Information School
UNIVERSITY of WASHINGTON

A Distributed Architecture for Intra- and Inter- Cloud Data Management

Presented by: Ian Kelley

Information School
University of Washington
E-mail: ikelley@uw.edu

ScienceCloud 2014 – June 23rd, 2014

Agenda

- Introduction
- Motivation
- (my) Data Landscape
- Challenges
- Attic Background
- Attic Architecture
- Use-Cases
- Applicability to Science Clouds
- Conclusion

Motivation

- Access and Share Heterogeneous Datasets
- Migration paths between different environments
- Share data within and between Clouds and other storage (e.g., local) compute resources
  - Both transient and long-term data
- Promote data reuse and research reproducibility

Motivation - Example

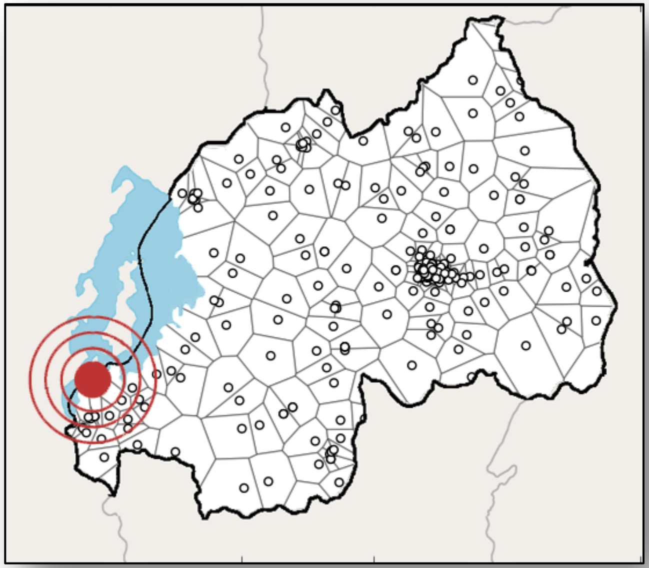
- Within iSchool DataLab, we have projects using different heterogeneous data sources:
  - Call Detail Records (CDRs) [terabytes]
  - Twitter data (archived from years of collecting)
  - Wikipedia data dumps
  - Citation databases & full article texts
- Analysis uses different datasets
  - All stored in different places with different technologies
  - Not easy to discovery or access
  - Time consuming and obtuse management

Motivation - Example

- Common problems for several DataLab projects
  - Large/sparse data with spatial and temporal attributes
    - (e.g., terabytes of time series Call Detail Records (CDRs)
- Need for “easy-to-use” tools and middleware
  - manage, compute, and analyze the data
- Different groups must be able to contribute to & use different portions of data and code
  - Varying expertise levels and project involvement
  - e.g., Ph.D. students; independent study classes; researchers

Example - CDRs

- Extract socio-demographic information from Call Detail Records (CDRs)
  - Metadata passively collected by telecommunications providers (i.e., who called who, when, and where)
- CDR data is very rich and informative
  - Locations of both parties
  - Social/business contacts
  - Mobile payments & salaries
  - Network usage patterns
  - Device information

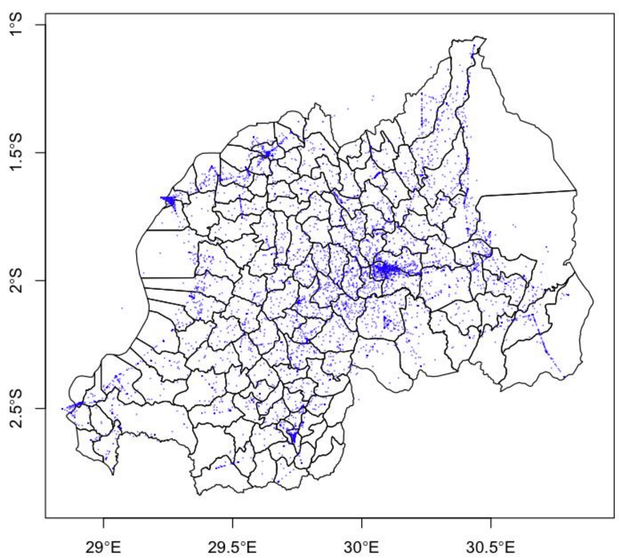


Example - CDRs

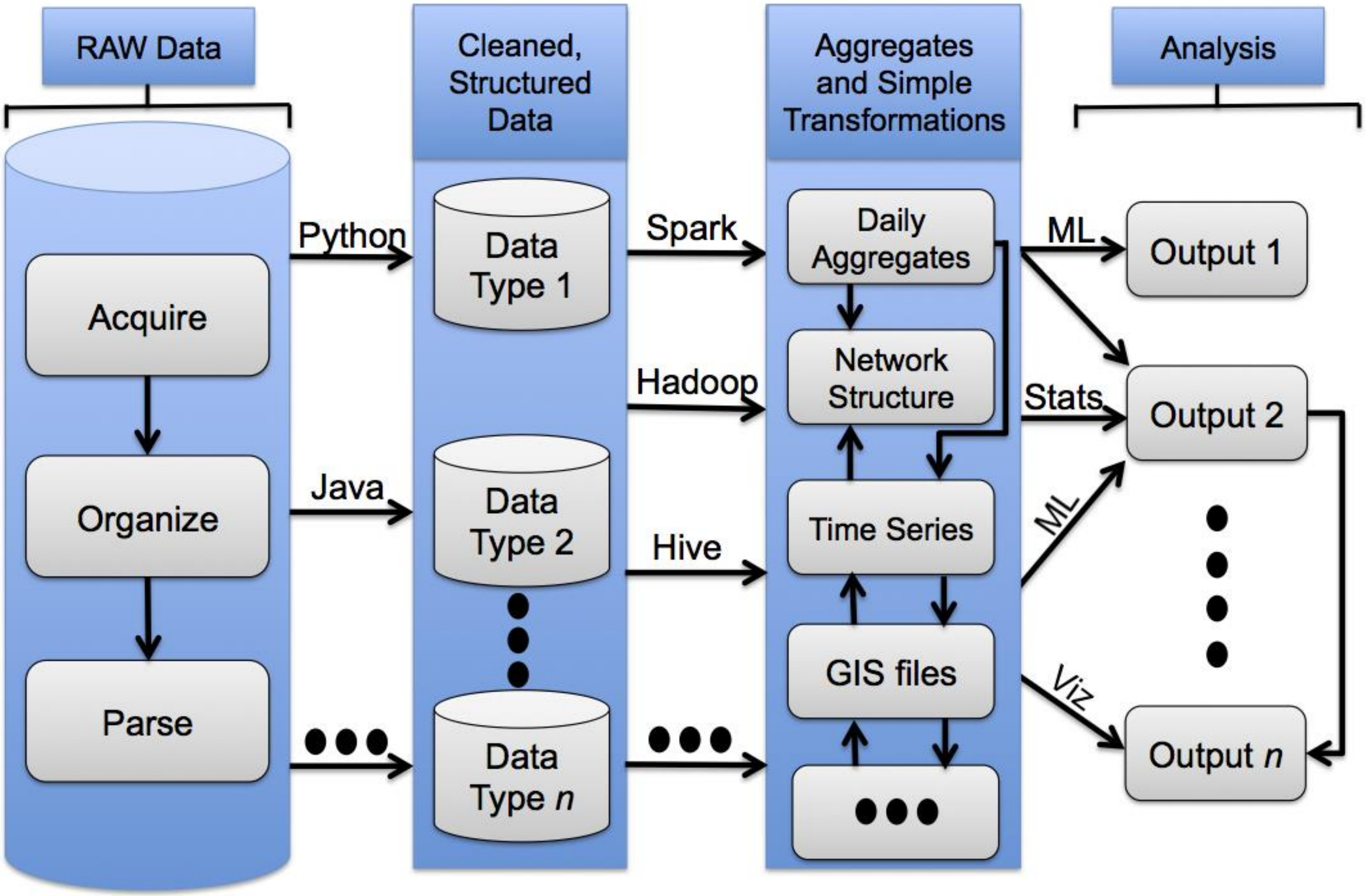
- CDR data can be used for analysis such as:
  - Map migration patterns of workers during labor market shortages
  - Discover the effects of different geopolitical and economic events on internal population mobility
- Analysis relies on combining one or more datasets
  - E.g., local labor market data; census data; spatial maps
- Additional data can lead to much deeper analysis
  - E.g., include social media data, public records, weather, etc
  - But: can be hard to find, manage, version, keep updated

Example - CDRs

- Even simple metrics like Center of Gravity require several datasets
  - (Average position during a time period (e.g., day, week))
  - Subset of mobile phone logs; tower GPS locations; up-to-date spatial files
- Richer analysis with more data
  - Labor markets, weather patterns, public holidays, geopolitical information
  - Social media (e.g., Twitter sentiment analysis)



CDR - Data Workflow



Vision

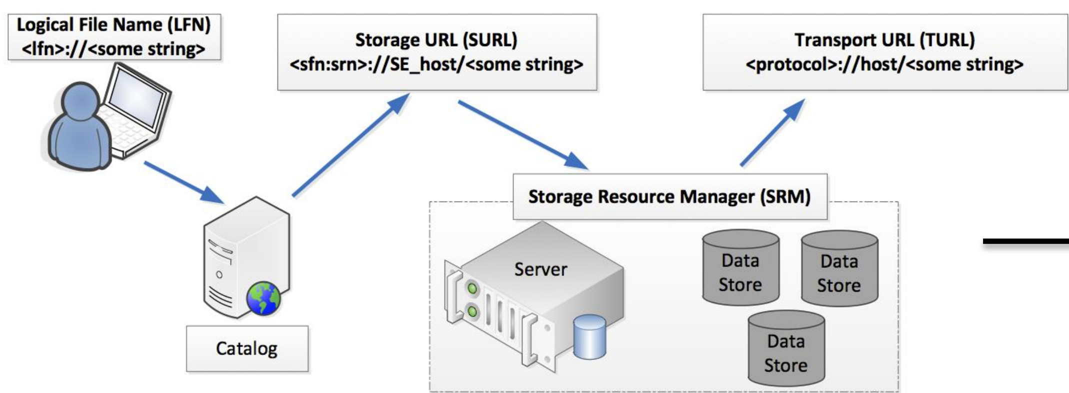
- Manage and share datasets
  - world; colleagues; self
  - short-term; long-term
- Bundle related data & discover it
- Provide tools for smaller (& non-CS) groups
  - Cannot assume large, robust, long-term, hard to administer system fits all needs
- Actively share data without "DOS attacking" hosters
  - Pull few copies, distribute remainder on "own resources"
  - Can remove additional copies after analysis is complete

