Distributed Systems

Djangothon’18

@tapaswenipathak

tapaswenipathak@gmail.com

Distributed systems are independent connected components

What problem are you trying to solve?

A good solution is slower and harder to write than synchronous systems

Can there be a perfect and scalable solution?

Reasoning about the knowledge of a group and its evolution can reveal subtleties that may not otherwise be apparent, can sharpen our understanding of basic issues, and can improve the high-level reasoning required in the design and analysis of distributed protocols and plans. - Knowledge and Common Knowledge in a Distributed Environment

This paper presents a design principle that helps guide placement of functions among the modules of a distributed computer system. The principle, called the end-to-end argument, suggests that functions placed at low levels of a system may be redundant or of little value when compared with the cost of providing them at that low level. – End to End Arguments in System Design

Are you using an appropriate algorithm?

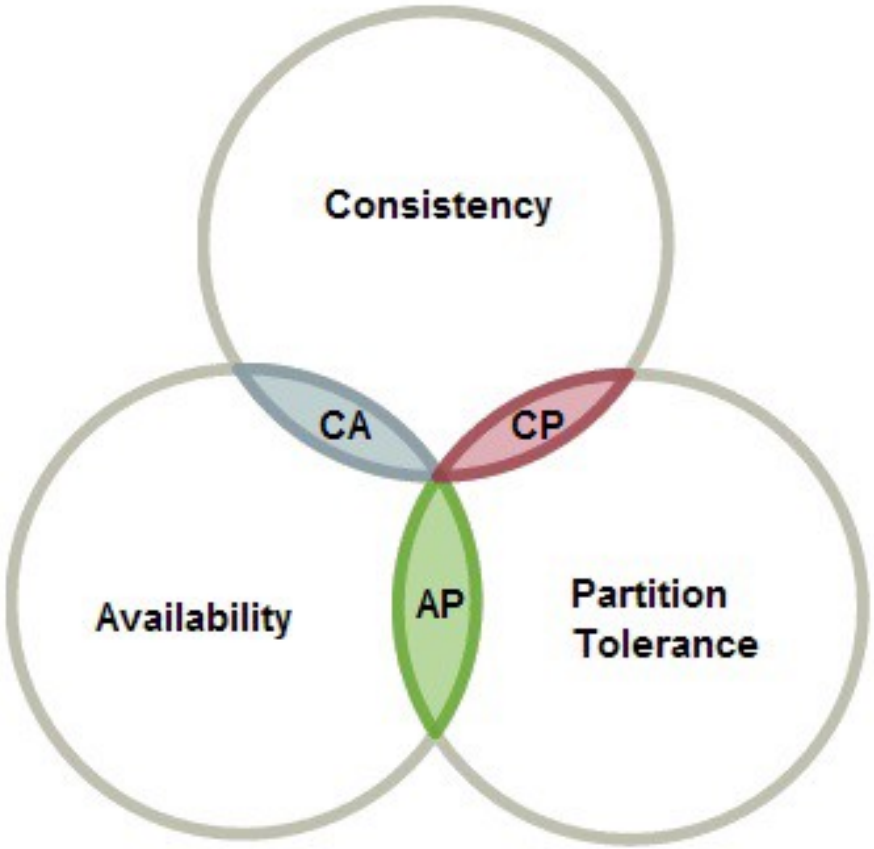
There are numerous examples of scalable algorithms and computational models; one only needs to look back to the parallel computing research of decades past. Boruvka's algorithm is nearly ninety years old, parallelizes cleanly, and solves a more general problem than label propagation. The Bulk Synchronous Parallel model is surprisingly more general than related work sections would have you believe. These algorithms and models are richly detailed, analyzed, and in many cases already implemented. - Scalability! But at what COST?

Consistency is an important consideration in computer systems that share and replicate data, data sharing and replication raise a fundamental question: what should happen if a client modifies some data items and simultaneously, or within a short time, another client reads or modifies the same items, possibly at a different replica? - The many faces of consistency

Replication is making copies of the same data on multiple machines; this allows more servers to take part in the computation. - The book Distributed systems: for fun and profit

To replication! The cause of, and solution to all of life's problems.

