A distributed publishing system

Leonhard Horstmeyer

March 7, 2023

Abstract

1 Introduction

With the technological possibilities that are at our disposal, such as the vast spectrum of data structures, of query and storage systems, of versioning and networking tools, we can create a publishing system that fits our needs. Instead our scientific output still behaves like physical paper in the way it's treated and guarded. It is as though the printing press had been used to create exact replica of handwritten texts rather than using movable types. One could rather start from requirements of a publishing system and reconcile them with the available technology. In this paper we provide one such system, whilst acknowledging and demarcating it from other proposals. We develop the specifications of a distributed publishing system that satisfies certain desirable requirements to a sufficient extent. We also discuss some possible shortcomings.

2 Elements of the Publishing System

The proposed publishing system is a distributed multibranch version control system. We first provide a high level overview of the elements and then discuss them in more detail, (starting from the requirements that we aim to satisfy.)

2.1 Overview of the Elements

What is the structure of content? Rather than relying on a folder structure, as is the case in git[**?**] or unix[**?**], we propose to build the publishing system on content buckets, called *atoms*, and context buckets, called *molecules*. For all practical purposes one may think of atoms as the smallest unit of published content such as captions, paragraphs, formulae, arguments, media files, data sets etc. Small chunks of content allow for better reusability and updateability. Molecules on the other hand can be thought of as content containers such as review articles, book chapters or monographs. They put atomic content into a context. Atoms can be part of multiple molecules, which is not possible in a folder structure.

How is new content submitted? It is submitted in terms of changes, rather than in standalone bulk. A submitted change is called a *submit*. A submit alters and references a previous submit, the *parent submit*. Every submit is comprised of changes to the individual atoms and molecules and has a unique history of submits that includes itself and all of its parent submits. This history is is called a *branch* and uniquely identified by its latest submit. The content of one branch may be included into that of another branch, a directional procedure referred to as a *merge*. Branches may have conflicting content. When attempting to merge one branch into another with conflicting content a *merge conflict* arises that should be *resolved*. The overall structure of submissions will be an arbolescence[1], a directed tree called the *submitree*. (Making conflict resolution manifest)

Likewise, at the level of atoms and molecules, there is a linked data structure. The atoms and molecules are arbolescences, called *atomic submitrees* and *molecular submitrees* respectively. They tend to be less branched, since not every branching submit alters every single atom or molecule. We add (better formulation) a remark on the design choice of submitrees at the bucket level: Once we allow pieces of content to be reuseable and therefore identifiable and addressable, any branching or merging has to keep track of these bucket identities. However, since there is no preferred branch, we consequentially need to branch (whilst preserving the identity) also the altered buckets and only those.

What is the data model for this structure? Like with git and many decentralized protocols, we propose to use a data structure that allows for easy and fast resolution of data using content addressing via cryptographic hashes.

What is the relation between submits and branches at the level of atoms and molecules? One may think of content and context buckets, i.e. atoms and molecules, as

Data model .

Every submit is identified by a submit hash. We explain the calculation of the hash as well as the resolution of content through the hash. To this end we first note that every submit is uniquely determined via the reference to the parent submit *and* a snapshot of its content. The snapshot of a submit's content, which we call a *section*, is a collection of the atomic and molecular snapshots of all the atoms and molecules that make up the content. They include the snapshots of the changes in the current commit, but going back in the branch history also include the snapshots of other atoms or molecules changed in previous submits. O

that have been added in the submit history of the branch. They also include those snapshots of the changes in the current submit. In order to cryptographically secure the content against tampering one may calculate a hash from the snapshot.

whereby we mean is a snapshot of its content. is uniquely determined by the atomic and molecular snapshots

[1]For information about the graph-theoretic term of an arbolescence refer to (TODO).

The submit hash is calculated as the Merkle root of all the atomic and molecular snapshots that comprise its content. Those are of course the ones that were altered in that particular submit, but moreover also all the other It can also resolve the

How is data stored?

have just one universal type of container.

Contrary to *git* (or the *unix* file system) there is

2.2 Rest

This paradigm facilitates a more continuous and organic content evolution compared to article or monograph submission, but at the same time contains those as special cases.

conflicts become manifest

It is more akin to the content growth in

that encodes the change a is not a standalone article, but in the format of a change. with reference to a branch, that is a unique history of changes.