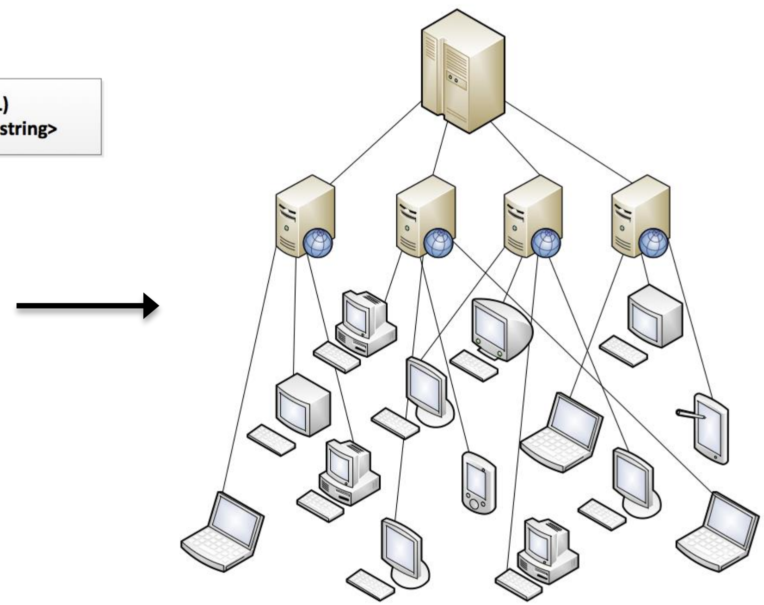
Challenges

- Data can be stored in a number of ways
  - Secured in online repositories (e.g., HPC data)
  - Hosted openly in its entirety (e.g., Wikipedia dumps)
  - Accessible by streaming (e.g., Twitter “Firehose”)
  - Offline (or otherwise inaccessible)
- Discovery can be difficult
- Access can be difficult (…or just “different”)

Data Cloud - Motivation

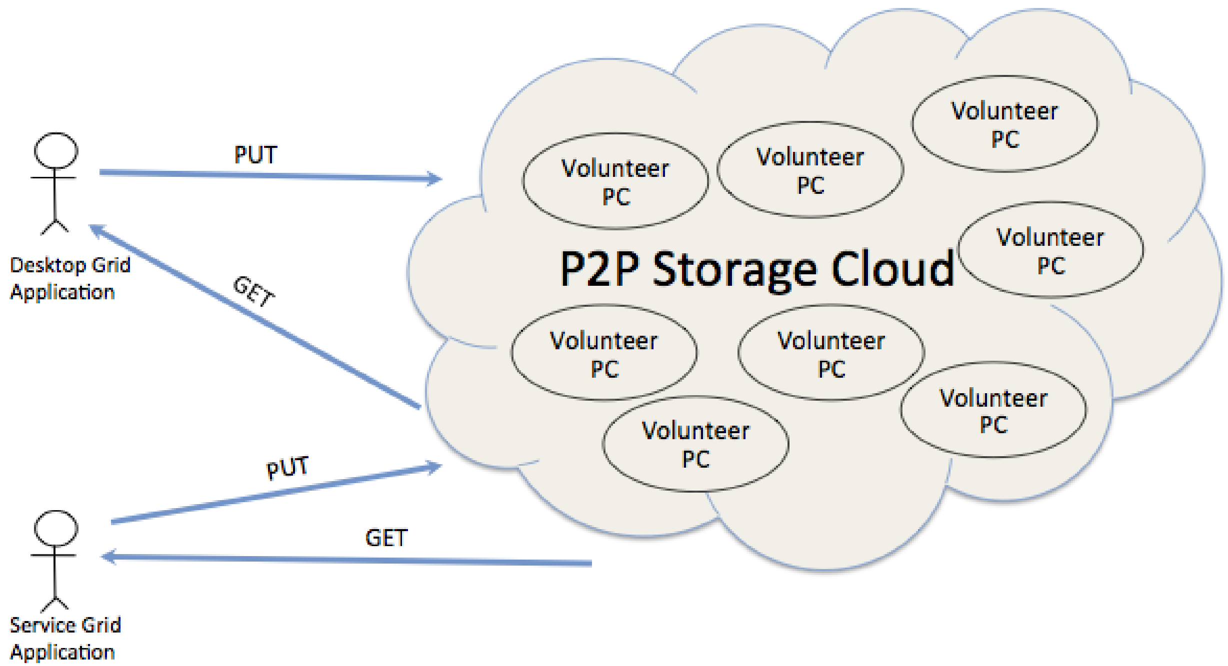
- EDGeS and EDGI Projects' goals
  - transition jobs from Service Grids to Desktop Grids
- Distribute Data *within* Desktop Grids (DGs)
- Transition Data from HPC resources to DGs





P2P-style Solution?

- Push data to a dynamic (P2P-style) network
- Network exposes data for broader consumption



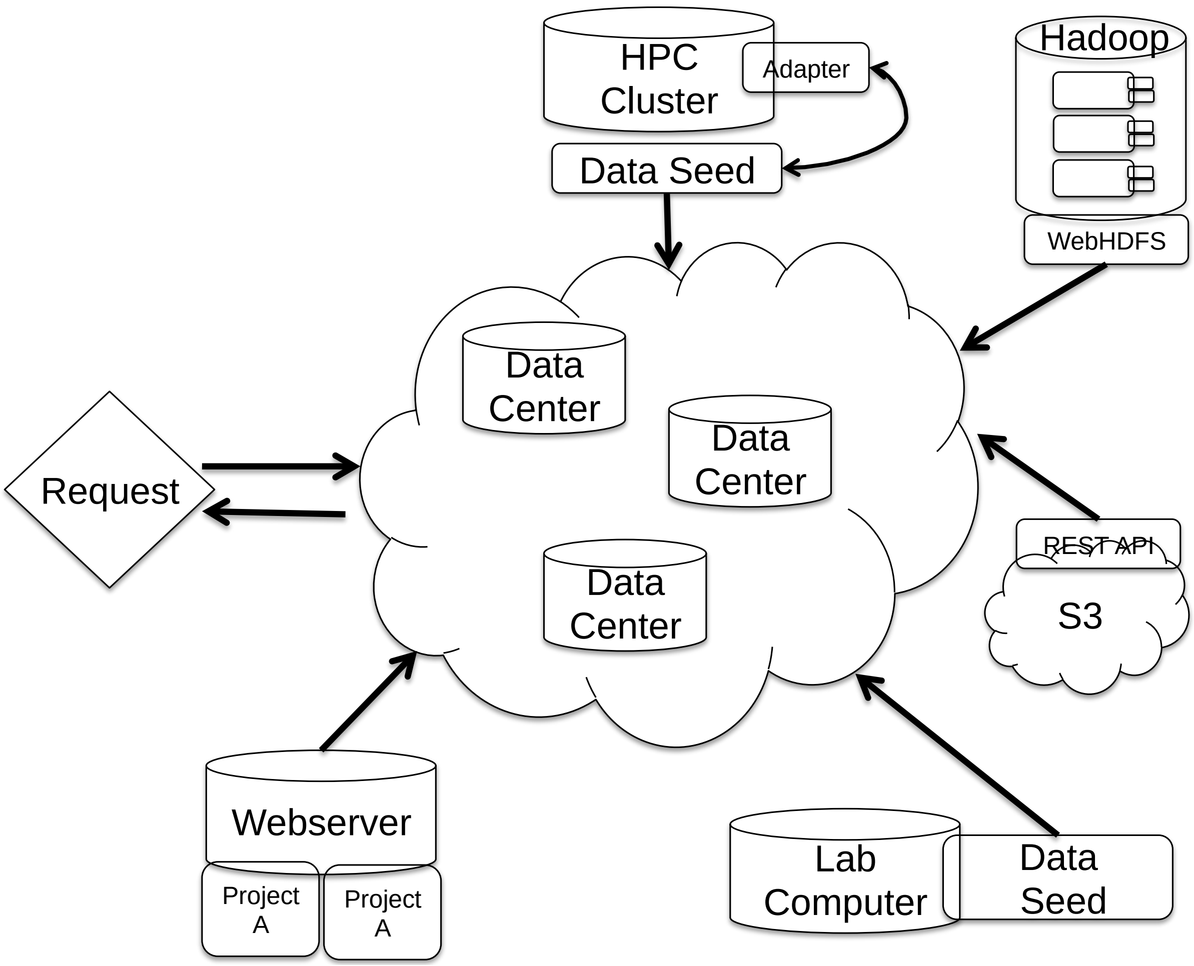
How this relates to Science Clouds

- Access data in heterogeneous resources
- Distribute and use data on-demand
    - E.g., in systems such as AWS, Azure, local clusters, …
- Share data/subsets with different access levels
- (Transient) data access and usage patterns
- Scale up/down to meet demand
- Utilize latent network/storage capacity in Clouds

The Idea

- Don't "reinvent the wheel" or "one size fits all"
- For some data: leave it where it is, and expose it
- For other data: cache and replicate
- In both cases:
  - Act as a dynamic data layer between resources
  - Provide unified access pattern of data (e.g., through URIs)
  - Share data/subsets with different access levels
- When replicating:
  - Scale to provide on-demand needs and utilize local disks
  - Utilize latent network/storage capacity in new environment

(my) Data Cloud Vision



Attic

- Overview of Attic P2P architecture
  - History
  - Overview
  - Message types
  - Protocols
  - Security
  - Features
- Scenario/Use-case outline
  - making data available to DG from Attic network

