Christoph Bussler
Aug 1 · 3 min read · Listen

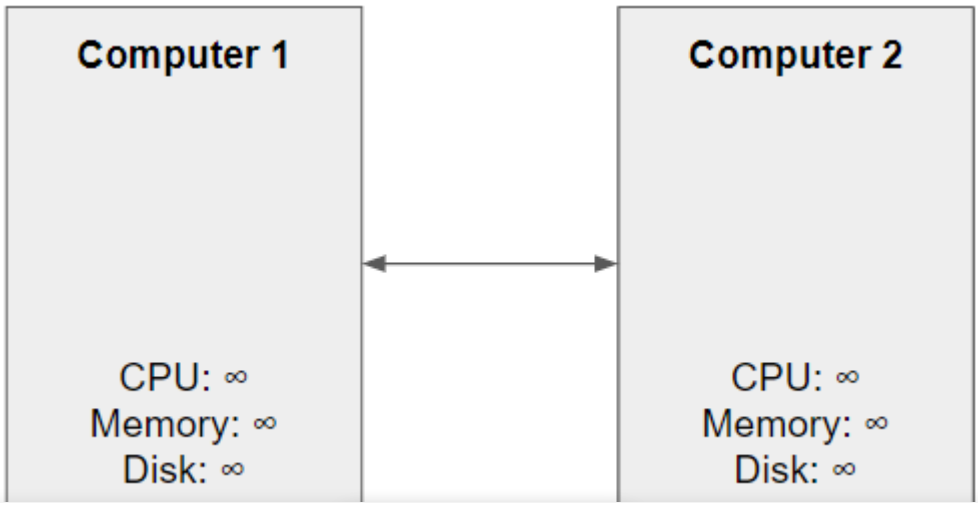
Two infinite networked computers — when would you use both?

A brief software architecture thought experiment

Regularly I come across the assertion that structural distribution of deployment units or resources is inherently beneficial and can be used as a basic base pattern in the architecture design process. As a consequence a distributed architecture does not need rationalization or justification, or so I am told. I disagree.

Thought experiment setup: two infinite networked computers

Let's assume two independent computers, each having infinite CPU, memory, disk. Both are linked over an infinite network connection (for simplicity infinite bandwidth, and speed of light latency — the latter a restriction that I would not propose to lift, even here :-)). Figure 1 shows both in a diagram.



The architecture and design approach on this infrastructure does not have to consider many limitations, if any.

The software design and architecture process can fully concentrate on

- **Abstraction**. Design functionality and components with a clear understanding of the functionality exposed by public access interfaces and private internal behavior.
- **Extensibility**. The design can consider making functional extensibility easy for future changes in requirements.
- **Efficiency**. Not having such systems, we cannot be sure if efficient algorithms will make a difference on such a system compared to inefficient algorithms. So let's assume that efficiency is a good property to consider, and avoid premature optimization until inefficiencies are known.

This list is incomplete and can be extended with more aspects, also depending on your background, experience and technology focus.

Basically, a designer, architect, or engineer concentrates on functionality and ease of future extension of the system. The fact that systems always need additional functionality over time in most cases is a "law" of software systems; so planning for that is good practice.

When would you use both computers?

Since the two computers are linked by a network, when would you consider designing the software architecture so that both computer systems are part of it?

My initial set of responses would be

- **Assurance/availability**. Due to power outages or component failures it is possible that a computer becomes unavailable. If availability is an important SLA that you design for, the ability to switch access traffic to an alternate system is necessary. A second computer, synchronized with the first, supports this. This requires that the

- **Geo-location data residency policies**. Another case are external requirements or restrictions that are mandatory for the architecture design process. For example, some jurisdictions require specific data (determined by a predicate like “citizen of country c”) to be within a specific geography (e.g., a country). In this case computer systems must be placed such that these predicates can be true.

Aspects that are not drivers are (a) **scale,** since one computer has no scale limits, (b) **high-availability,** because the second computer can take over immediately, (c) **concurrency**, because one system has infinite resources. There are for sure additional aspects that can be added to this list.

Summary

This brief thought experiment aims to show that distribution has to be rationalized and justified depending on the characteristics of the available resources. Each time a part of the overall architecture is distributed (aka, made independent of the remaining architecture while having to be coordinated with it at design and runtime), benefits are realized, but also additional difficulties to be overcome by additional implementation like for example consistent synchronization.

Please ping me if you have feedback or input.