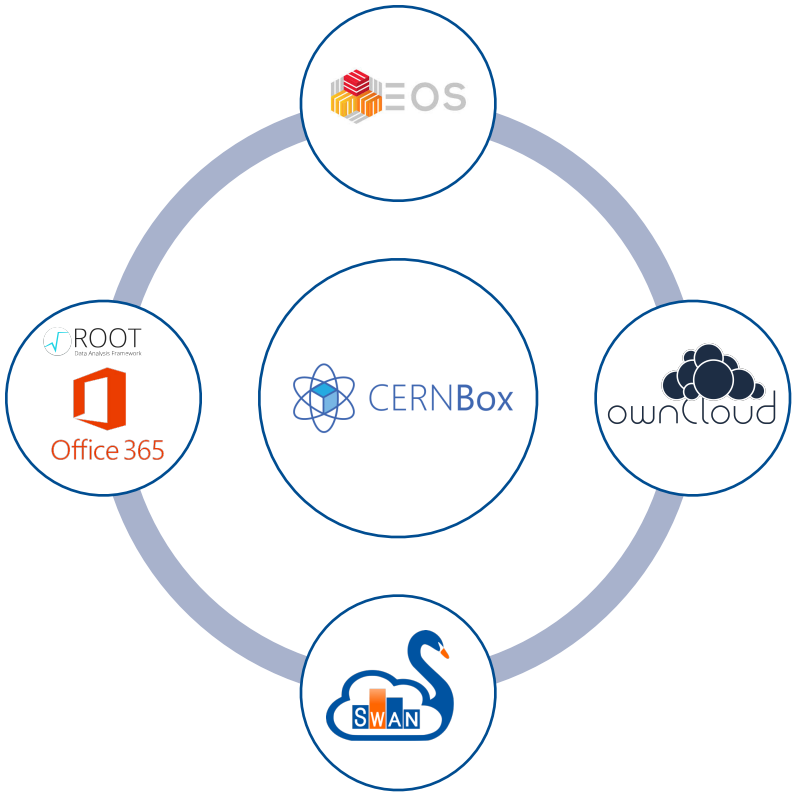
Making Reva talk to EOS
Ultimate scalability and performance for CERNBox

Fabrizio Furano
Hugo Gonzales Labrador
Ishank Arora
Samuel Alfageme Sainz
CERN IT-ST (Storage group)

What is CERNBox

- CERNBox provides a cloud synchronization and sharing service
  - Available for all CERN Users (1TiB/ user)
  - Synchronize files (data at CERN) and offline data access
  - Easy and convenient way to share data across users and groups
  - All major platforms supported
  - Based on ownCloud, powered by EOS
  - Integrated with various productivity apps
  - Going through important core evolution steps



EOS

- EOS is a disk-based, low-latency highly scalable storage service, managing many hundreds of Petabytes at CERN
- Having a highly-scalable hierarchical namespace, and with data access possible by the XROOT protocol, it was initially used for physics data storage and massive data access
- It also supports a subset of HTTP/WebDAV data access, optimized for performance
- It also has a gRPC interface, for metadata access and commands
- Today, EOS provides storage for both physics and user use cases, instances of EOS include EOSHOME, EOSATLAS, EOSCMS, EOSATLAS, EOSLHCB and of course CERNBox