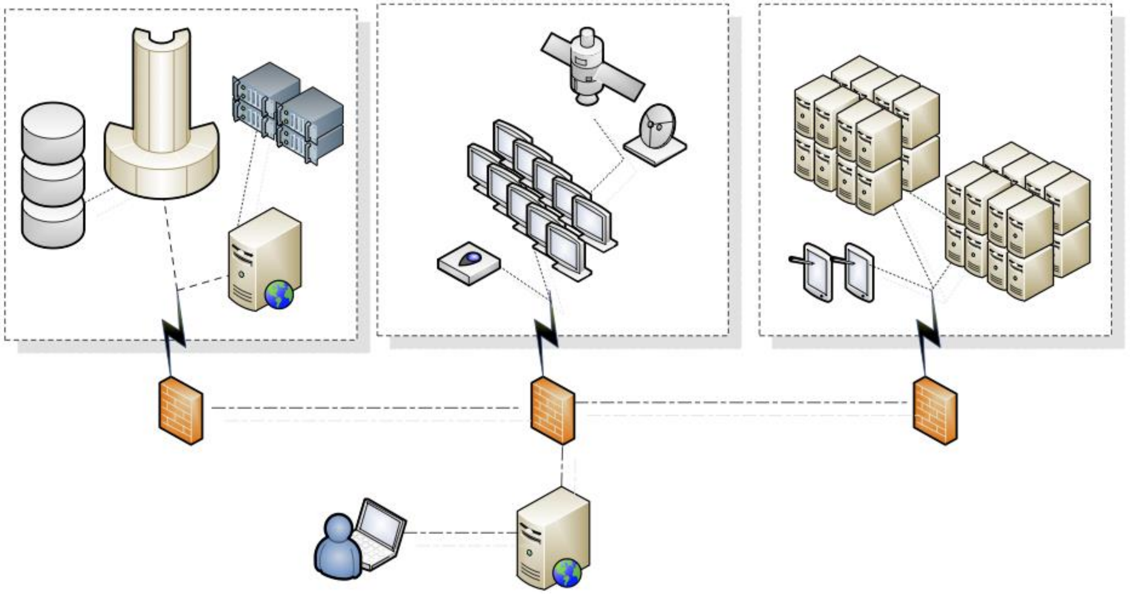
History

- Started as part of a UK EPSRC proposal in 2005
  - Focus on providing data distribution inside Desktop Grids, with target community being Einstein@home
- Continued development under EU FP7 EDGeS (2008-2010) and EDGI (2011-2012) projects
  - Need to provide a way to support data distribution within Desktop Grids for load balancing
  - Additional focus on moving Service Grid data and jobs to Desktop Grids, and legacy application support
- "Data Management in Dynamic Distributed Computing Environments" (Ph.D., 2013)
  - http://orca.cf.ac.uk/44477/

Project Website: *http://www.atticfs.org*

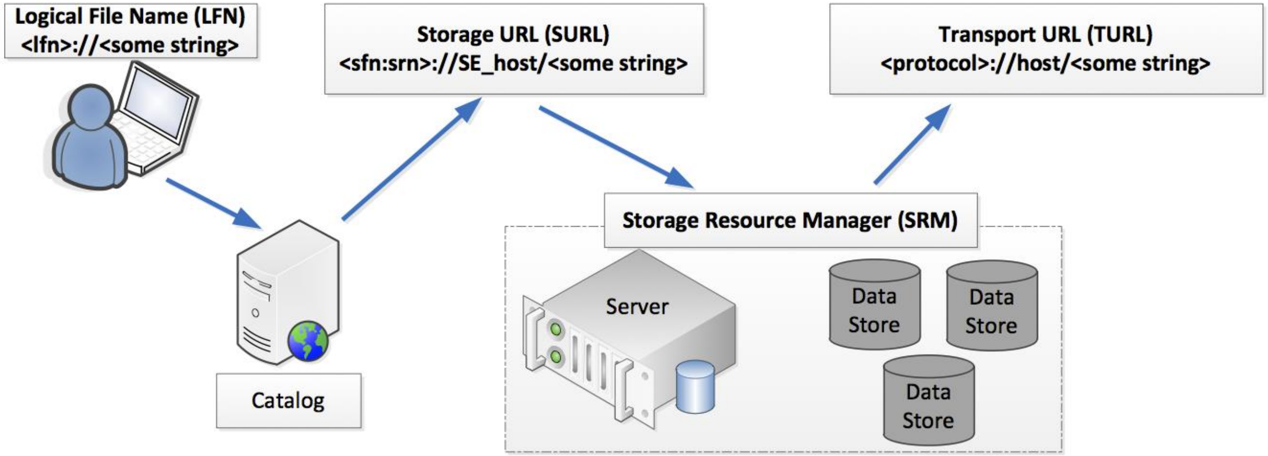
Attic: 10,000 foot view

- Distribute Data *within* Desktop Grids (DGs)
- Transition Data from HPC resources to DGs
    - E.g., ARC-> DG, EGEE -> DG, Unicore -> DG

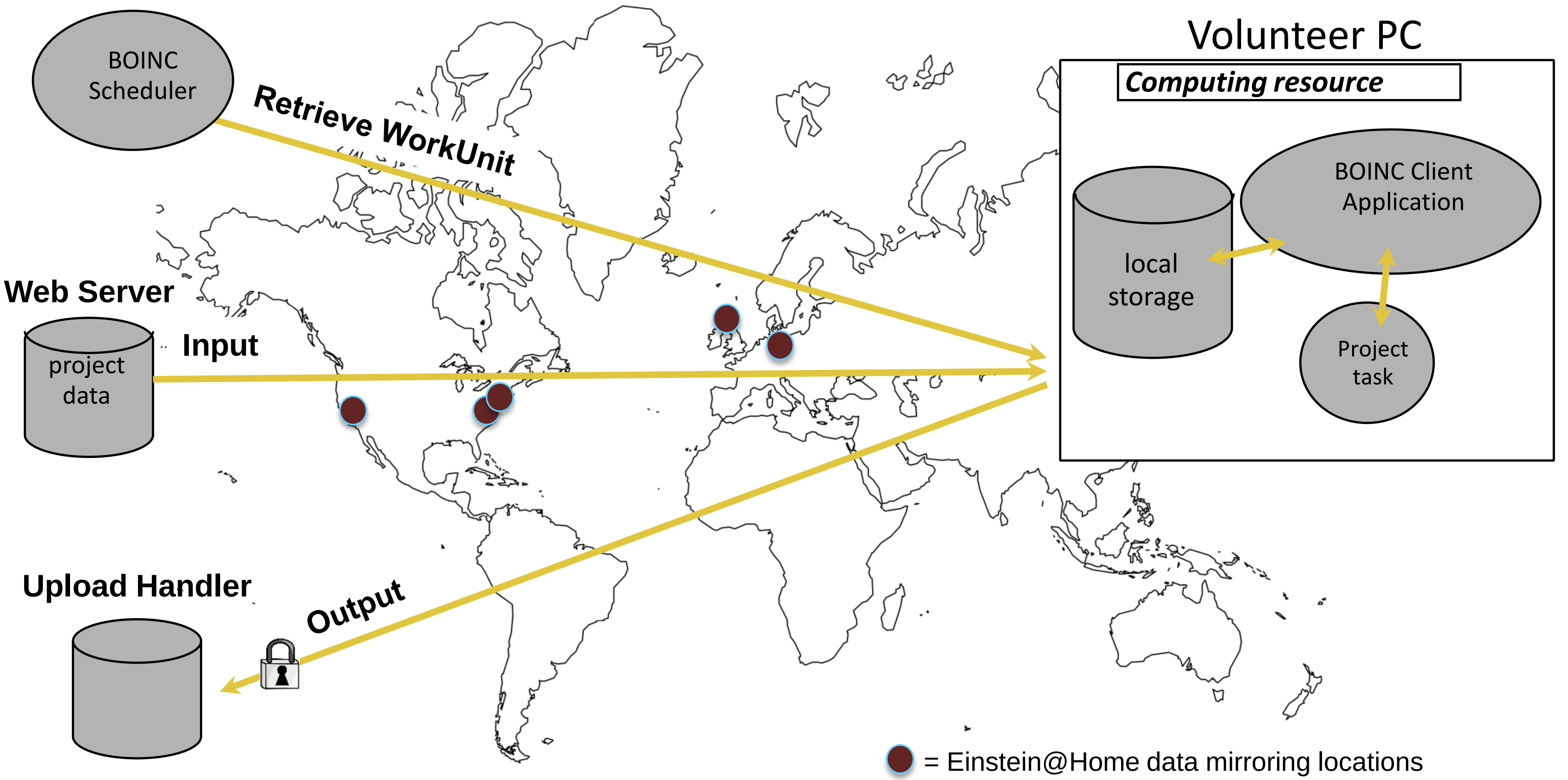


EGEE Data Access

- Files stored on secure repository
- Referenced by LFNs
  - resolve to concrete replica locations
- Similar data access patterns for ARC & Unicore



BOINC Data Distribution



Typical BOINC Applications

- **SETI@Home**
  - Size of a Work Unit: 340 KB
  - Processing Time of a Work Unit: 2h
  - Size of Initial Data: 2.5 MB

SETI@Home				
Replication	Task Size	Upload Rate	Processing Time	Tasks Per Day
2	340 KB	50 MB/s	≈2 hours	≈1,000,000

- **Einstein@Home**
  - Size of a Work Unit: 3.2 MB
  - Processing Time of a Work Unit: 5h
  - Size of Initial Data: 40 MB

Einstein@Home				
Replication	Task Size	Upload Rate	Processing Time	Tasks Per Day
Gravitational Wave Analysis				
2	6–7 MB [a]	(5 mirrors)	≈5 hours	≈75,000
Binary Radio Pulsar				
2	≈32 MB[b]	30 MB/s	≈40 min. (GPU)	≈700,000

EDGI DG Applications

- **Fusion Physics Application**
  - Institute for Biocomputation and Physics of Complex Systems
  - Execution time: ~30 minutes
  - Input files: ~10 MB
- **Material Science Applications**
  - G.V. Kurdyumov Institute for Metal Physics
  - Execution time: ~30 min per scenario
  - Input files: 1 – 10 MB
  - Jobs: 10^3 – 10^4 per day
- **Signal-and Image Processing**
  - Forschungszentrum Karlsruhe
  - Execution time: 4 days
  - Input files: ~20 GB

Name	Files/WU	File Size[gh]	Output[g]	Exec Time	Tasks/Day[g]
ISDEP	6[a]	2	1	30 min.	50,000
pLINK	2	380	0.5		
ViSAGE	2	2	1	1 min.	10,000
Desktop Grid Pattern Finder (DGPF)	2	0.5	0.5	5 min.	3,100
Distributed Audio Retrieval using Triana (DART)	6[b]	52	0.01	2 min.	1,000
itemgrid	1	<1	<1	60 min.	1,000
E-Marketplace Model Integrated with Logistics (EMMIL)	1[c]	1	10	10 min.	1,000
X-ray	0.1[d]	0.2	1	20 min.	10,000
VisIVO	4[e]	1000	50	30 min.	2,000
GT4Tray	1 – 1000	1 – 1000	1 – 1000	1 – 240 min.	200
Multiscale Image and Video Processing	5[f]	1	1	20 min.	1024
Sequence Correlations	1	100	50	180 min.	100,000
MOPAC	3	0.1	0.5	4 min.	100,000

