1. Motivation

Scientific communities require global access to data which is effective, simple and convenient. Logical integration of independently managed resources used by particular research groups is needed.

2. Existing Solutions

None of the existing services and tools possesses all needed features to provide global, easy and efficient access to data in organizationally distributed environment.

3. From Federation to Globalization

Shortcomings of globalization vs. federation:
- No trust between providers (see 4).
- Lack of accounts integration (see 4, 5).
- No data sharing with users not known to provider (see 5).
- No common authentication/authorization mechanisms (see 4).
- Higher network delay and lower bandwidth (see 6).

4. Global Cooperation

Solution: trusted mediator.

Global Registry stores information about entities in the system, e.g., who the user is, which spaces he/she is a member of, and where to find other providers supporting his/her data. Global Registry also serves as a certificate authority.

5. System Entities Organization

Spaces are logical data containers for storing user files. Providers contribute to onedata with storage resources.

A space can be supported by many providers. Transparent access and coherent view on data. Shared spaces by many users.

6. Working with Many Providers

Automatic propagation of metadata. Propagation limited to providers that support the space connected with particular change.

On-demand many to many nodes connections. Data requests split and prioritized. Prefetching by connections with lower priority. Multiplication of intensively used file parts.

Implementation and Conclusions

Cooperation between providers is necessary in order to achieve global data access with minimal performance loss. As a number of simultaneously transferred between providers files and metadata may be very high so server applications that manage data have to be massively parallel. For this reason scalable management server application was implemented in Erlang and FUSE client, that minimizes server load operating on storage systems directly, was implemented in C++.

As a result of combining Global Registry, spaces, multiple cooperating providers and space members, we obtain a safe, commonly available, shareable, efficient and easy to use middleware for using distributed storage systems.

Acknowledgments

This research is supported partly by the European Regional Development Fund program no. POIG.02.03.00-12-137/13 as a part of the PLGrid Core and the AGH-UST grant no. 11.11.230.124. MW is grateful for AGH-UST grant no. 15.11.230.153.