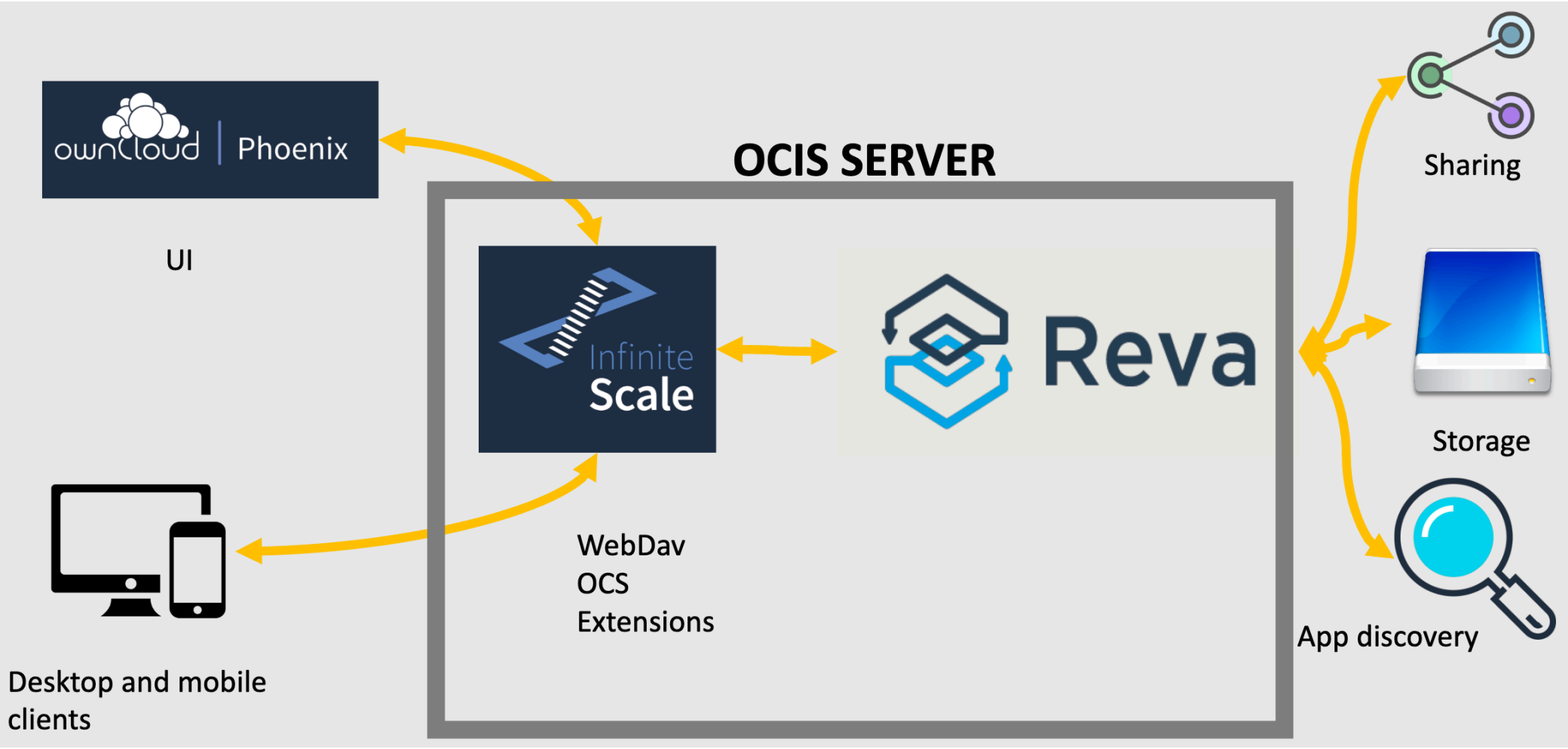
REVA

- The Reva project aims to make cloud storage and application providers inter-operable through a common platform
- The goal of the project is to offer a straightforward way to connect existing sync and share based services in a simple, portable and scalable way. In order to do that, it leverages the CS3 APIS