Published data is stored in a distributed hash table and addressed via an IPLD encoded hash linked data structure.

2.3 Data Structure and Data Storage

Like other versioned data structures, such as *git* or *ceramic*, we propose to use content identifiers (CIDs) to reference data and to use Merkle DAGs to aggregate CIDs.

Data is either content bearing or referential data, i.e. data that references or addresses data.

Like most distributed systems, we propose that data is stored by content addresses We describe now the proposed data structure and data storage.

We propose a system where research content is grown continuously on a sharded database. *local consensus*

2.4 The state

There are branches and there are

2.5 Submission

enemy of science is personal agendas. single ... interests as gate keeper

try to avoid ideas from being gated away...

A *submit* is similar to a commit in *git*. It contains a pointer to the parent contains a number of

2.6 Storage

2.7 Consensus

The tasks and purpose of editors and reviewers are all subsumed under the local consensus algorithm.

3 Requirements

In the first scenario, we keep a single book of scientific ”truth”. Everything that has been found true for ever after should be written into that book in a cumulative and immutable way. Two questions immediately arise: Who writes and what is true? If a single person or entity writes, then where does this entity take its legitimacy from. If more than one person or entity write, then how can they find a consensus and how can a consensus even be guaranteed in the first place? Moreover, if truth was so easily established, they why are scientists that divided with respect to their results and why do seemingly consensual results get overthrown again and again. It will be hard to argue for this scenario as an appropriate system to produce scientific output. Yet, this system is one that is very much in line with a popular conception of science as an linearly growing accumulation of truths. A positive feature – if any – is its clear direction of progress. In order to implement such a system, the alleged truths could be written into a single database and the mighty author would be a generally trusted person or entity. Alternatively, if writing was done through a consensus algorithm, one could implement this scenario as a blockchain.

Let us now consider another scenario on the other end of the spectrum. Results are being published in lots of small, isolated and self-contained blobs without any agreed consensus mechanism how to bring them in agreement. The question that can be raised, is how discrepancies are resolved and who resolves them. As was the case for the former scenario, this one is also hypothetical and likewise not an appropriate system for publishing scientific output. Yet, one can argue that this system features similar aspects to the one we currently witness, which has different kinds of consensus mechanisms regarding the acceptance of research output, that are nevertheless not always overt. This scenario has several drawbacks. There is no clear path of resolving discrepancies. One major advantage is that pluralism of research outputs is built into this scenario.

A publishing system should live somewhere in the middle ground from these two scenarios. The system should allow content to be grown dynamically, discrepancies to be made visible, consensus mechanisms to be overt and pluralism to be manifest.

4 Distributed Publishing System

Everyone may connect to the network and send a *submit* into the network.

4.1 Data Structure

The smallest unit is a *blob*[2]. There are two types of blobs, those that carry content and those that carry context. We call the content blobs *atoms* and the context blobs *molecules.* A typical content blob would be

5 Rest

Each technology

We are free to choose our form of

Progress in or of science has ever since been a field of inquiry. Mapping the human undertaking of acrueing knowledge

There have been many metaphors, that have been worked out to greater or lesser extend in an effort to map this undertaking.

"Wissenschaft", "Fortschritt", "advance" (directional).f

These metaphores range [3] form cumulative processes to growing and waning socio-epistemic ecosystems. Even though it is not fair to speak of metaphores of one and the same thing here. The cumulative approach to science is different in terms of ontology and outlook from the Lakatosian for instance.

The interesting thing is that there is such a strong mismatch of the perceived metaphor from the operationalized one. From conversation with people and reading of articles it becomes apparent that more or less science is progressing cumulatively. Yes, people know that there are various schools, that see things differently, but somehow there is this one thread of science, where the loose ends are tied together. This is a common view. On the other hand we have an actualization of our research in research paper publications, reviews and monographs. So more like a set of beats loosely attached to each other. It is this divergence that we would like to address.

This process is mapped to our publishing methods.

Given our current technological means it would be fair to raise the point whether we can map

How could we use our current technological status to build a new science? A thought experiment.

6 Data Architectures ((of other knowledge systems))

6.1 git

At its heart git's data model is a Merkle tree. git is a merkle tree. It

[2]This terminology is borrowed from *git*.

[3]We don't intend to imply a one-dimensionality.

