Dynafed - federating, aggregating

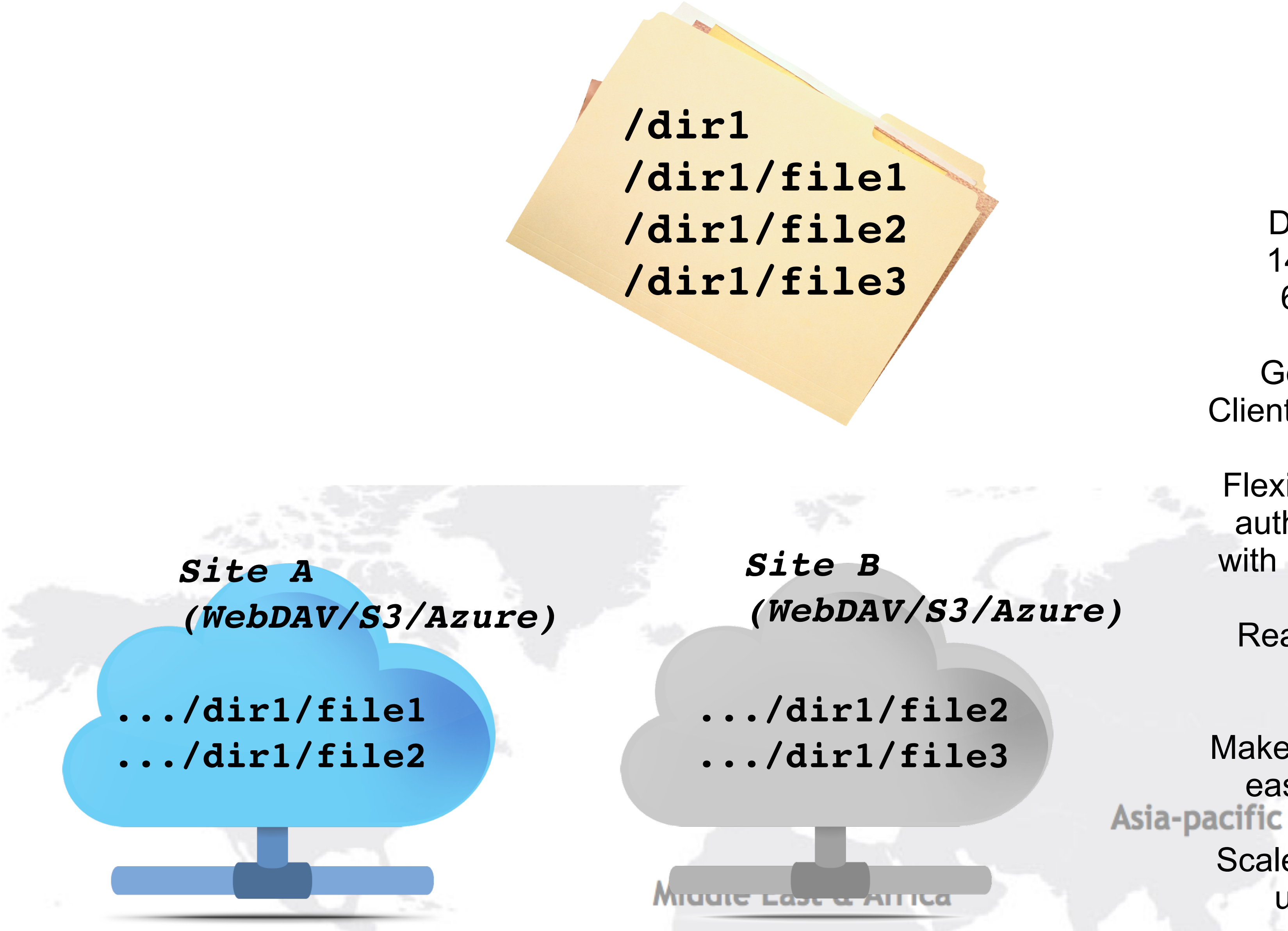
Slightly oriented to cloud storage in grid workflows
June 2019

F.Furano
O.Keeble

What’s Dynafed

- Dynafed is a browser-friendly realtime scalable aggregator of HTTP/ WebDAV/S3/MS-Azure metadata sources
- Aggregates/caches/presents metadata, redirects clients to resources for reading or writing. Geography-aware redirections
- Does not need central persistency (e.g. DBs), does everything on the fly and caches the information
- Technically it can both support or accommodate external DBs

- Realtime detection of sites up-ness, no need of installing anything special at the sites
- Presentation is through WebDAV and HTML
- Very threaded, very asynchronous, works fine in LAN and WAN
- Project started in EMI, 2011
- DAVIX used to be its internal client, then became a successful project per se



