IPFS

It is a peer to peer merkle-dag based, versioned permanent web. An alternative to http. The uri includes the hash of the content of the file you're storing and this is mind blowing! Because then the url itself is like a security mechanism, i.e. you can immediately check if the data you received from a peer is what you had requested. If the hash of the content matches the hash in the url, it is correct, otherwise it isn't!! Simple!! This is worlds apart from a normal http url where there is no constraints on the url, so the server can serve you anything from behind a url and you would have no way to know if it has changed due to network error or because of an actual attack. This can be applied to blockchain storage due to the determinism and the immutability requirement of blockchain. But also, it will find applications in simple distributed applications even if you aren't necessarily worried about malicious actors, but just transmission errors.

According to IPFS.io:

“HTTP is inefficient and expensive  
HTTP downloads a file from a single computer at a time, instead of getting pieces from multiple computers simultaneously. With video delivery, a P2P approach could save 60% in bandwidth costs.”

Other benefits they claim are decentralization( which provides opportunities to all players - big and small), immutable (versioning) of any resources you share on IPFS, persistent availability without necessarily needing an internet backbone.

So what exactly is needed to make this compete with web 2.0?

<https://youtu.be/ewpIi1y_KDc?t=1020>

So while there are a lot of scalability issues, there are things that can be provided that couldn't have been ever before:

1. Trustless verifiable computation
2. Applications that aren't supported by a company but just by the network (what is the incentive? Smart contracts?)
3. Web 2.0 relies on advertising - that's the entire business model and so they can't do end-to-end encryption. Dapps don't have that need. The incentive structures are different.

Here's a useful video:

Explanation by example: <https://youtu.be/BA2rHlbB5i0>?t=100

-Bandwidth

-Latency is another problem this will solve because it brings the servers closer to you.. edge computing or cdn

-need for persistence of web content(prevent broken links 404s)

-Centralization

Basically, instead of a single server delivering everything, you get served by a network of multiple peers. So obviously this makes it cheaper and faster. So, now why haven't we been doing this all along?

The first problem was fault tolerance and integrity of data.. how do you trust your peers on a decentralised network? Ipfs solves this problem.

The second problem is availability - Filecoin is trying to solve this problem.

Merkle DAG usage:

Merkle DAGs are used in git for version control. IPFS used them for content addressing. Essentially, the merkle DAGs represent a folder hierarchy, in which each hash points to a subfolder (which is also a hash) and finally to a file. This key value pair is stored in a distributed hash table.

IPLD: the data structure for ipfs

So json is a standard for exchanging data between systems in a language agnostic way. It doesn't however give any semantics to the data.. in order to give semantics meaning to content.. a merklized version of this data can be used to access the object graph associated with some content in a semantic way.. each of the data is merkle hash pointer to another piece of data.. So you directly link to content in a context independent way without needing to drive semantic meaning based on the context. So the key idea is content addressing. Everything is directly addressed using an ipfs address which consists of the hash of the content at that address. This is important for several reasons:

1. It allows for content addressed object graphs.
2. Since data is being addressed based on content and not location, it will find the nearest peer that has the data to serve the request instead of always going all the way to centralised service's nearest server.

[https://youtu.be/ewpIi1y\_KDc?t=1320](https://youtu.be/ewpIi1y_KDc?t=1020)

Questions:

1. Each network node stores only content it is interested in, and some indexing information that helps figure out who is storing what
2. How would you make a search engine (keyword indexed search engine for ipfs)