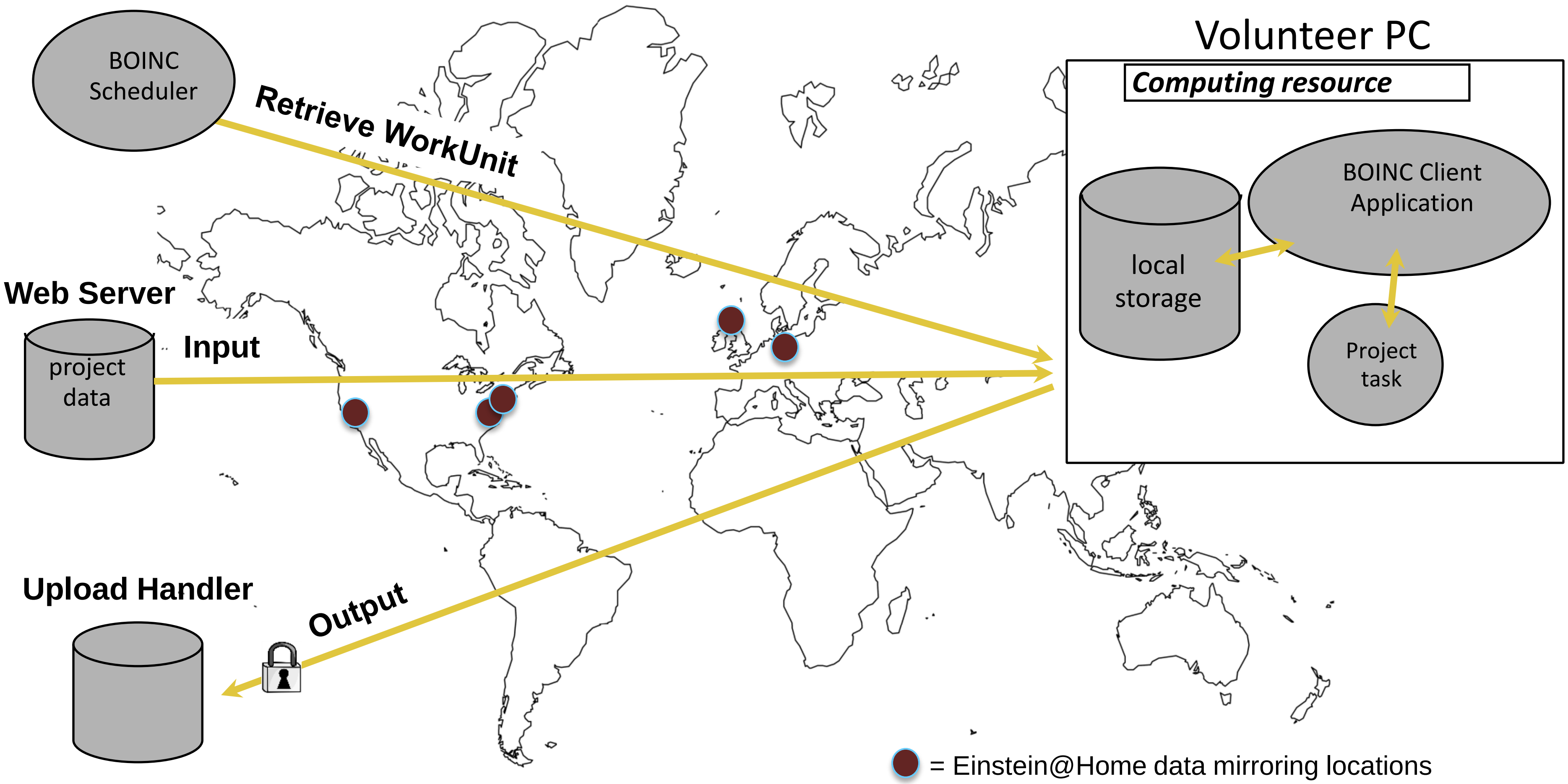
EDGI DG Applications

- **Fusion Physics Application**
  - Institute for Biocomputation and Physics of Complex Systems
  - Execution time: ~30 minutes
  - Input files: ~10 MB
- **Material Science Applications**
  - G.V. Kurdyumov Institute for Metal Physics
  - Execution time: ~30 min per scenario
  - Input files: 1 – 10 MB
  - Jobs: 10^3 – 10^4 per day
- **Signal-and Image Processing**
  - Forschungszentrum Karlsruhe
  - Execution time: 4 days
  - Input files: ~20 GB

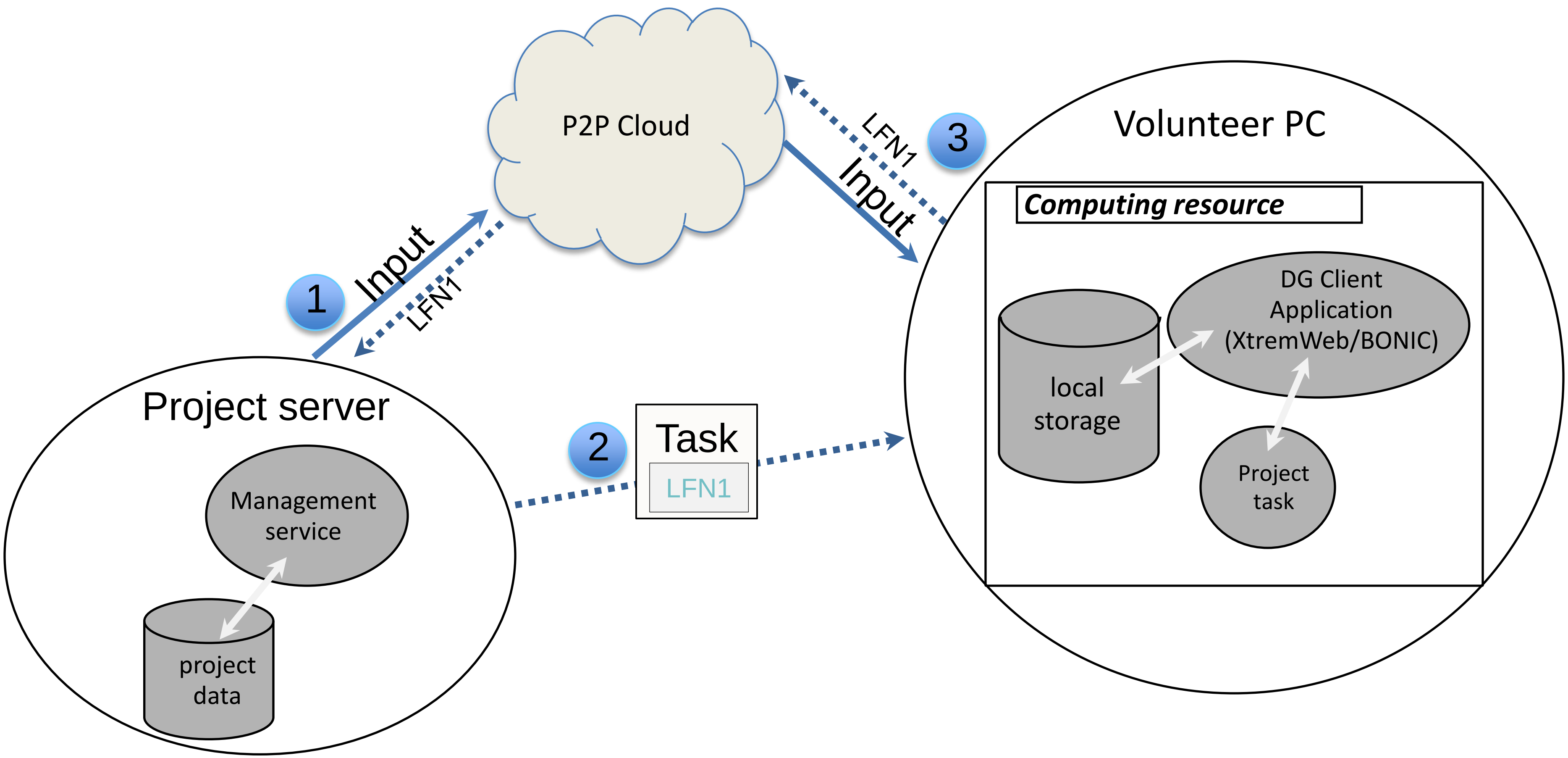
Name	Files/WU	File Size[gh]	Output[g]	Exec Time	Tasks/Day[g]
ISDEP	6[a]	2	1	30 min.	50,000
pLINK	2	380	0.5		
ViSAGE	2	2	1	1 min.	10,000
Desktop Grid Pattern Finder (DGPF)	2	0.5	0.5	5 min.	3,100
Distributed Audio Retrieval using Triana (DART)	6[b]	52	[illegible]	2 min.	1,000
itemgrid	1	<1	<1	60 min.	1,000
E-Marketplace Model Integrated with Logistics (EMMIL)	1[c]		10	10 min.	1,000
X-ray	0.1	0.2	1	20 min.	10,000
VisIVO	4	1000	50	30 min.	2,000
GT4Tray	1 – 1000	1 – 1000	1 – 1000	1 – 240 min.	200
Multiscale Image and Video Processing	5[f]	1	1	20 min.	1024
Sequence Correlations	1	100	50	180 min.	100,000
MOPAC	3	0.1	0.5	4 min.	100,000

Bigger Data

BOINC Data Distribution



BOINC Data Distribution



Desktop Grid Issues

- Overall bandwidth requirements can be high, especially with replicated jobs
- Project's need persistent data webserver, and potentially N mirrors to balance load
  - For smaller groups servers might be hard to maintain or mirror
  - For Service Grids, data might be restricted and it would be useful to have a staging ground for DG data.
- Network peak demand problem