Historical
DESY Prototype:
14/15 LHCb sites
60 ATLAS sites

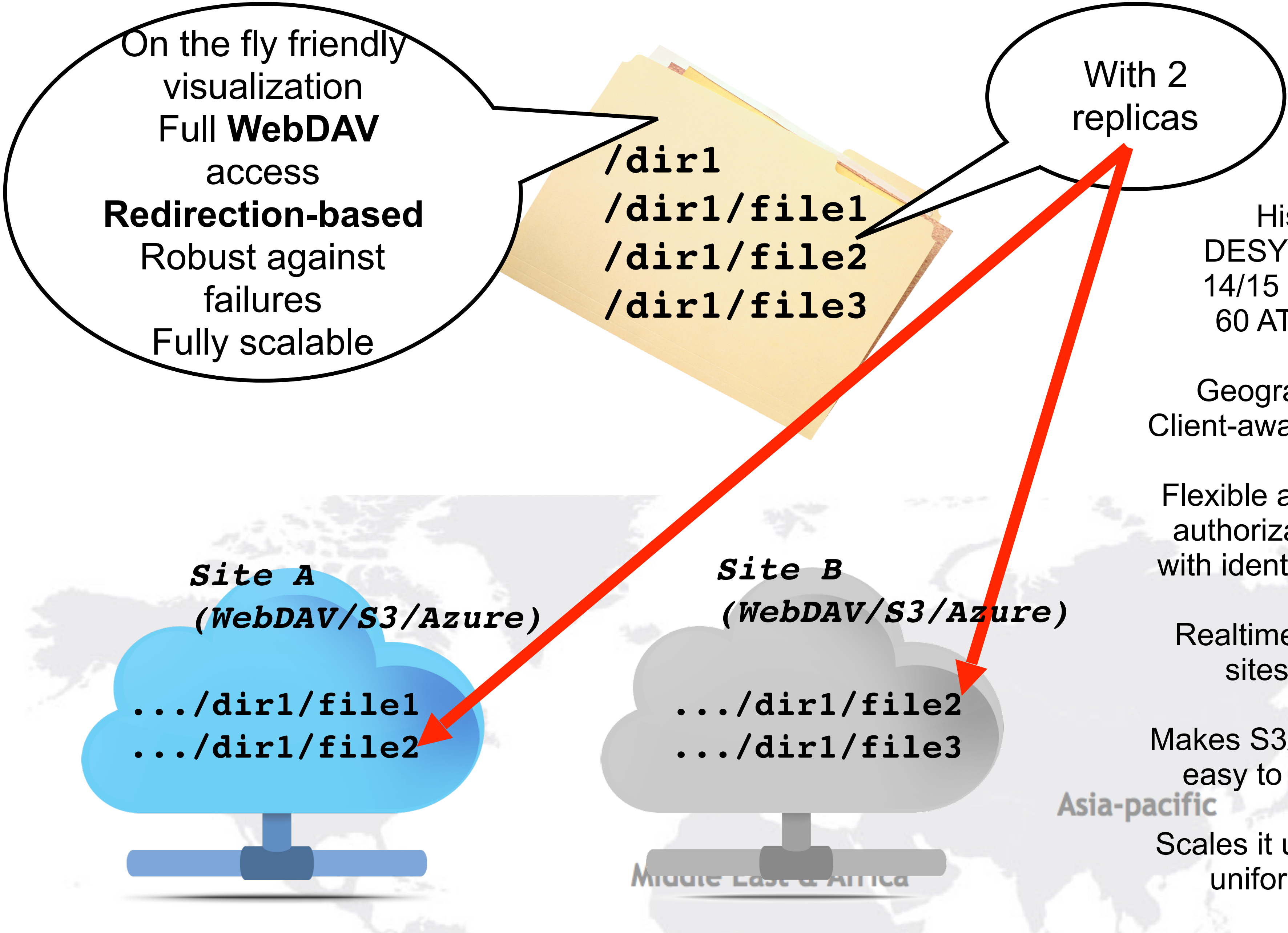
Geography-based
Client-aware redirections

