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)

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
- 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 services in a simple, portable and scalable way. In order to do that, it leverages the CS3 APIS

Scalable gateway mode

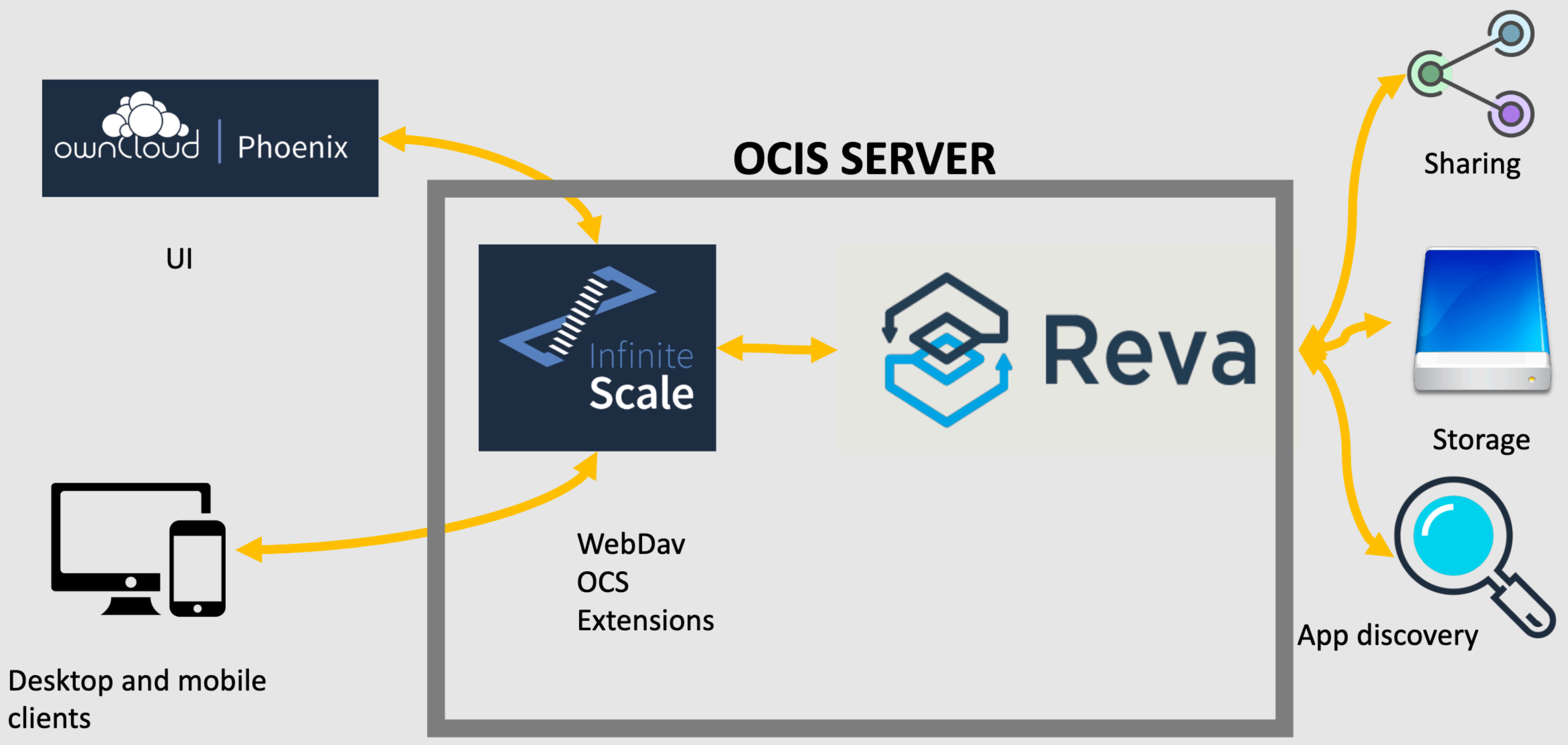
- In practice REVA is the component that adapts the APIs of the sync&share clients to the APIs of the background storage and metadata services

