself from failures involves *recovery*: the remaining parts of the system that will handle the repairing have the necessary resources to go back to its correct operation, meaning that no data loss should be expected after recovery.
- **System's Maintainability**: if a fault tolerant system has automatic detection and recovery, it means that the system's failures are easily repaired, and the system is therefore highly maintainable, because the failure recovery does not need other resources other than the system itself.

Apart from these advantages, one that also arises is that a fault tolerant system also considers communication failures to tolerate, and therefore guarantees a **reliable client-server communication**. These systems often do transactions twice to ensure that communication errors such as timing or omission from the network are considered and do not harm the system. This also ensures that your system has **reliable message-handling components** deliver proper communication among your other nonfaulty servers to correct fault detection.

Finally, another advantage that is important to mention is that a fault tolerant system that recovers itself is usually a **replayable system**, due to the fact that most checkpointing that allows recovery is done through Message Logging. To have such system is an advantage in itself because in case of failure, the states of the entities can be restored or tracked, which means that its state is backed up, and this can be useful for further analysis or research purposes.

3 Security

Security is one of the most important principles of any distributed system. If one decides on taking the challenge of implementing a secure distributed system, other advantages come entailed.

The first can be called **a hierarchical structure of power** among the system parts. This comes as a consequence of Authorization, which ensures that every entity in the system gets the access rights it is entitled to. This is an advantage that can help protect from fatal errors that come whenever a system offers uncommitted access to either users or entities that could harm accessed resources.

The above mentioned leads to other advantage that is called **confidentiality**. A secure system guarantees that its information is disclosed only to its authorized entities, and any external party should be unable to understand or access unauthorized data. In this way, a secure system becomes one that **the user can trust** to deliver the promised services. Such system, also offers to have a high degree of **integrity**, where every action that is performed is sure to be done in an authorized way. This becomes

another advantage, since it means that a system has confident control, and detects and recovers from failures.

One of the most common advantages from security is that it offers **protection** for the service and user data from external security threats, such as undesired or harmful interception, interruption, modification and fabrication of the application data. This prevents private data from being illegally copied when intercepted by a third party.

By avoiding interruption, the system protects private data from being **corrupted, unusable or lost**, and therefore, security also offers **consistency and reliability** for the user and its internal parties.

A secure system uses *Auditing Tools* in order to trace action performed on the system, and this reports who accessed what and in which way. These type of access logging offers an enormous advantages as it makes **attacking riskier** because any action by this intruder is logged and analysed to take further measures and their identity is therefore exposed. A secures system, in other words, scares off attackers and reduces this risk for the future of the said system or other's.

4 Development, Test and Release of Distributed Systems

Testing your distributed system largely depends on what the system's purpose is. However difficult the testing task is, the advantages are immense, and is here when I consulted The Linux Foundation article[1] about testing-release importance:

First, automating tests developed before release **prevents regression** and makes **code reviews** easier among the team. Basically, a system that depends on development, testing and release makes the development of the future iteration simpler and analyse, fix or scale.

From another perspective, having the focus on this three things ensures **cost-effectiveness** by increasing system availability without having it to shut down due to insufficient testing, and therefore, unpredicted errors.

A distributed system, at its core, means that the system is not entirely run on one single computer. This also suggests that most distributed systems rely on applications on other servers, other external APIs or services. These services are naturally prompt to change, and sometimes they do so abruptly. Focusing the development on continuous test and release, **prevents abrupt external dependency changes from affecting the provided service.**

Next, having tests that simulate prioritized network failures **prevents your system from having poor user experience** during errors, which are the norm in distributed systems. Testing for errors helps build rich user experiences when these are inevitable, such as network errors.

5 Conclusions

It is clear that a distributed system has its own challenges when it comes to its implementation, but its advantages are enormous. The first advantage comes from the definition itself: you get to use more than one compuer's power to provide a service. But this was not the only advantage. In this essay I presented what I presumed were the main advantages of a distributed system that focuses on Consistency, Fault Tolerance, Security and Testing, and while these often represent challenges, the results answer why

the industry of distributed computing is increasing at such rapid pace: a distributed system with the mentioned focus produces a service that is *trusted*.

References

[1] K. Hughes. "Why Testing Is Important for Distributed Software". In: *The Linux Foundation* (2017). URL: `https://linuxfoundation.org/blog/testing-important-distributed-software/`.

[2] M. van Steen and A.S. Tanenbaum. *Distributed Systems*. distributed-systems.net, 2017.