Flexible authentication/
authorization, friendly
with identity federations

Realtime detection of
sites' up-ness

Makes S3/Azure storage
easy to use and mix

Scales it up and applies
uniform security



Historical DESY Prototype: 14/15 LHCb sites 60 ATLAS sites

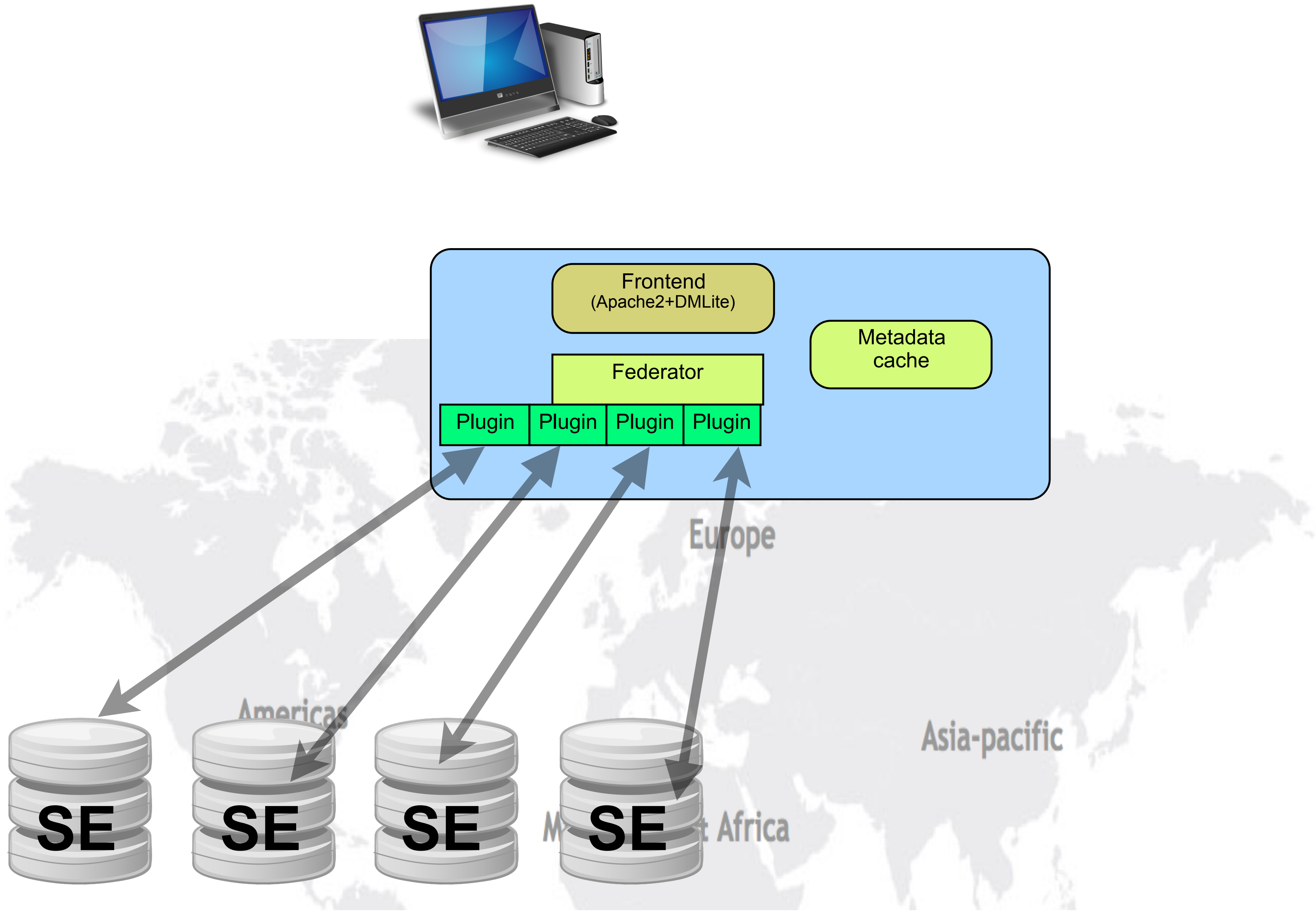
Geography-based Client-aware redirections