- We can also see this as a sort of highly configurable gateway

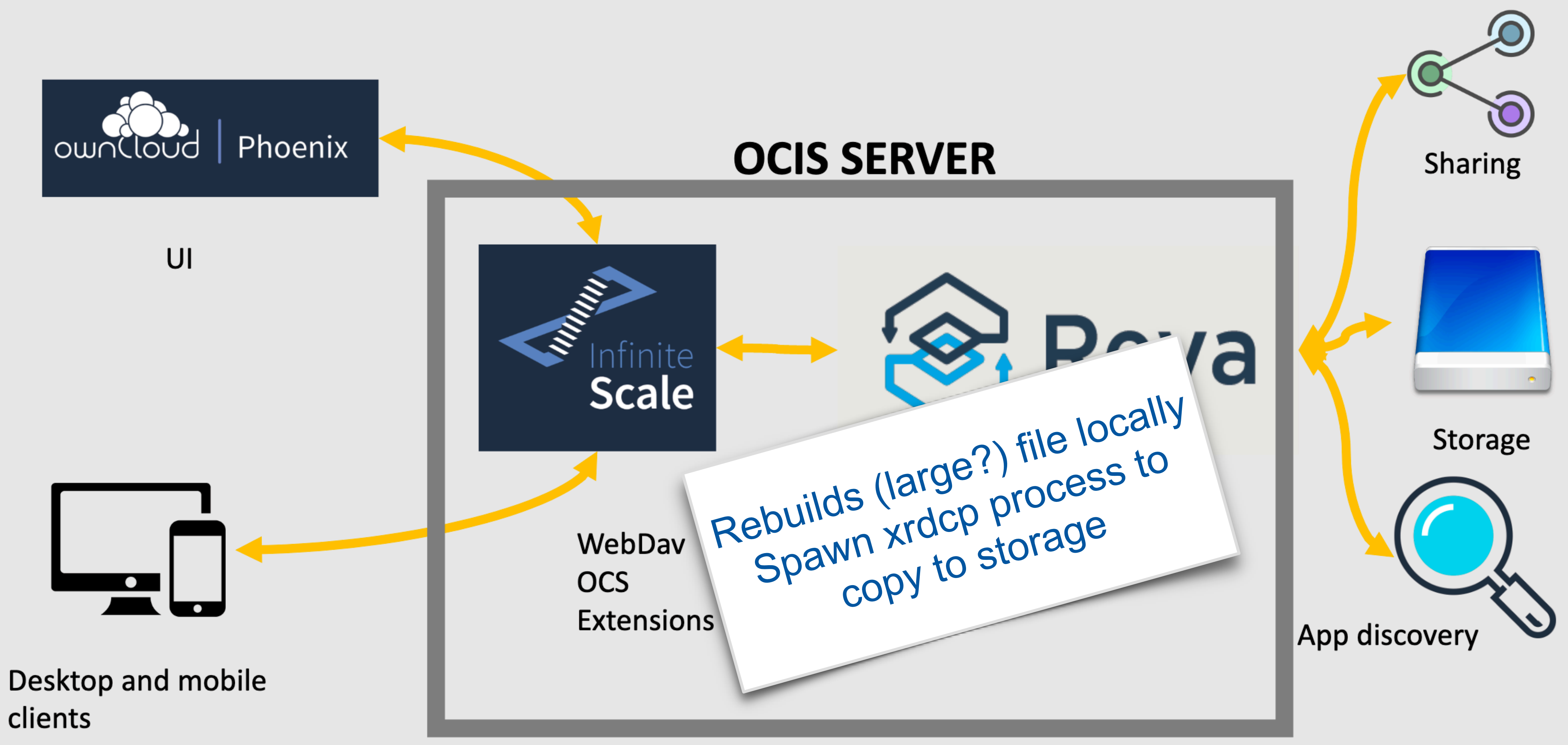
- It becomes more akin to a gateway especially when it starts managing the data (on top of metadata or redirections)

- This is the direction chosen for the evolution of REVA, in particular when EOS is the chosen backend like at CERN