6.2 ethereum

6.3 wikipedia

6.4 traditional publishing

7 A survey

The way that science is done depends on the incentive / reward system as well as the output system. Although it is surely not right to disentangle them, we just do it nevertheless and cut away a little bit of the complexity involved. We want to survey a few extreme cases. We want to give a very high level, topological analysis, rather than a technical low level analysis here.

Why not be brave and turn this into practice.

Perpetuation:

- Verbal communication
- One single book with millions and millions of chapters
- Millions and millions of papers

Incentives:

- Names associated to contribution
- Citations
- The greater good of humanity
- social prestige ...

8 Literature

In [**?**] ... they propose that the reward system comes from cryptocurrency. However any reward solution should come from within the framework, otherwise all the advantage of decentralization is lost.(p.4637) ... Who is the editor, who will invite, who is invited.

"In the event of a reviewer sending a review when the time has expired, a penalty is applied to the reviewer's reputation in the reputation system." This is a very hardcoded and not emergent border.

Blockchain is for absolute values a wonderful technology. The problem that blockchain solved is the

9 A commit network

No single truth build in!

One single ledger. Many branches. Different kind of activities. Contributions are commits.

Advantange: If one wants to rely on another result one would have to find the diff.

It should allow for the plurality. Where would you want to draw the line between crack-pottery and science. It's hard. Well don't. No one can and no one should. There will be branches that are more main stream and those that are less. They should not be censored.

Who should keep the entirety of the wisdom? There is really only one answer here: Everyone who participates. Knowledge is too precious to have it on a server or to have it in the hands of a few.

There should be somethin like a local consensus. A consensus that is directed at a branch. So every branch is like a block chain. Consensus on that branch is like a pull request. Then there can of course also be merges and forks.

As a new-comer you can either fork off somewhere or you contribute somewhere.

Every participant of one branch has only their branch as a blockchain.

Question: What about tempering. How easy is it to troll the network? Any troll should be able to open a new branch, why not (but somehow you wouldnt want that to go into a storage that is shared by everyone, otherwise it would be an attack vector for DoS).

Lets make the terminological distinction between stem (like fork) and branch. A branch can be merged into a stem by pull request and consensus, but a stem cannot be merged simply into another stem without the consensus of their community leaders.

One can build a new stem by cherry picking other stems. Every stem keeps its data. That prevents trolls from spamming the entire network.

Some archival nodes can choose to select which stems to display. A user can then choose between one of the versions and also get the diff!

Any node can always be queried of course, but there is a threat that you will be lost in oblivion. But actually this threat is what we are anyway faced with. Something doesnt get published. It will be lost in oblivion!

ANd this system would lend itself ideally for all sorts of reputation schemes. Such as anonymizing the network or commits will emerge as something reputable.

Pseudocode:

10 Protocol

10.1 Objects

There are three types of objects: atoms, worldlines and contexts. Atoms can be thought of as the smallest unit of a submission, e.g. paragraphs, data items or media files. A worldline is a history of atoms that are connected through deltified updates. They can be thought of as the various versions of an atom.

A context provides a higher-level semantic environment for the worldlines. One may think of a context as a paper, a wikipedia entry or a collection of data. All atoms have the same structure and all contexts have the same structure. There is no difference between an atom that holds an image file or an atom that holds a comment.

Atoms contain metadata, data and links. Let A be an atom. Its metadata is made up of an identifier id_A, a timestamp τ_A and a pointer to a data decoder $^\star dec$. Its data is bytes-encoded. The link is an ordered list of atom identifiers.

The context contains metadata and and atoms. There must always be at least one atom, that accounts for the organisation or the arrangement of the context. We call it $O_{\mathcal{X}}$. Then there is an arbitrary amount of atoms $A_{\mathcal{X}}, B_{\mathcal{X}}, \dots, Z_{\mathcal{X}}$

10.2 Opcodes

Opcode	**args**	**reference**
CREATE_ATOM	cell2	10.2.1
UPDATE_ATOM	cell2	cell3
CREATE_CONTEXT	cell2	cell3
ADD_CONTEXT_TO_ATOM	cell5	cell6
REMOVE_CONTEXT_FROM_ATOM	cell8	cell9

10.2.1 CREATE_ATOM

A new atomic item is created. As arguments it takes the pointer to the context $^\star ctx$.