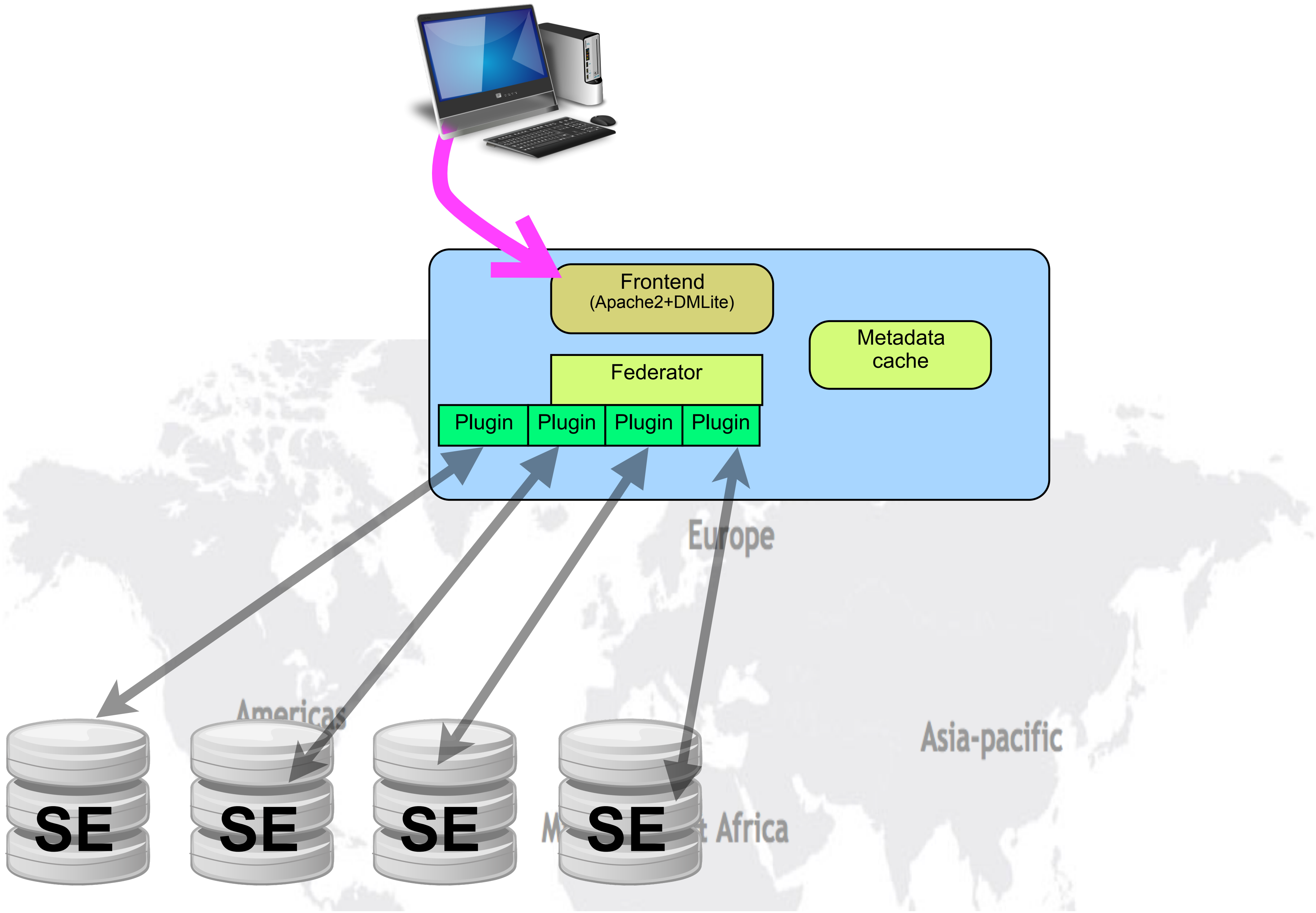
Flexible authentication/ authorization, friendly with identity federations