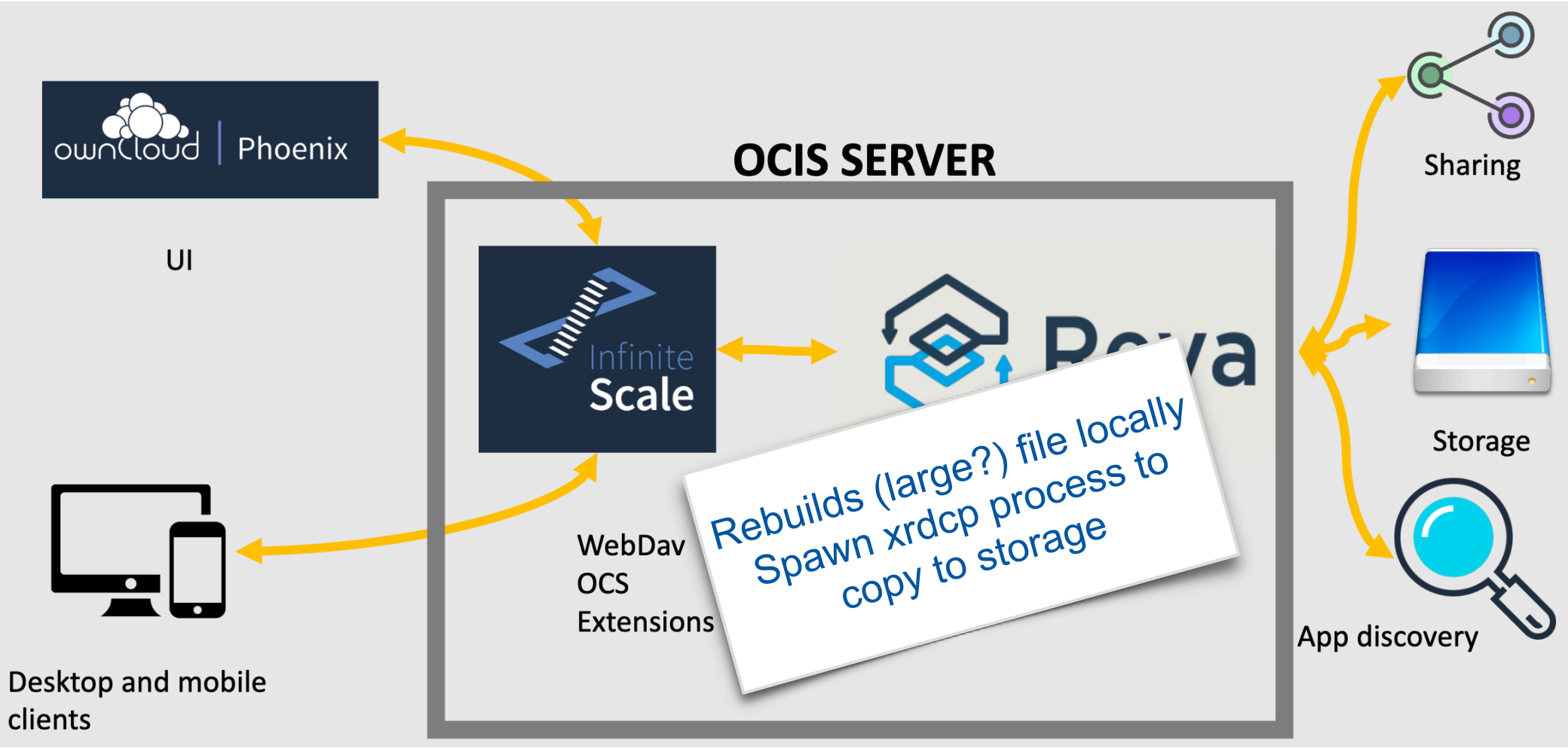
Scalable gateway mode

- In practice REVA is the component that adapts the APIs of the sync&share clients to the APIs of the background storage and metadata services
- We can also see this as a sort of highly configurable gateway
- It becomes more akin to a gateway especially when it starts managing the data (on top of metadata or redirections), on top of metadata
- This is the direction chosen for the evolution of REVA, in particular when EOS is the chosen backend like at CERN