The evolution

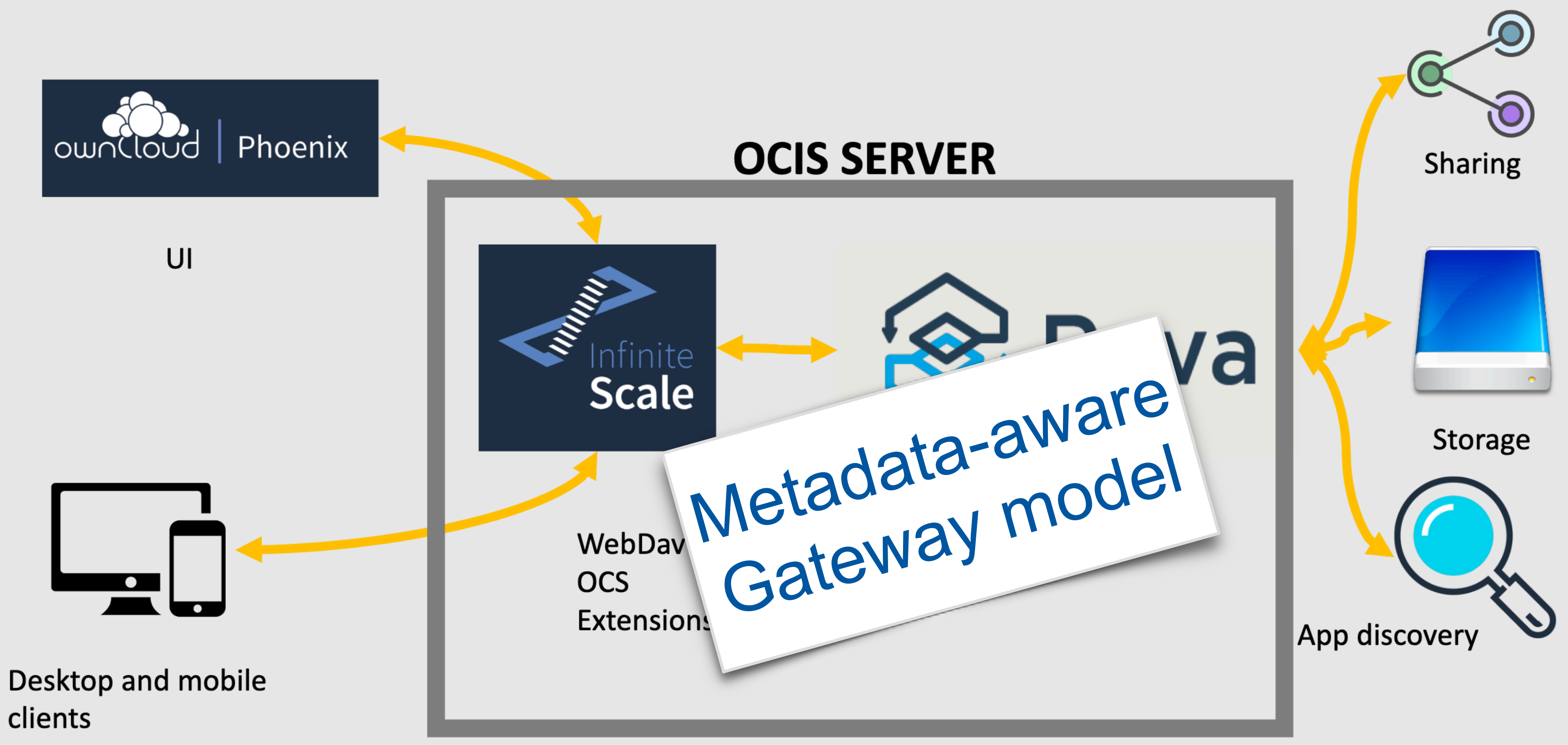


The evolution



