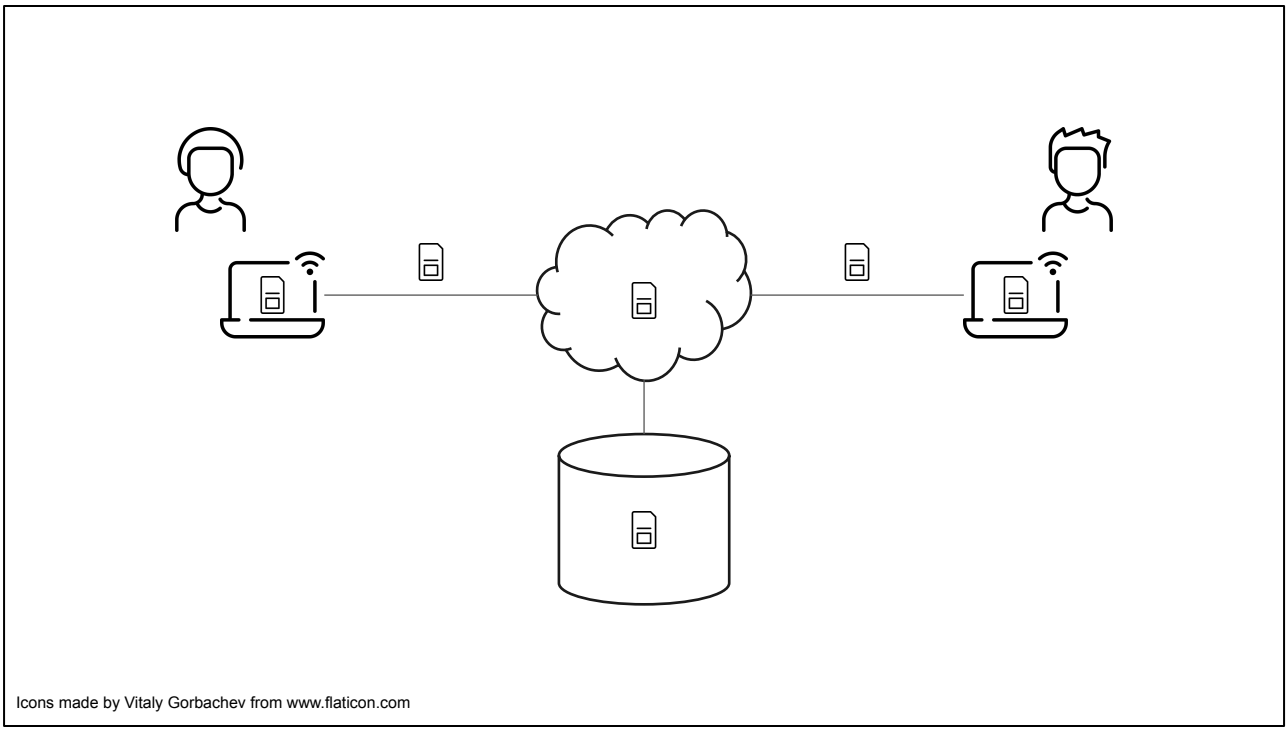
Live Shared Linked Data

I'm George Svarovsky (george@m-ld.io). I've been working on a project called m-ld. It's a software component for live information sharing, and it uses Linked Data as its data representation.

I want to talk about *sharing* linked data to multiple collaborators at the same time.

Today I'll introduce m-ld to you, relate it to Linked Data, and talk about the big idea behind it which has excited me all along; and the direction we're going next, with the help of NLnet and the NGI.



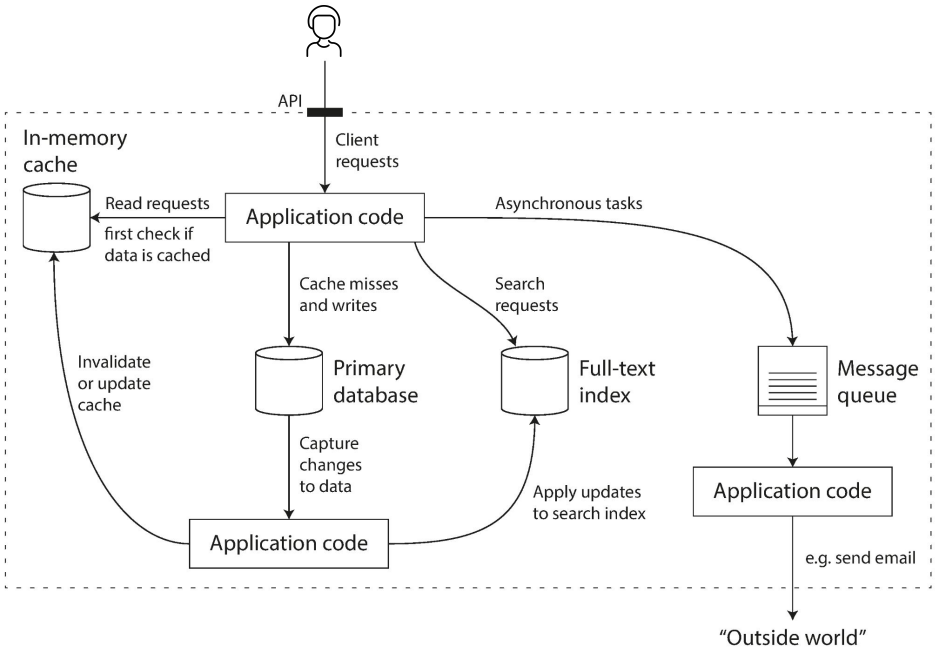
Icons made by Vitaly Gorbachev from www.flaticon.com