CAP Theorem



Notes on Distributed Systems for Young Bloods

- If you can fit your problem in memory, it's probably trivial.
- Metrics are the only way to get your job done.
- Use percentiles, not averages.
- Learn to estimate your capacity.
- Computers can do more than you think they can.
- Use the CAP theorem to critique systems.
- Writing robust distributed systems costs more than writing robust single-machine systems.
- Robust, open source distributed systems are much less common than robust, single-machine systems.
- Coordination is very hard.
- Implement backpressure throughout your system.
- Find ways to be partially available.

c@Something Similar, Jeff Hodges

A single center is not always bad if the alternative is multiple fragile and weak ones; bad if that is not the case, you may end up not solving the problem at all

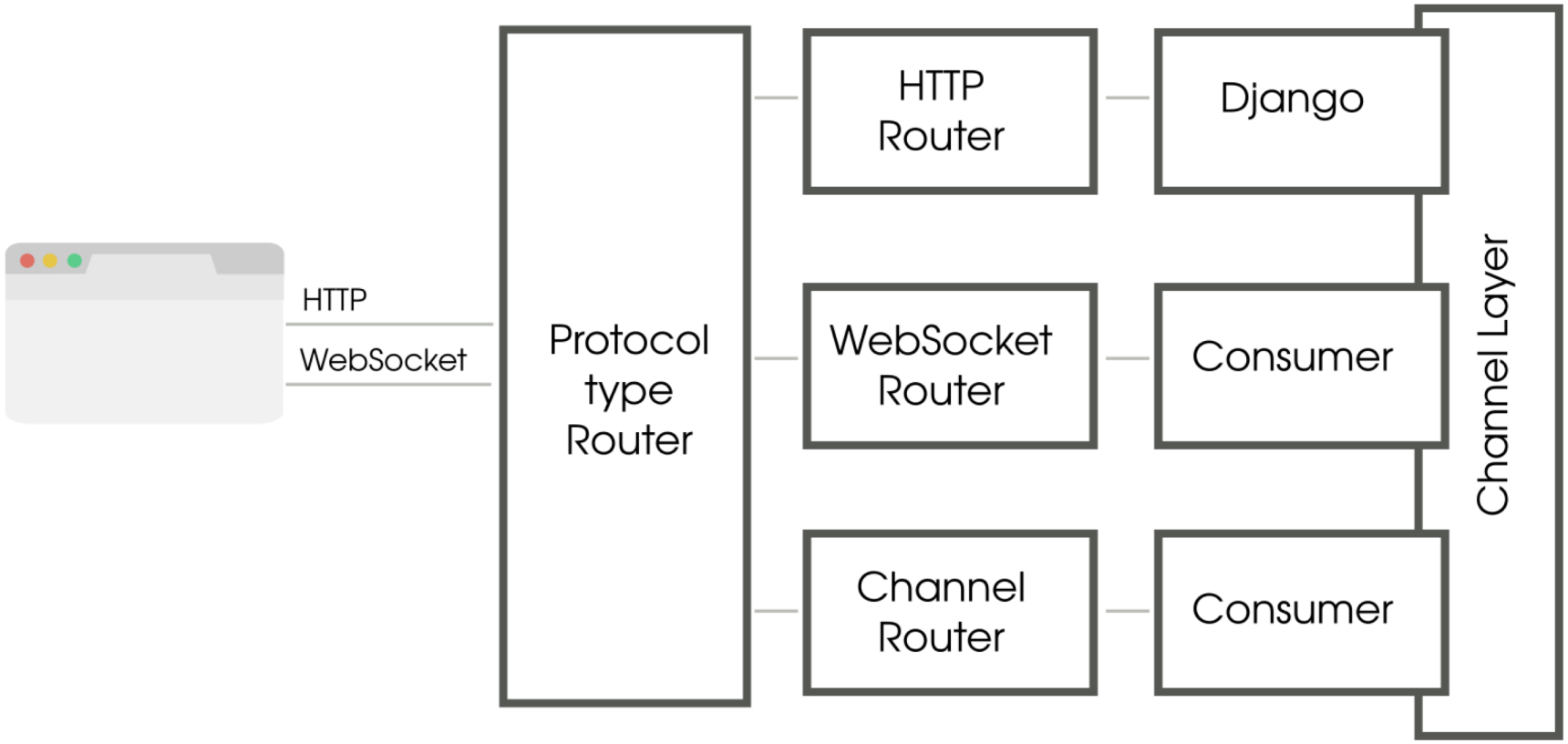
Ideate more, design more, own and understand what you are making

Django

Channels and ASGI provide a message bus built with few tradeoffs

Django Channels

- A **channel** is like a pipe. A sender sends a message to this pipe from one end and it reaches a listener at the other end. A **group** defines a group of channels who are all listening to a topic.



Questions?

Thanks!