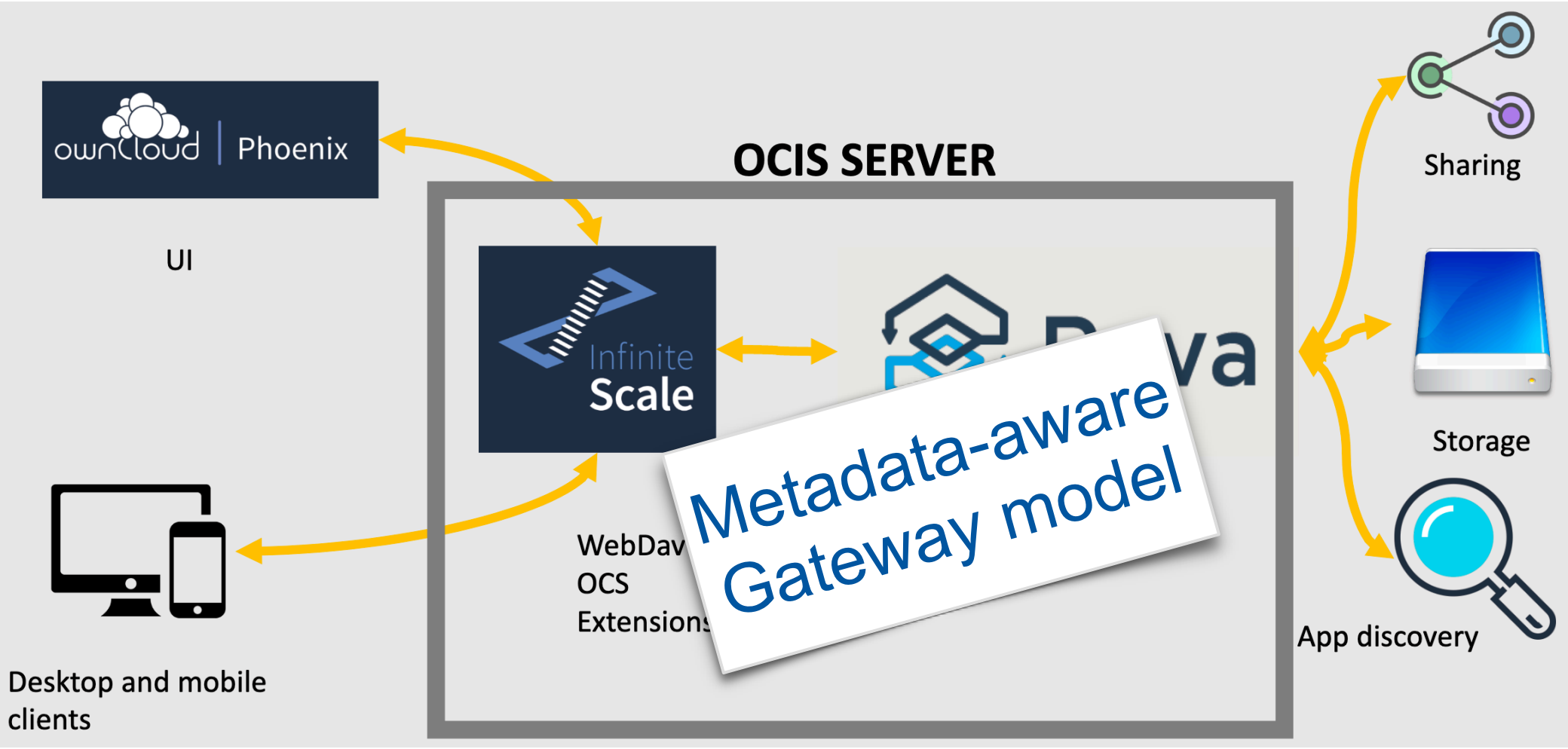
The evolution



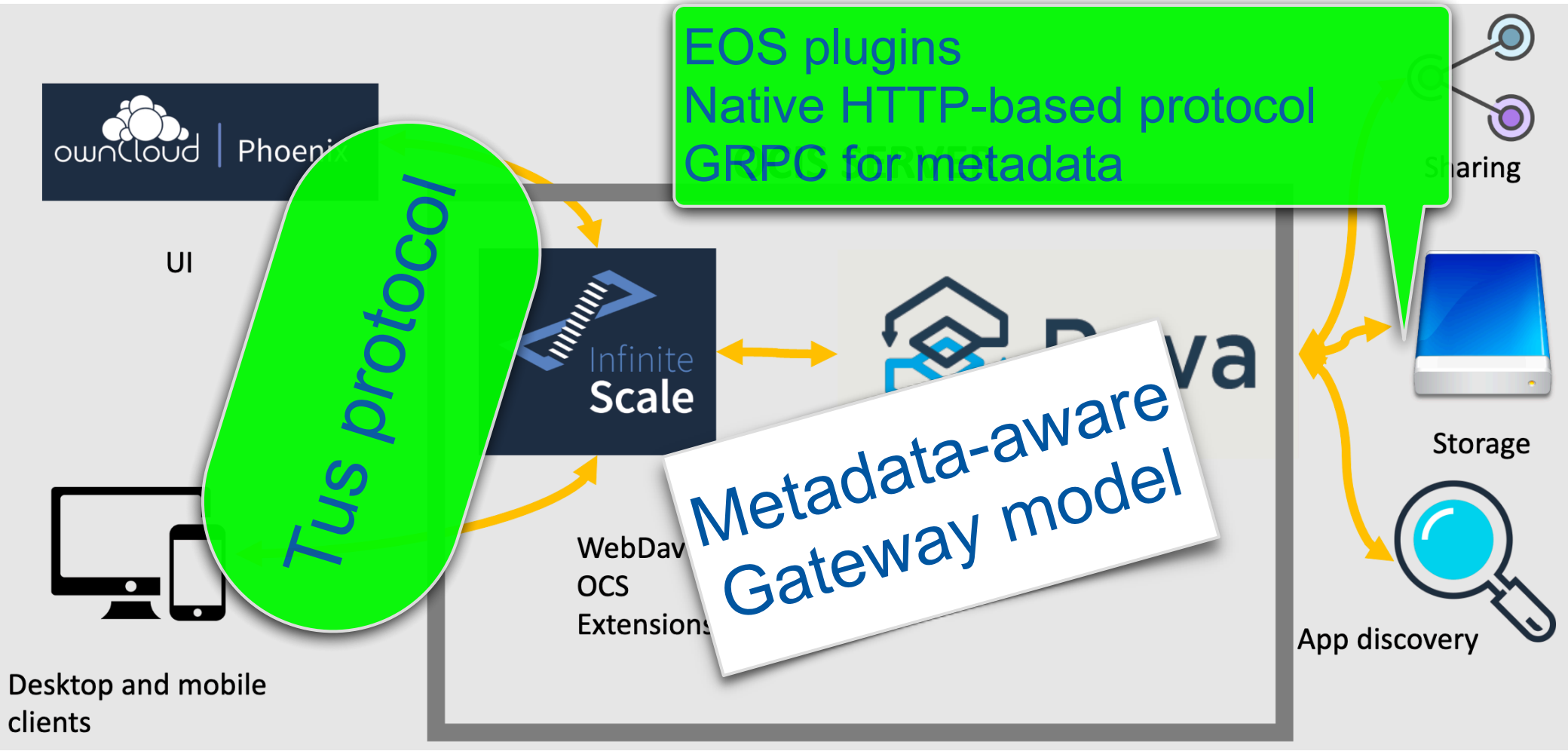
The evolution



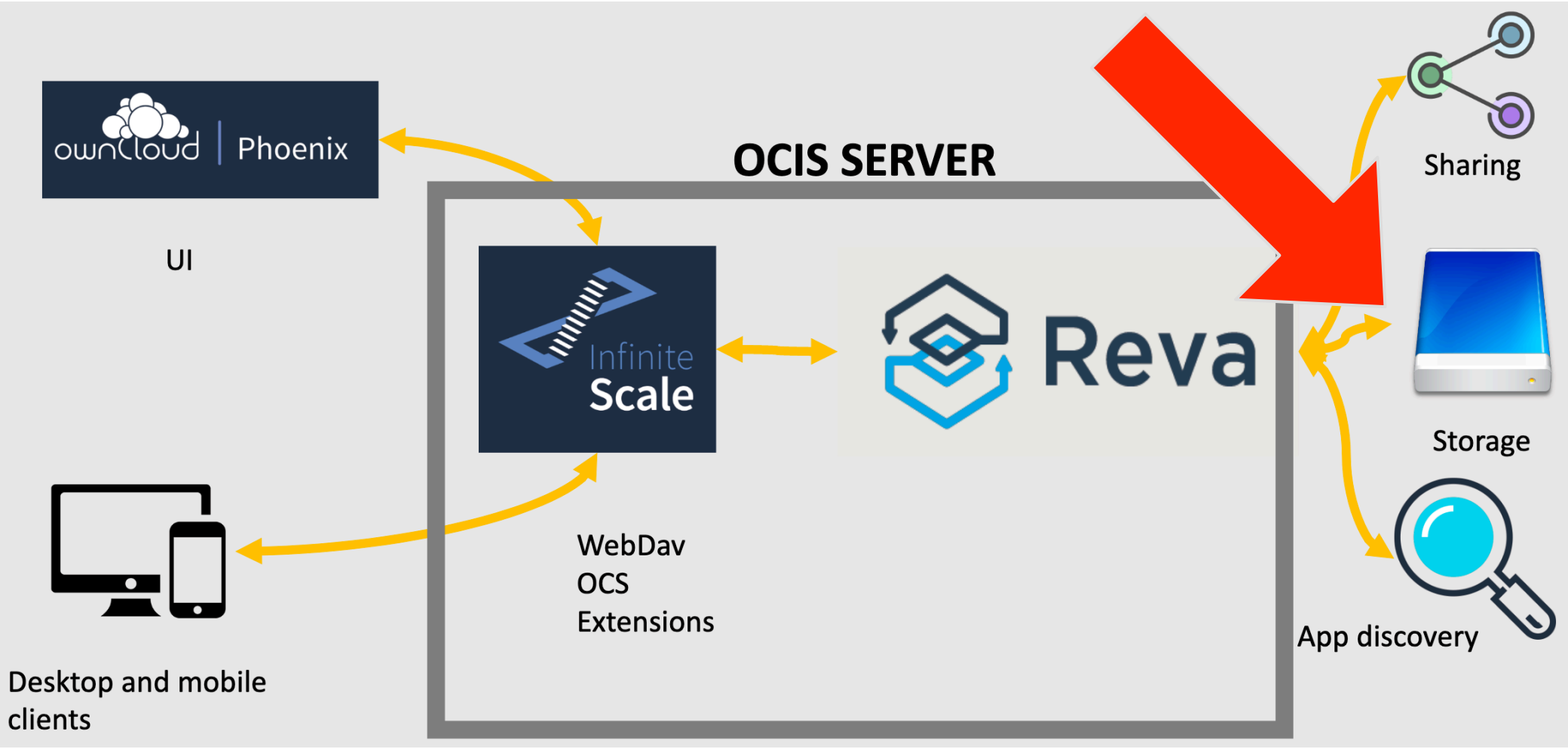
The evolution



The evolution



The evolution



Important place

- The connection between REVA and the EOS storage is being reworked

- Use the GRPC interface of EOS for fast metadata access (done, in testing phase)
- Use the HTTP support of EOS for data r/w
  - And make sure it can support this use case

- The step is more evident if we have a deeper look at how it works now…

Current HTTP gateway behaviour

- Get chunk (partial file) from client
- accumulate in local disk
- Do this for all the chunks, then build the whole file locally

- File complete? Upload it to the EOS servers by spawning xrdcp (xrootd protocol)

- Issues: many large files can fill the tmp space of REVA
- The spawned upload technique is suboptimal
  - consumes many resources, e.g. file descriptors
  - can be slow for small files

GRPC interface to EOS

- Reva can now use the GRPC interface to EOS
- previous model was spawning the EOS command line interface for each metadata request

- We expect more than one order of performance improvements in **metadata** internal transactions (we will evaluate this with the first prototypes)