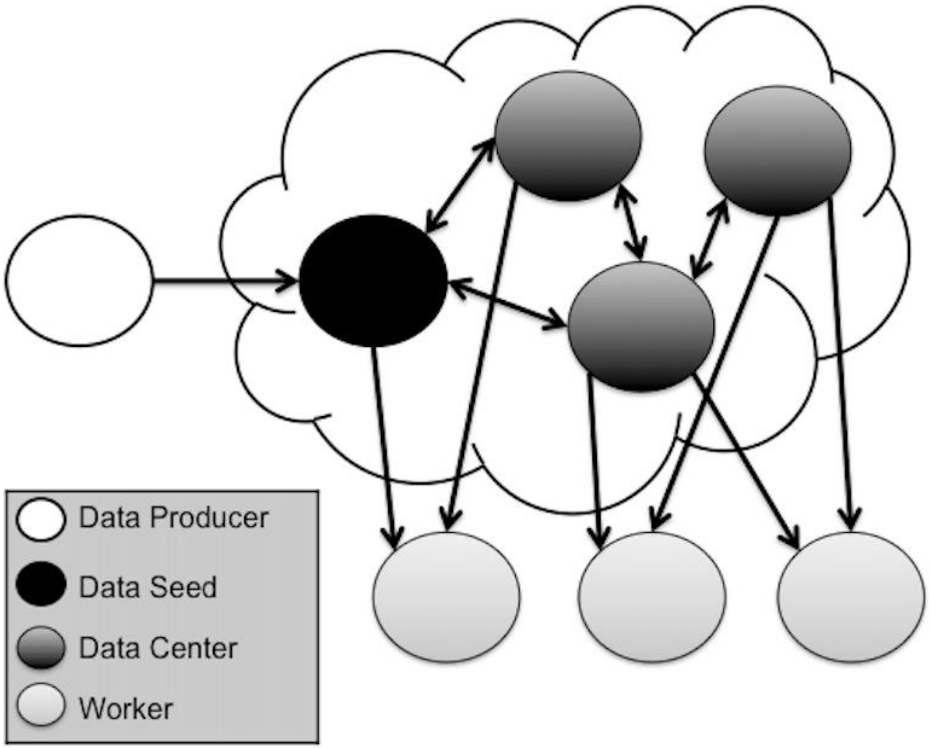
➢ Possible to construct a "P2P" system using clients and/or (potentially dynamic) set of project/partner servers to serve and cache input data

Desktop Grid Data Issues

- General architecture requirements
  - Need to protect end-users and have opt-out system
    - compulsory open ports on all workers is not possible
  - Protect the project's data
    - may want limited caching on any given peer to limit exposure
    - need to ensure data integrity and potentially have authentication techniques for data cachers
  - Beneficial to support different network topologies (WAN, LAN)

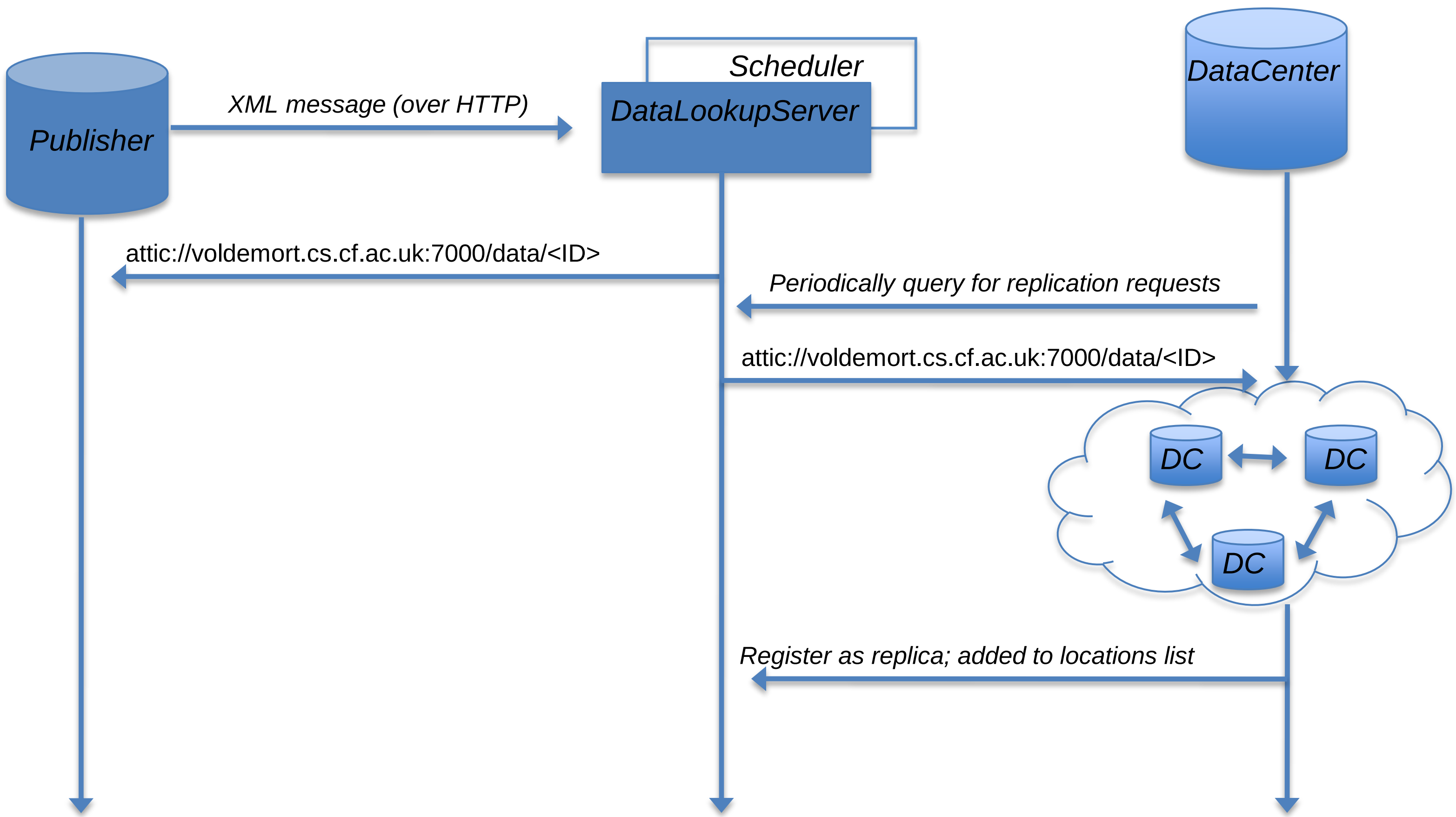
  - *These requirements discount many established P2P systems such as BitTorrent*

Distributed Data Centers



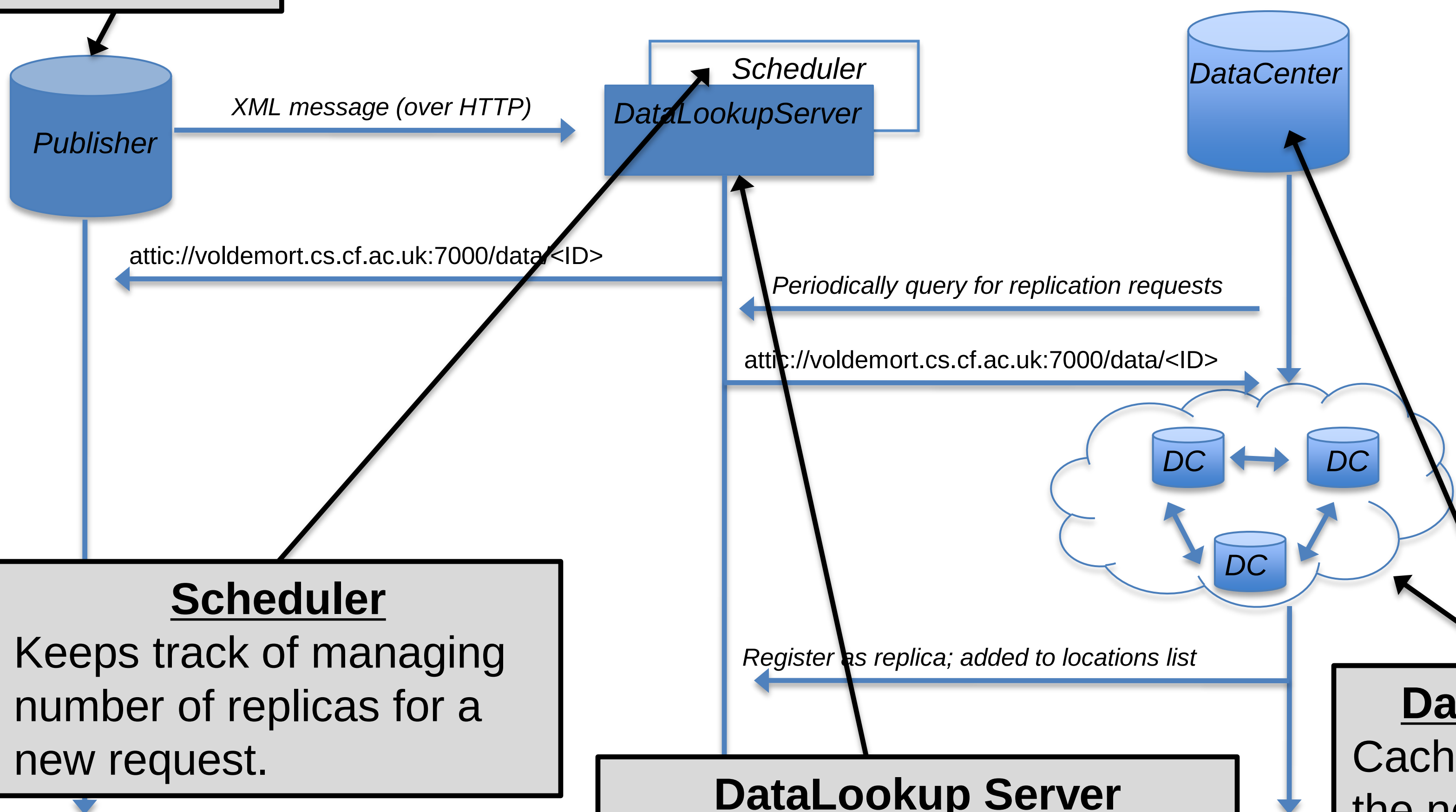
- ***Data Caching layer***
  - Data Caching peers exchange data amongst themselves and serve client machines
  - Authentication can be turned on between Data Cachers (Data Centers)

Component Overview



Component Overview

Publisher
Any entity that publishes a file



Scheduler
Keeps track of managing number of replicas for a new request.