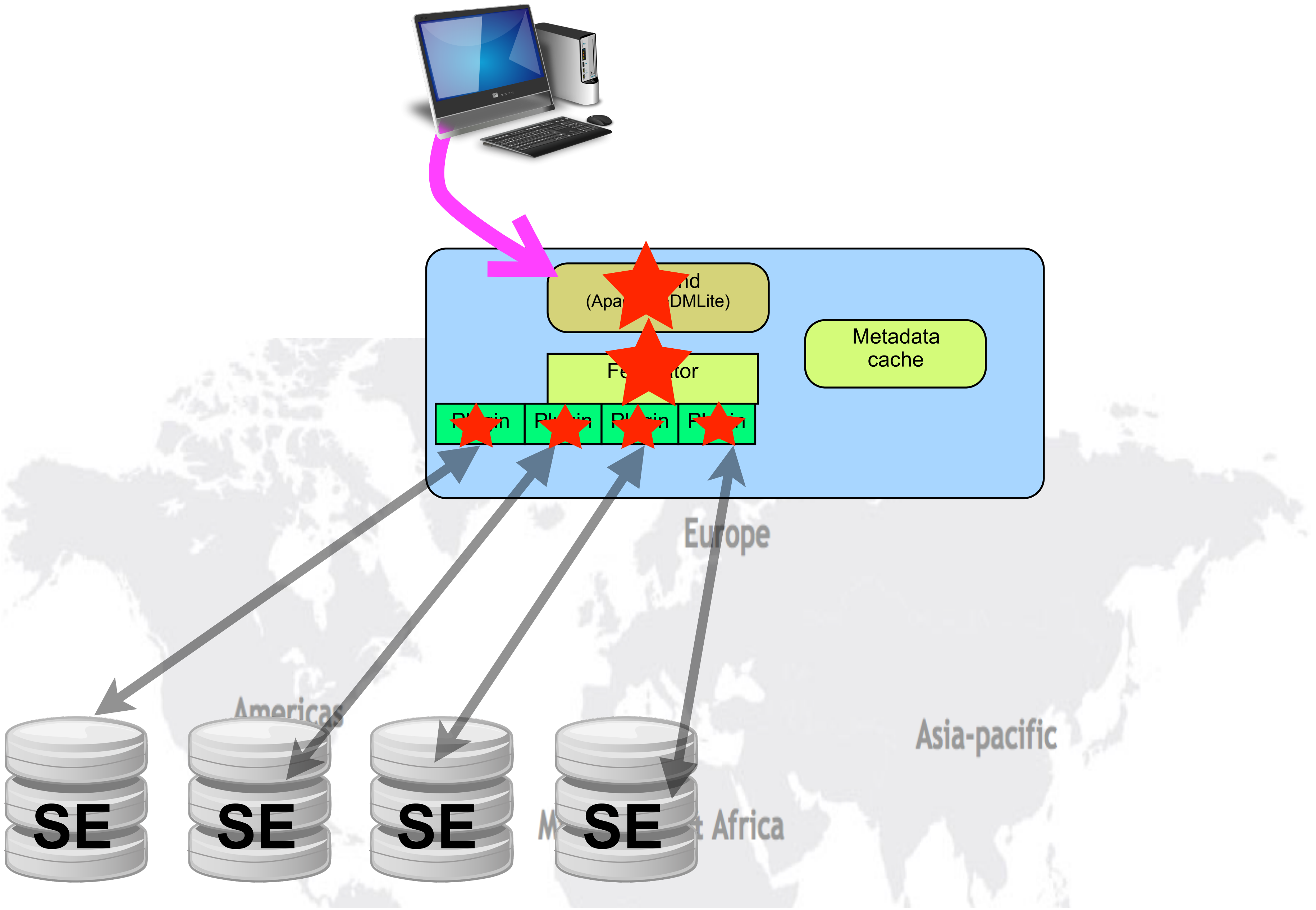
Realtime detection of sites' up-ness

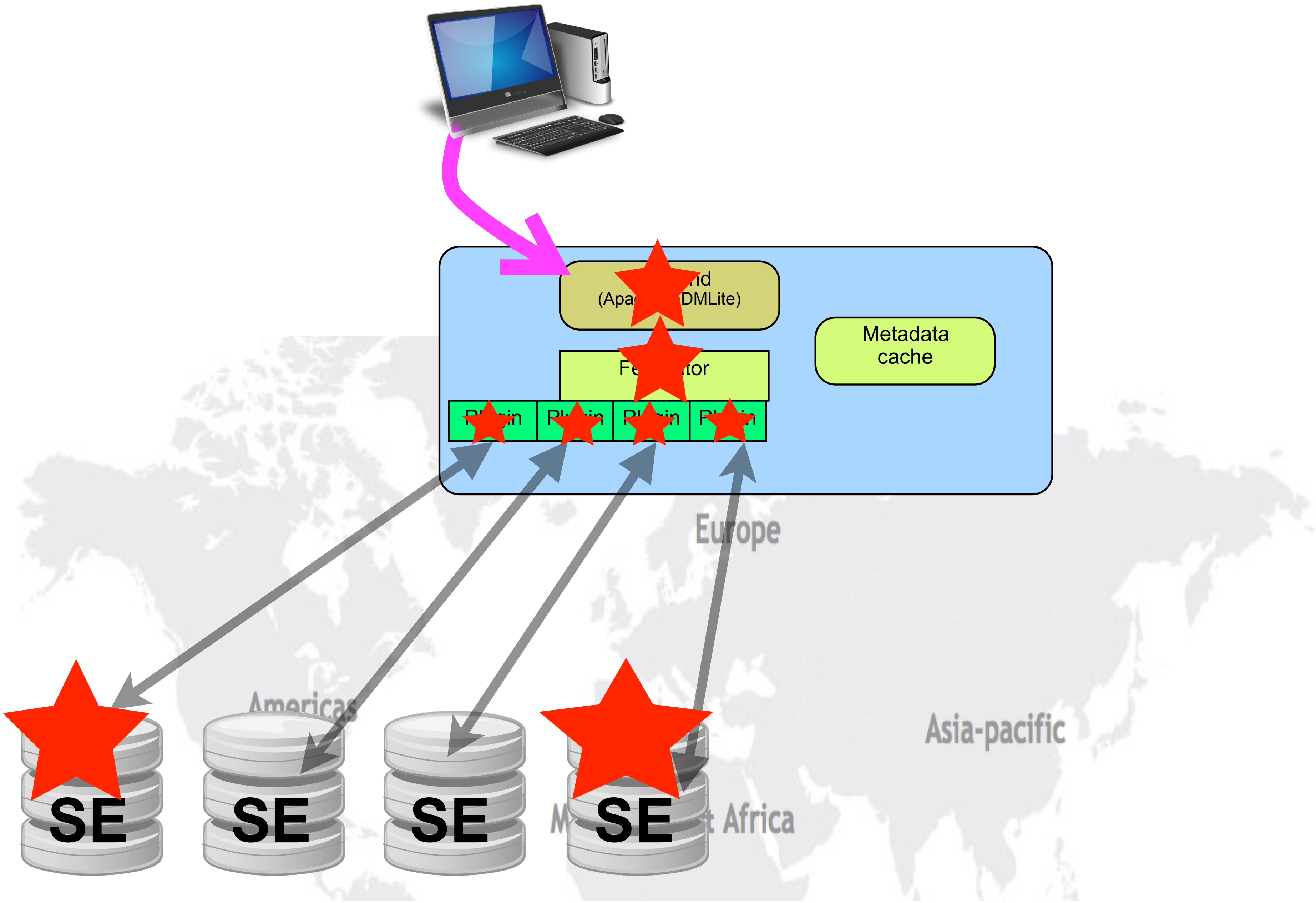
Makes S3/Azure storage easy to use and mix