- Together with the other advantages of GRPC… e.g. load balancing, interoperability, compatibility, etc.

- Status: the interface is almost finished and working. Will require contributing a couple of primitives to the EOS gRPC interface

Two phases

- We decided to proceed in two phases

1. Implement the gRPC+HTTP communication (on the REVA-EOS side)
2. Implement the TUS protocol in REVA and review what's missing for that in point 1

- Point 1 will already be production-ready, as it will use the current client protocol

Phase 1: REVA as native HTTP gateway to EOS

- Get chunk from client as “little” independent file
  - Forward it to EOS using the current OwnCloud upload protocol
  - **On the fly, no local temp copy**
- Repeat until all the chunks have been forwarded

- After the last chunk, EOS rebuilds the full file directly in the EOS backend

Phase 2: REVA as native HTTP gateway to EOS

- At some point Reva will use the TUS protocol towards the clients
  https://tus.io/

- Chunks will come as real chunks, through PATCH requests
  - Forward them to EOS, passing the appropriate offsets in the final destination file
    - Either “PUT with offset” or “bytestream PATCH”
  - **On the fly, no local temp copy**

- Repeat until all the chunks have been forwarded then rename the final file from temp to final

- This will have the goodies of the TUS protocol, keeping its details in REVA

REVA as native HTTP gateway to EOS

- Challenge: requires Reva to have a rock-solid HTTP client
  - The golang HTTP client is a good start

- Challenge: iron out the details in the EOS-side HTTP implementation

- benefits: no more spawning per each remote client
- all thread based, native support, lower replication latency and improved performance for smallish files

- We would like at least Phase 1 to be “production quality” during 2021

Status

- gRPC communication: working! Misses 2-3 quota-related calls (also EOS will be enhanced for that)

- Native HTTP(S) communication… works, and seems pretty snappy
  - Timeout managing… it's there, to be cleaned up
  - HTTP Retry logic across redirections to FST, it's there, to be cleaned up
  - Maybe some more parameters need to be exposed to the config file

- The EOS "vid" configuration works now, still with some question marks. Will it be the final one? Is it missing anything? We'll see…
- As soon as it's a bit consolidated it will be documented, especially for the configuration part

Useful references

Documentation: https://reva.link/
Tus protocol: https://tus.io/
Reva Github: https://github.com/cs3org/reva
GRPC: https://en.wikipedia.org/wiki/GRPC
Q&A: https://gitter.im/cs3org/REVA
Xrootd: https://xrootd.slac.stanford.edu/