DataLookup Server
Manages number of replicas for a new request and responding to cache requests

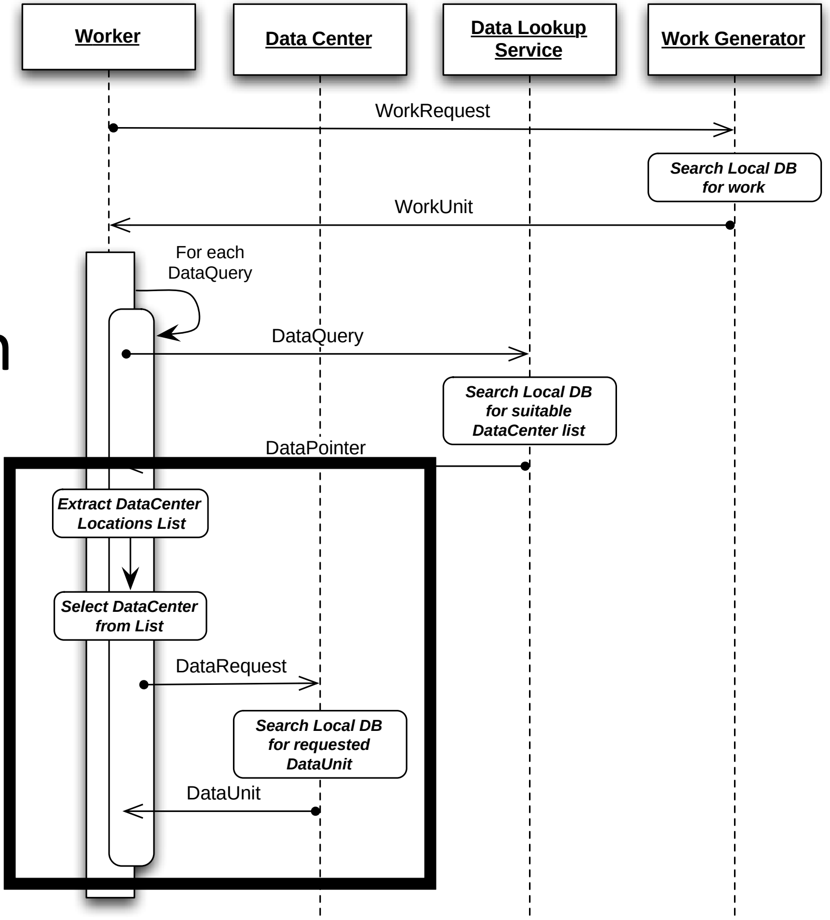
DataCenter
Caches data on the network.

Terms

- DLS – Data Lookup Service
  - receives requests to publish data
  - receives requests to cache data
  - does not store any data, only keeps mappings between endpoints and data
  - acts as a scheduler as well – controls exposure of data according to constraints defined by the publisher
- DP – Data Publisher
  - publishes an advert to the DLS about data
  - typically the DP is also a seed endpoint (but not always)
- DC – Data Center
  - requests data references from the DLS
  - caches data from other endpoints
- Worker
  - downloads data from DCs for processing.

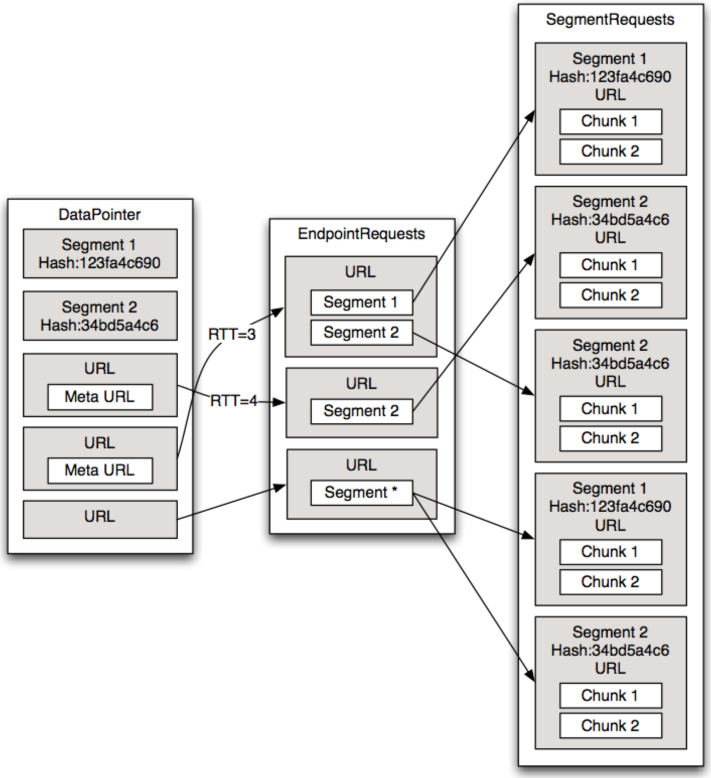
Attic

- Network participants distribute data
- Opt-in strategy
- Restricted publication of data
- URI scheme
- File swarming
  - By simultaneously downloading different chunks from multiple DataCenters
-

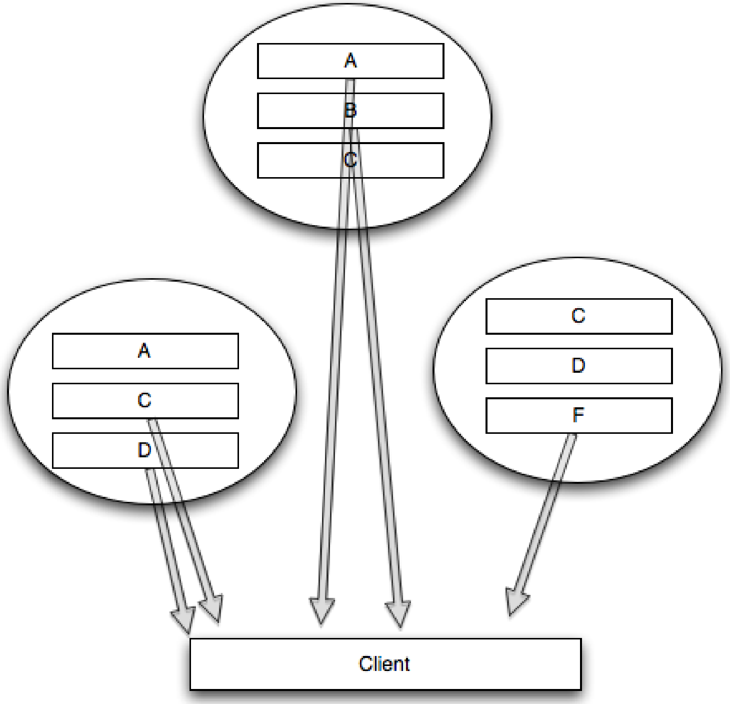


File Distribution

Files can be split into individual chunks for distribution to data caching layer.

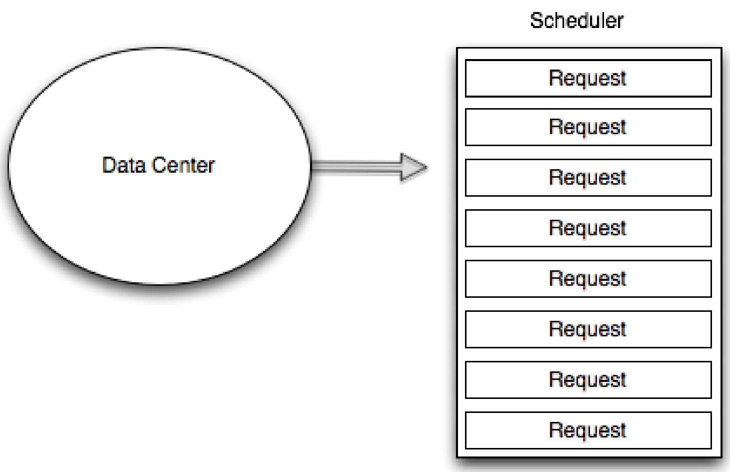


Clients can download different parts of the file from multiple data centers.

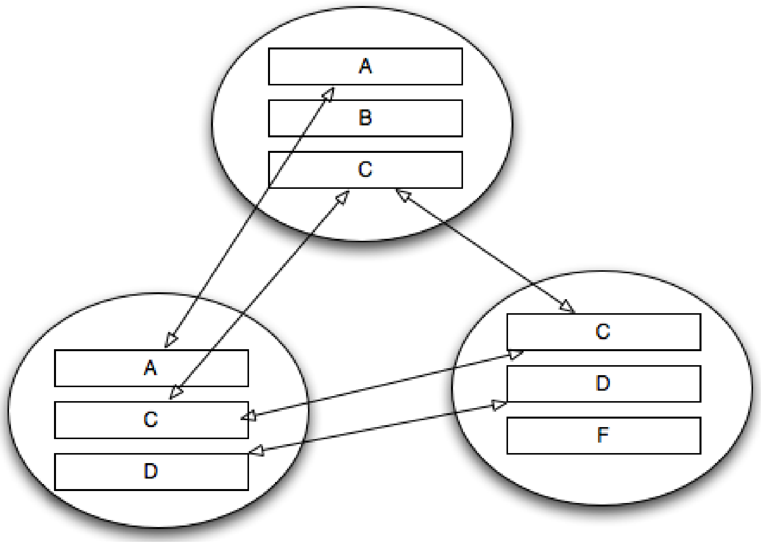


Data Center Caching

Data cachers contact ascheduler to receive replication requests.