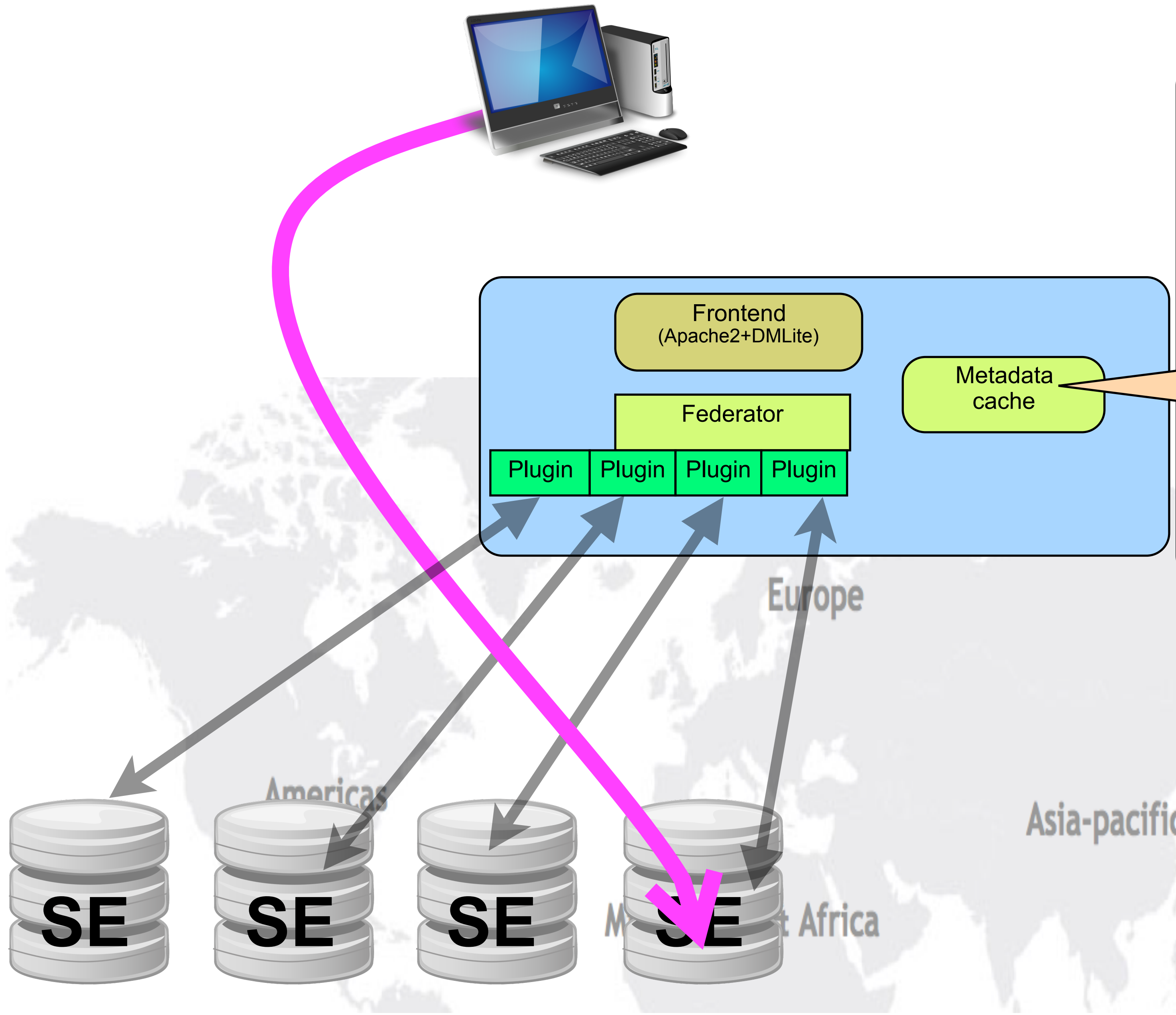
Scales it up and applies uniform security