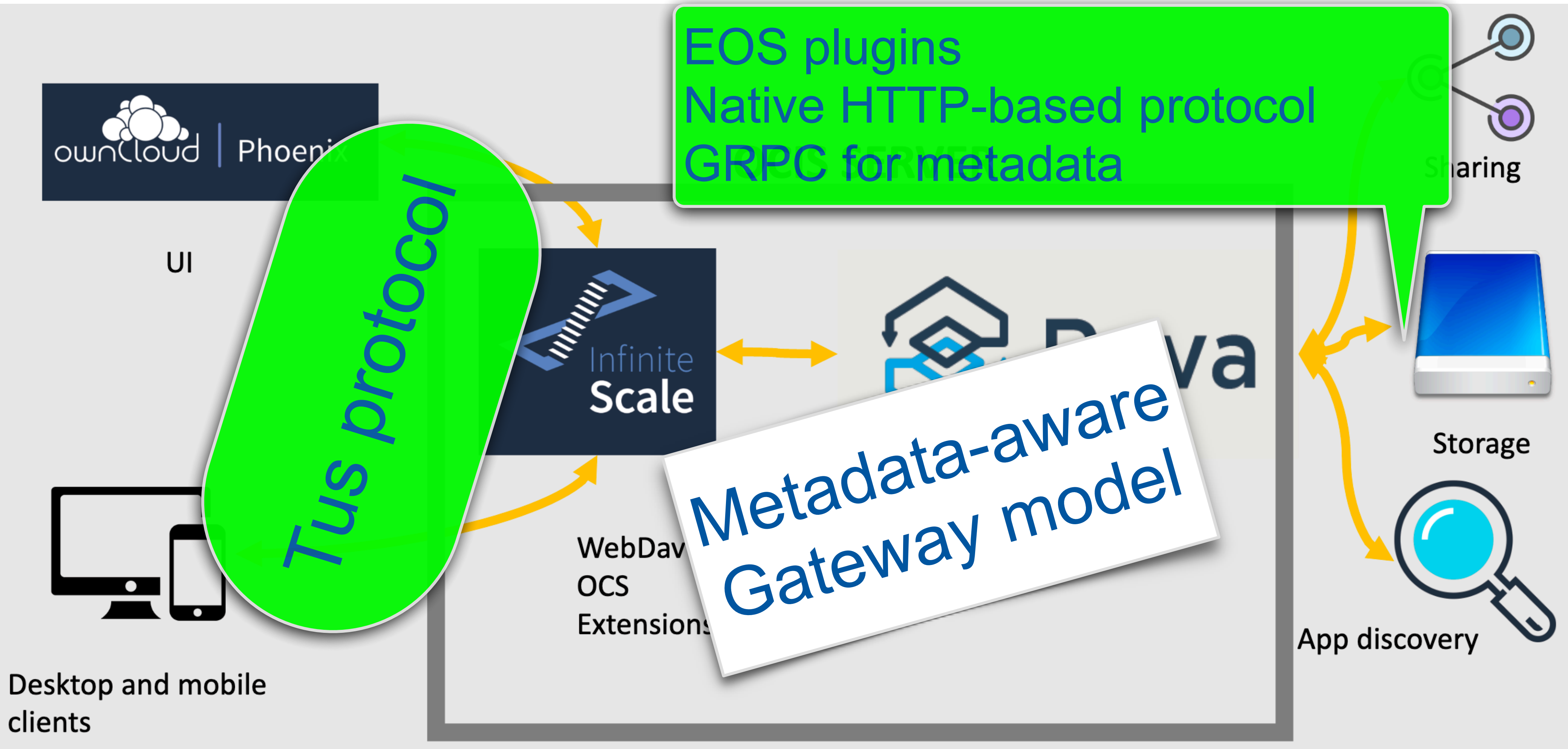
Rebuilds (large?) file locally
Spawn xrdcp process to
copy to storage