Let's start back in the last century, with people using applications on their local machine. They're manipulating information, locally. One edit follows another. Every now and then they hit save and push the information to their local disk as a file.

But our user wants to share their information with other people. They could publish a web page. Or they could send the file to someone else. But doing these things creates new copies of the information, which creates a new problem. Every time anyone changes the information, they have to somehow update everyone else.

So instead of making copies, we're going to put the information in one place, a database. We typically have some middleware, maybe in the cloud, to make the data visible to all the participants. Great! Problem solved.

Only, it isn't solved at all. This picture is an illusion. You can't look at data in a database. What you still do is create copies of the information on your local machine to look at. And getting it to your local machine means distributing it, via the middleware and the network. Instead of a few copies, we have lots of copies.



Designing Data-Intensive Applications. Kleppmann 2016

Here's inside that cloud. We engineers spend a lot of effort moving copies of information around. And they're all in different formats – we've got database rows, JSON, object-oriented objects, text indexes, all kinds.

But what's the real heart of the problem here? I want to argue that the real problem is not that we have lots of copies. We have to have those copies, because you can't see data that's far away.

The problem is that each copy is static, and we have to work really hard to bring it to life, which means updating it and sending out updates to it. The labels on the arrows in this picture give some idea of the complexity involved.

And note that it's the application's responsibility to implement all of this. There are plenty of great off the shelf components for databases and caches and text indexes, et cetera. But making them all work together and present the data is down to you, the application developer.

a tale of two woes

Applications don't talk to each other well

- Linked Data provides **base syntax** and **extensible semantics**

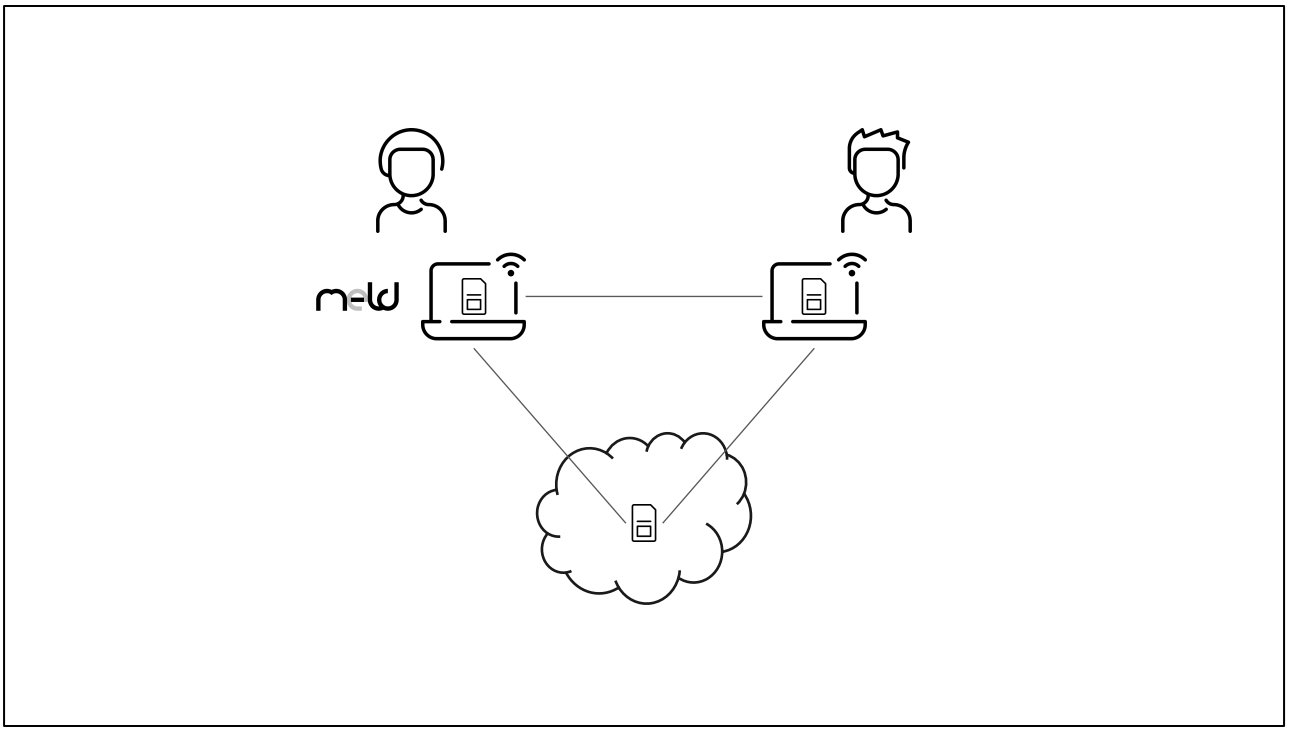
Applications don't even talk to themselves well

- Shared Data provides **base truth** and **extensible distribution**

Let's relate this problem to Linked Data. Linked Data gives us extensible and machine-readable meaning for information. That helps immensely with the data format problem.