Frontend
(Apache2+DMLite)
Metadata cache
Federator
Plugin
Plugin
Plugin
Plugin
Europe
Americas
Asia-pacific
Africa
SE
SE
SE
SE


Frontend
(Apache2+DMLite)
Metadata cache
Federator
Plugin
Plugin
Plugin
Plugin
Europe
Americas
Asia-pacific
Africa
SE
SE
SE
SE


nd
(Apa DMLite)
Metadata cache
Fe tor
Pl in
Pl in
Pl in
Pl in
Europe
Americas
Asia-pacific
M Africa
SE
SE
SE
SE


(Apa DMLite)
Metadata cache
Europe
Americas
Asia-pacific
Africa
SE
SE
SE
SE



The cache remembers what happened

The next **metadata** interactions will very likely be fed by the cache

The cache can be shared

Strong points

- Very good metadata performance and scalability
- Agility in setting up and maintaining federations of any size
- So seamless that it's difficult to explain that there's something behind the WWW browser

- At some point someone realized that adding S3 support was opening very interesting possibilities. Then also Azure came in the same way

Dynamic Cloud support

- Dynafed can federate any number of remote S3 buckets or Azure shares (also together with other regular WebDAV)
- **This is not a proxy, data access works with redirections**
- This fed will appear as a unique read/write WebDAV storage, totally seamless, fast and scalable
- Presents in a familiar way the weird S3/Azure implementation of “folders” and hierarchical content

- Object stores are not “flat”, that’s an urban legend. They have a different behaviour

Dynamic Cloud support

- Clients/Users/jobs do not need to know about S3 or Azure mechanics, just use a clean URL and valid credentials in a decent client (i.e. curl with many crazy parms or davix)
- *The keys of the buckets stay secret in the Dynafed frontend*
- *Clients/Users/jobs accessing data do not need to know storage keys*
- **Clients/Users/jobs transparently receive short-term delegations encrypted in the URL signatures**

CERN

Dynamic Cloud support

- Tested with MS Azure, Amazon S3, Ceph S3 implementations
- **A federation of "object stores" can apply uniform, flexible authorization/authentication**
  - Authentication: Can be X509, login/pwd, OpenID-Connect, Macaroons, in principle whatever mechanism that works as an Apache module
  - Authorization: flexible mechanisms to define rules of basically any kind, making choices on the information about the client's request… headers, identity, OIDC information, path…
- The pioneer of these mechanisms has been the BOINC Data Bridge, needing to match username/pwd with X509/VOMS

Dynafed and HEP workflows

- Some HEP workflows can use HTTP resources, e.g.
  - FTS transfers
  - Rucio-based things

- These frameworks have challenging requirements for
  - Checksum support
  - Third-party copy (TPC)

- This has improved recently in Dynafed

Checksum support in Dynafed

- Dynafed does not own or host the files, they reside elsewhere
- When asked for a checksum, Dynafed can:
  - collect checksums from the endpoints that support checksumming (HTTP/Dav only), see options
    - `locplugin.<ID>.candochecksums`
    - `locplugin.<ID>.checksumcalc`

  - OR running a helper hook that is supposed to do something to return the requested checksum

Checksum helper

- Any executable able to process the given parameters and return a checksum on stdout
  - `<site LFN> <checksumtype>`
  - The output must be

```
>>>>> HASH <result>\n
```

Ideas for a checksum helper

- *Dead simple*: use gfal-sum to stream the file through dynafed. Good only for a proof-of-concept
- *Pragmatic*: Use pwdless ssh to spawn the calculation through a simple cluster
- *Glamour*: Use AWS Lambda to calculate the checksum on the cloud where the file resides

- This tool (for S3/Azure) could also cache the checksum into the file's ext attributes
    - Before recalculating a checksum it would check the ext attributes and save a lot of time

COPY verb in Dynafed

- Now dynafed supports the COPY verb, hence it can be requested to push/pull a file elsewhere

- In response to it it can invoke a callout that has the responsibility of triggering the data movement