The evolution



Metadata-aware Gateway model

The evolution

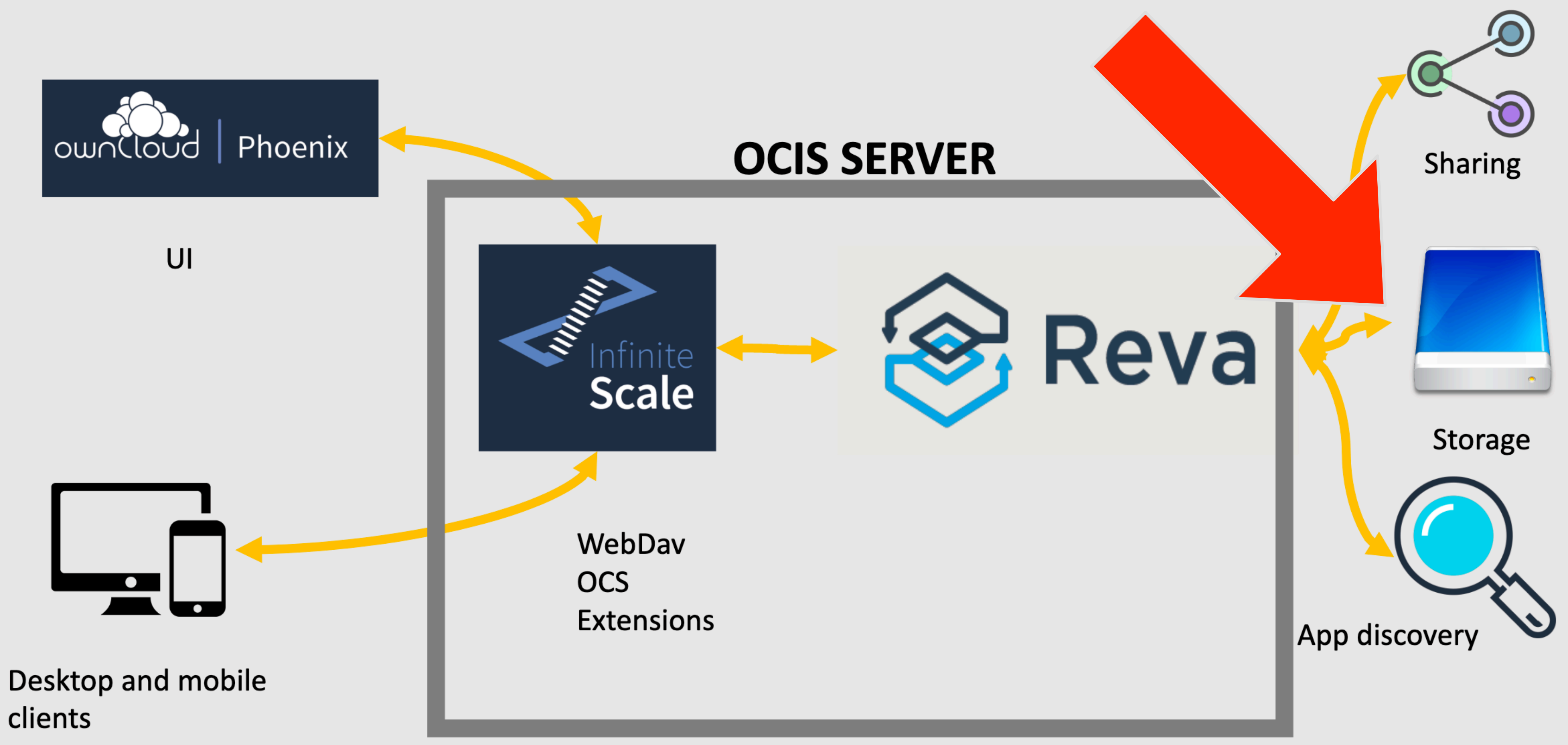


EOS plugins
Native HTTP-based protocol
GRPC for metadata

Tus protocol

Metadata-aware Gateway model

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 improve it so that 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

- 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.

Next: REVA as native HTTP gateway to EOS

- Get chunk from client
- Rebuild temp full file directly in the EOS backend
  - Forward chunks passing the appropriate offsets in the final destintion
    - Either “PUT with offset” or “bytestream PATCH”

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

Next: REVA as native HTTP gateway to EOS

- Challenge: requires Reva to have a rock-solid HTTP client
- 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 this to be “production quality” during 2021

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/