What I mean to say, is that Linked Data lifts the baseline of data representation. Instead of binary, it's semantically-rich human- and machine-readable triples.

What if we could also bring the data to life? Instead of having static data as the basis of truth, let's have live shared data as the basis of truth; and an extensible way to distribute that data. Again, I want to change the baseline.



Let me illustrate what I mean. Here's the user again, with her local information, and again, she wants to share it with another user.

The principle of m-ld is that the chunk of information is inherently live and shareable. So to share it, she just shares it directly. And the other user can edit it too.

In this model, all that she needs, to do this, is a network connection. There's no server or database holding a master copy of the information. Something (m-ld) is holding these two copies in sync with each other.

Of course, there's interesting computer science and engineering going on behind the scenes, but the important thing is that it really is behind the scenes. The app itself is built on top of inherently live data, instead of inherently static data, and does none of the synchronisation itself.

Now, one thing you might wonder is whether this is as resilient and secure as the cloud app we had before. Well, in some cases it might be better, for example because we don't have that cloud service any more, or a central database. So if Alice and Bob have a private channel between them, then the information doesn't have to ever be visible anywhere but on the devices that belong to them.

Resilience is harder to square – what if Alice and Bob both drop their laptops in the bath?

But remember, the data is live. There is nothing to stop the application developer from putting it on the cloud as well, if that suits the application better.

```
it('transacts parallel list inserts from two clones', async () => {
  clones = await Clone.start(2);

  await Promise.all([
    clones[0].transact({ '@id': 'shopping', '@list': ['Bread'] }),
    clones[1].updated('shopping')
  ]);
  await Promise.all([
    clones[0].transact({ '@id': 'shopping', '@list': { 1: 'Spam' } }),
    clones[1].transact({ '@id': 'shopping', '@list': { 1: 'Milk' } }),
    clones[0].updated('Milk')
  ]);

  const shoppings = await clones[0].transact({ '@describe': 'shopping' });
  // => e.g. [{ '@id': 'shopping', '@list': ['Bread', 'Milk', 'Spam'] }]

  expect(shoppings[0]['@list'][0]).toEqual('Bread');
  expect(new Set(shoppings[0]['@list'])).toEqual(new Set(['Bread', 'Milk', 'Spam']));
});
```

Let me give you a hint of what m-ld looks like to use. This is a snippet from the compliance test suite that checks an implementation of m-ld on a particular platform (like Alice and Bob's laptops).

Most of the code on this slide is test fixtures, but you can see the API being used from the test service to poke operations into two live copies of the data, called 'clones'. As it happens, both of these clones are local, but of course in reality each clone would be inside an app, separated by a network.

The first thing to note is that m-ld uses JSON-LD, which is a serialisation of RDF. So all information in m-ld is Linked Data. You can also see that it's stretching JSON-LD to encompass making changes to data (here, inserting items into a list at an index, concurrently on two clones), and also querying data. These are extensions which I'd love to talk more about one day!

m-ld is available now as a developer preview; there's documentation, and a demo, and a playground, and a couple of starter projects.

Securing Shared Decentralised Live Information with **m-ld**

https://github.com/m-ld/m-ld-security-spec

Who 'owns' shared information?

Who has modified it?

Who owns its schema?

Who has attested its truth?

Who owns its access control list?

A moment ago I pretended that all there was to security, was privacy. Of course, it's not that simple. So thanks to NLnet and NGI Assure, I'm now working on a project to look into securing live shared information in a lot more detail.

In particular, if the new baseline for information is 100% shared and live, and anyone we share it with can make edits to it (in principle), how do you make sure that only people with best intentions do so? Clearly some level of authorisation controls are needed. But who sets those controls, and how?

Obviously not all data is really created equal. Changes to schema information and access control lists can have severe impacts on the information they relate to. Should these also be live and shared by default? I argue yes, they should.

And of course, we also want to keep identified people and machines accountable for the changes they do make. If data is always live, we need ways for people to sign their name to information, or agree to it, but not necessarily have to sign their name to every future version of it. I hope you agree that many of the answers will be in common with existing ways of securing information, and will use tried and tested techniques like cryptography.

But as other speakers have said, we know it's too easy to rely on opaque data app providers who may not actually share our values, who take control of our communication. I think that partly this has arisen because of the sheer complexity of building apps that distribute information. If instead, shared information is the baseline, and so the apps of the future are that much easier to build, maybe the controls we place on information will be easier for us to manage, not harder.

https://nlnet.nl

https://www.ngi.eu

https://m-ld.org