They then download from one-another to propagate data on the network



Message Types

- DataDescription
  - contains metadata, e.g., name, description, project
  - file data, e.g. size, MD5, and a list of chunks with byte ranges and MD5s
- DataAdvert
  - contains DataDescription
  - Constraints, e.g., replication count
  - Used when publishing data
- DataQuery
  - contains Constraints
  - Used when Querying for data to cache/replicate
- DataPointer
  - contains DataDescription
  - List of endpoints associated with the description
  - Returned to a query for data
  - The data structure pointed to by an attic:// URL

Data Pointer

XML

```xml
<?xml version="1.0" encoding="UTF-8"?>
<DataAdvert xmlns="http://atticfs.org">
   <DataDescription xmlns="http://p2p-adics.org">
      <id>12c667d6-2d5d-4904-9c2c-6746251b81ef</id>
      <name>SimulationInputFile.dat</name>
      <project>EDGeS</project>
      <description>Input for simulation</description>
      <FileHash>
            <hash>661c7f5e462be8ced9a8a6d8a1c7e6</hash>
            <size>12407432</size>
            &lt;Segment&gt;
               <hash>bedbfd11fa5fb4fd6b97349f45b6b3</hash>
               <start>0</start>
               <end>524287</end>
            &lt;/Segment&gt;
            ...
            ...
            ...
            &lt;Segment&gt;
               <hash>32ce5368d98243a2a9abeccc2ddc5c</hash>
               <start>12058624</start>
               <end>12407431</end>
            &lt;/Segment&gt;
      </FileHash>
   </DataDescription>
   <Constraints>
         <Constraint type="Date">
            <key>expires</key>
            <value>Sat Jun 30 23:59:59 GMT 2012</value>
         </Constraint>
         <Constraint type="Integer">
            <key>replica</key>
            <value>3</value>
         </Constraint>
      </Constraints>
</DataAdvert>
```

JSON

```json
{
"DataAdvert": {
   "DataDescription": {
       "id": "b426c41b-d5a3-4138-93ec-60a8be2a6c0c",
       "name": "SimulationInputFile.dat",
       "project": "EDGeS",
       "description": "Input for simulation",
       "FileHash": {
           "hash": "661c7f5e462be8ced9a8a6d8a1c7e6",
           "size": 12407432,
           "Segment": [
               {
                "hash": "bedbfd11fa5fb4fd6b97349f45b6b3",
                "start": 0,
                "end": 524287
               },
               ...
               ...
               ...
               {
                "hash": "32ce5368d98243a2a9abeccc2ddc5c",
                "start": 12058624,
                "end": 12407431
               }
           ]
       }
   },
   "Constraints": {
       "Constraint": [
           {
               "type": "Date",
               "key": "expires",
               "value": "Sat Jun 30 23:59:59 GMT 2012"
           },
           {
               "type": "Integer",
               "key": "replica",
               "value": "3"
           }
       ]
   }
}
}
```

Information School
UNIVERSITY of WASHINGTON

Message Types

- https://voldemort.cs.cf.ac.uk:7048/dl/meta/pointer/dceae487-bb18-4dfd-9391-3a4b701b1fb7

– <DataPointer>
– <DataDescription>
<id>dceae487-bb18-4dfd-9391-3a4b701b1fb7</id>
<name>dceae487-bb18-4dfd-9391-3a4b701b1fb7.dat</name>
<project>edges</project>
<description>Test file</description>
– <FileHash>
<hash>d6a5f4aae746e18c92f18eaba9d77c61</hash>
<size>6435839</size>
– <Segment>
<hash>44bc74bb4d6225b8b8e68ea6848a57</hash>
<start>0</start>
<end>524287</end>
</Segment>
– <Segment>
<hash>bf5b8da3143d769241b21675347691</hash>
<start>524288</start>
<end>1048575</end>
</Segment>

Message Types

- https://voldemort.cs.cf.ac.uk:7048/dl/meta/pointer/dceae487-bb18-4dfd-9391-3a4b701b1fb7


```
      - &lt;Segment&gt;
          <hash>cdabbd2444a8b3c182a69528cb119c1</hash>
          <start>5767168</start>
          <end>6291455</end>
        &lt;/Segment&gt;
      - &lt;Segment&gt;
          <hash>1e6fe5fa73723a1cc3b02f8b5a3cc3d5</hash>
          <start>6291456</start>
          <end>6435838</end>
        &lt;/Segment&gt;
      </FileHash>
    </DataDescription>
  - <Endpoint>
    - <url>
        https://d220.cs.cf.ac.uk:7049/dp/data/dceae487-bb18-4dfd-9391-3a4b701b1fb7
      </url>
    </Endpoint>
  </DataPointer>
```

Message Types

- https://voldemort.cs.cf.ac.uk:7048/dl/meta/pointer/dceae487-bb18-4dfd-9391-3a4b701b1fb7


```
    - &lt;Segment&gt;
        <hash>1e6fe5fa73723a1cc3b02f8b5a3cc3d5</hash>
        <start>6291456</start>
        <end>6435838</end>
      &lt;/Segment&gt;
    </FileHash>
  </DataDescription>
- <Endpoint>
  - <url>
      https://d220.cs.cf.ac.uk:7049/dp/data/dceae487-bb18-4dfd-9391-3a4b701b1fb7
    </url>
  </Endpoint>
- <Endpoint>
  - <url>
      https://electricline.cs.cf.ac.uk:7047/dc/data/dceae487-bb18-4dfd-9391-3a4b701b1fb7
    </url>
    <meta>https://electricline.cs.cf.ac.uk:7047/dc/meta</meta>
  </Endpoint>
</DataPointer>
```

Information School
UNIVERSITY of WASHINGTON

Message Types

- https://d220.cs.cf.ac.uk:7049/dp/meta/filehash/dceae487-bb18-4dfd-9391-3a4b701b1fb7

– <FileHash>
<hash>d6a5f4aae746e18c92f18eaba9d77c61</hash>
<size>6435839</size>
– <Segment>
<hash>44bc74bb4d6225b8b8e68ea6848a57</hash>
<start>0</start>
<end>524287</end>
</Segment>
– <Segment>
<hash>bf5b8da3143d769241b21675347691</hash>
<start>524288</start>
<end>1048575</end>
</Segment>
– <Segment>
<hash>e69da54b2e42c423364640ad490dd4</hash>
<start>1048576</start>
<end>1572863</end>
</Segment>

File Chunk info available from meta endpoint

Message Types

- Once a Data Center has downloaded the data and notified the Data Lookup Service, it appears in the DataPointer.
  - i.e., it gets added to the replica list
- The metadata endpoint is where clients can get meta info about data from a Data Center
- Note: the seed does not provide a metadata endpoint
  - Therefore it becomes a fallback endpoint during downloading
  - As more DCs get the data, the seed becomes redundant

Protocols

- Uses HTTP(S) for all exchanges
  - message and data
  - uses HTTP byte ranges to specify chunks
- Message serialization
  - default serialization is JSON (JavaScript Object Notation)
  - also XML (e.g., for demo)
  - JSON is about 1/3 to 1/2 as verbose as XML
    - but still Unicode

Protocols

- Why HTTP?
  - Attic is about data. HTTP is good at data.
  - allows nodes to take part transparently, for example a server without knowledge of Attic may be used as a fallback during downloading. It exposes no metadata, but responds to byte range requests
  - easy integration with other systems, e.g., BOINC uses curl libs.
  - Allows use of common libraries to directly download data and/or build new clients/servers

Security

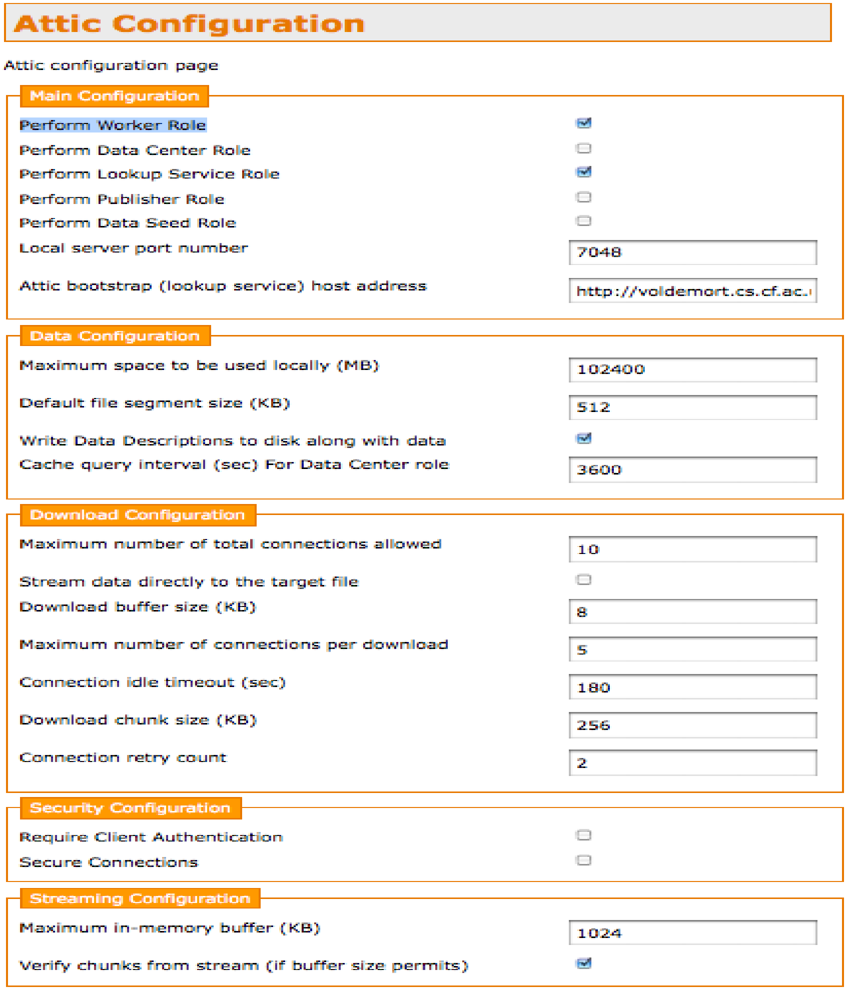
- Authentication (optionally enabled) uses X.509 certificates with TLS
  - mutual
- Authorization is done using the Distinguished Name (DN) in the peer's (e.g., DC) certificate
  - Identities based in DNs are mapped to actions, e.g., PUBLISH, CACHE
  - E.g., a Worker may only need a certificate signed by a CA trusted by a DC to download from that DC
  - But a DC may need the above, as well as its DN mapped to the CACHE action in order to cache data

Download Features

- Rebuilding data from multiple nodes with only partial data
  - before downloading, a metadata request is made to discover chunks at an endpoint
- Endpoint selection based on
  - availability of metadata endpoint
  - RTT of metadata request before download
  - endpoint history
  - duplicate chunks at lower priority endpoints can be used in the event of errors
- Chunk prioritization based on
  - sequentially (used for streaming)
  - endpoint status (fastest first)