- NOTE: “triggering” does not necessarily mean “perform”. Performing it can be offloaded… like in the checksum case

"Fourth-party" copy callout

- A callout is the action of executing something (a command, a script…) locally in the machine running Dynafed
- This callout has the responsibility of triggering the data movement described (and already authorized) by the passed parameters

- The simple-minded deployment could be just to invoke the default script, which will use gfal-copy to move the data
  - Hence user->gfal-copy->dynafed->gfal-copy->endpoint
  - The script (actually, gfal2) is able to forward the COPY request to one of the endpoints, making the COPY a "fourth party COPY"
- A better implementation could run the same helper script in a set of companion "datamover" machines, using pwd-less SSH

- These things are simple to describe, they also need some devops cleverness to be put in place

Two callouts: push/pull

- copypull: to copy a file INTO this dynafed’s endpoints
- copypush: to copy a file FROM this dynafed’s endpoints

- The parameters are quite normal things, however the combinations of the details can be challenging to master

The push/pull parameters

cksumcheck: an integer, whose value is nonzero if the copy has to verify the checksums
cksumtype: a string identifying the checksum type that has to be used, e.g. adler32

srcURL: the full URL of the source file
destURL: is the full URL of the destination file or the local logical file name in case of a pull copy

x509proxypath: a local path pointing to the x509 delegated proxy certificate of the user that requested the file copy
auth: additional authorization information. In the case of Apache HTTPD this field comes from the content of the Authorization header of the original COPY request. Useful for macaroons or OIDC

additional optional parameters: these are filled by copying the value of selected HTTP headers. To activate this mechanism, the Ugr configuration must provide one or more directives *glb.filepull.header2params[]* or *glb.filepush.header2params[]* respectively for file push or pull requests.

The 5GB tricky detail

- **Files larger than 5GB can be PULLED into cloud storage backends only if the PULL script is given as destination the dynafed URL, not the endpoint**

- This is because uploading big files into cloud storage must follow the proprietary S3 or Azure workflow
- This workflow is supported by Dynafed, the script ultimately must PUT to dynafed, not elsewhere :-)
- In this case Dynafed and Davix shield the application from the technicalities of the S3 or Azure transactions.

- These things are well described in the Dynafed whitepaper and in the abundant comments of the example scripts

OIDC/OAuth2 support

- Dynafed has added support for two OAuth2 modes of client authentication
  - OAuth 2.0 Resource Server (RS)
    - Validates bearer access tokens sent by OAuth 2.0 clients.
    - CLI–oriented solution
    - No browser required
    - Client provides the bearer token in the Authorization header
  - OpenID Connect Relying Party (RP)
    - Authenticates users against an OpenID Connect Provider
      - Requires a browser
      - User is redirected for interactive login
    - Dynafed receives user identity information from the IdP in an ID Token

OIDC/OAuth2 support and configuration

- Dynafed can now support 3 authentication systems - X509, OIDC and OAuth2
  - These are all implemented by the relevant apache modules
  - They are configured at the Dynafed level by establishing a different namespace prefix for each auth system
    - Add the following to /etc/ugr/ugr.conf
      - glb.n2n_pfx: /oauth2 /x509 /oidc
  - Configured in apache by creating a separate prefix (<Location /...>) for each system
    - Each then configured for a particular auth
      - AuthType oauth20
    - Relevant attributes are then available for authorisation decisions
  - In this way, the different authentication front-ends share the same Dynafed cache.

Free-space aware writing

- Dynafed can now redirect writes only to endpoints which advertise enough (configurable) free space
- Based on “dynafed_storagestats” from Fernando Fernandez Galindo
  - https://github.com/hep-gc/dynafed_storagestats
  - Thank you for this contribution!
- dynafed_storagestats is run periodically (e.g. cron) and populates Dynafed's memcached cache
- Dynafed can now consult this information when redirecting a write
  - Enabled and configured via /etc/ugr/ugr.conf
  - glb.minfreespace: 10

Dynafed

- An advanced system that now is quite consolidated
- Extremely flexible and scalable, can be used in many ways
- This does not intend to be an instruction manual
- Many items have been left out
  - E.g. the macaroons support. Working fine, usable only for pure cloud storage backends
- If you need info on a specific subject, just ask

http://lcgdm.web.cern.ch/dynafed-dynamic-federation-project

https://gitlab.cern.ch/lcgdm/dynafed/raw/develop/doc/whitepaper/Doc_DynaFeds.pdf?inline=false

Next phases

- Support for production deployments
  - Checksums
  - 4th party copy

- Waiting for OIDC to come
  - with the WLCG profile
  - Unlikely that it needs developments for that, likely support

- Integration, consolidation, support

- EPEL 8