Configuration Features

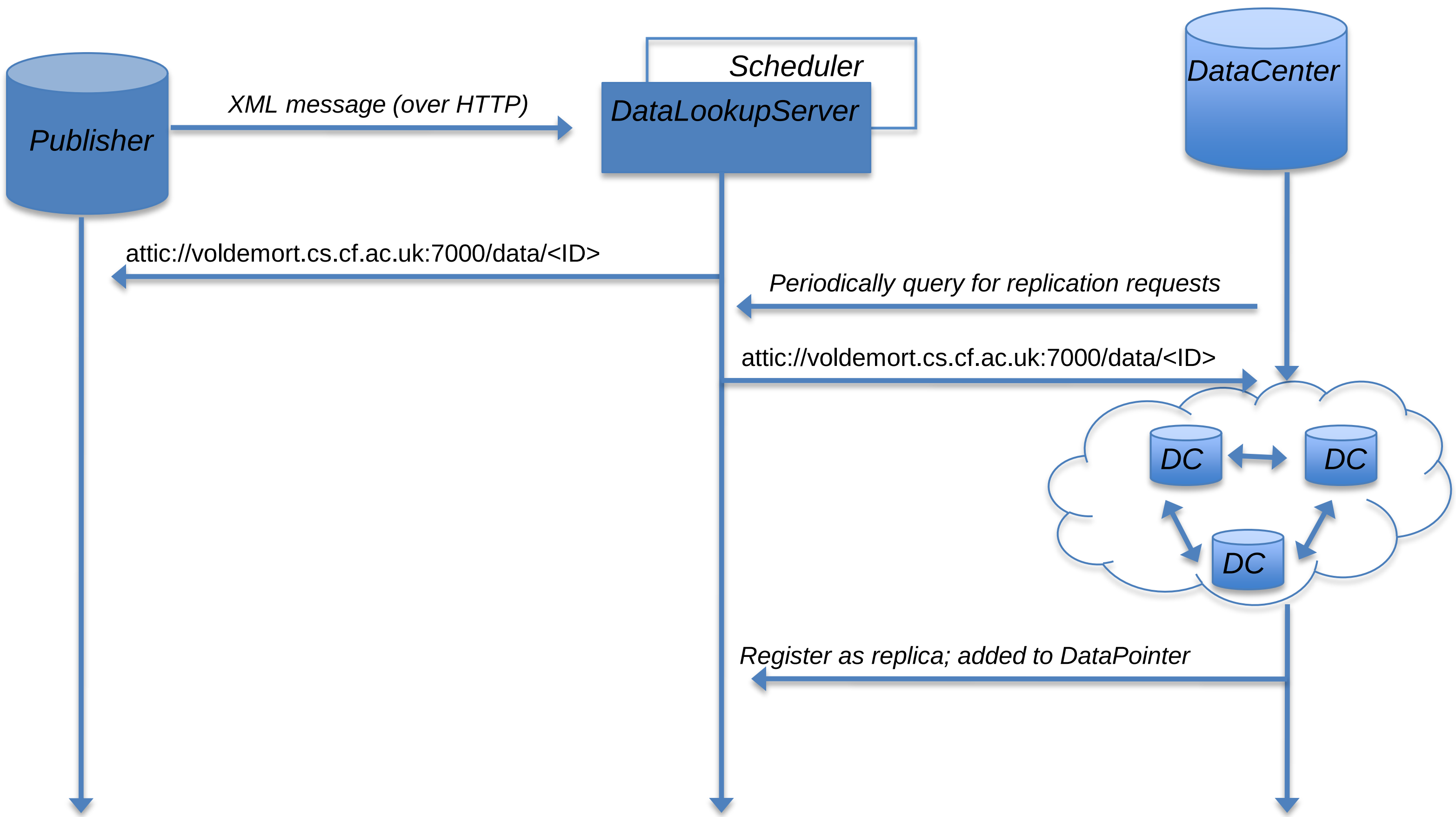
- Web access (TLS mutual authentication)
- Options include:
  - Role(s)
  - Disk space usage and download file type (single file & multiple files that are rebuilt)
  - Connection settings
    - chunk size, number of connections overall/per download, memory footprint, security



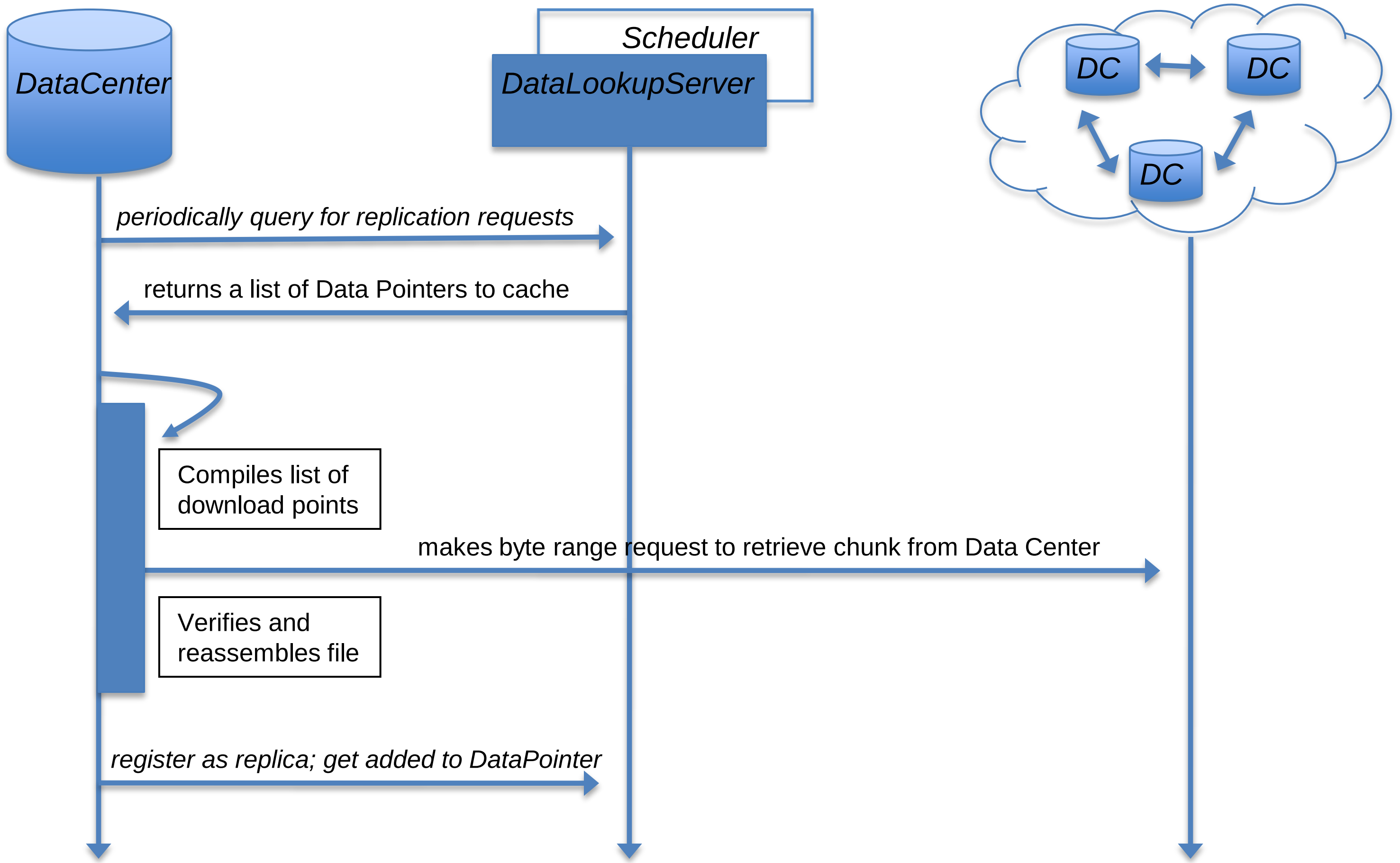
Publish Data to Attic

- Option 1: Use native Java Libraries
- Option 2: Use curl-based CLI w/ Data Seed node
  - Used by the EDGI 3GBridge
    - needs no knowledge of Attic protocol
    - Just a single .sh script to register and send data
    - requires curl on the $PATH
    - main parameters
      - local file to send
      - seed HTTP endpoint
      - certs/keys for mutual authentication
      - others (project, expiry, replica, etc)
    - Outputs Attic URL e.g., `attic://dls.org/1234`

Attic: Publishing



Attic: Data Center Overlay

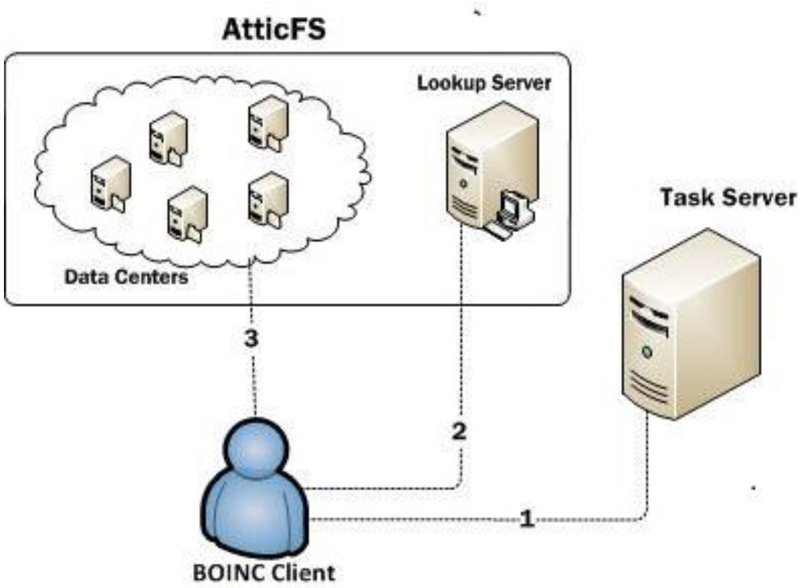


Direct Integration

- Attic URL Stream Handler
  - Java component that handles URLs with an attic scheme.
  - takes an attic URL e.g., **`attic://dls.org/1234`**
  - returns a java.io.InputStream for reading the data.
  - Requires that the application is:
    - written in Java
    - registers the Attic URL handler.
  - Based on the configuration and data chunks, the stream handler will attempt to verify chunks before passing them to the application

Project – BOINC (w/Proxy)

- Using Attic instead of HTTP in the download URL.
- Reference Data Lookup Server for locating data centers that have input data.
- Generate work units using attic:// URL instead of http://



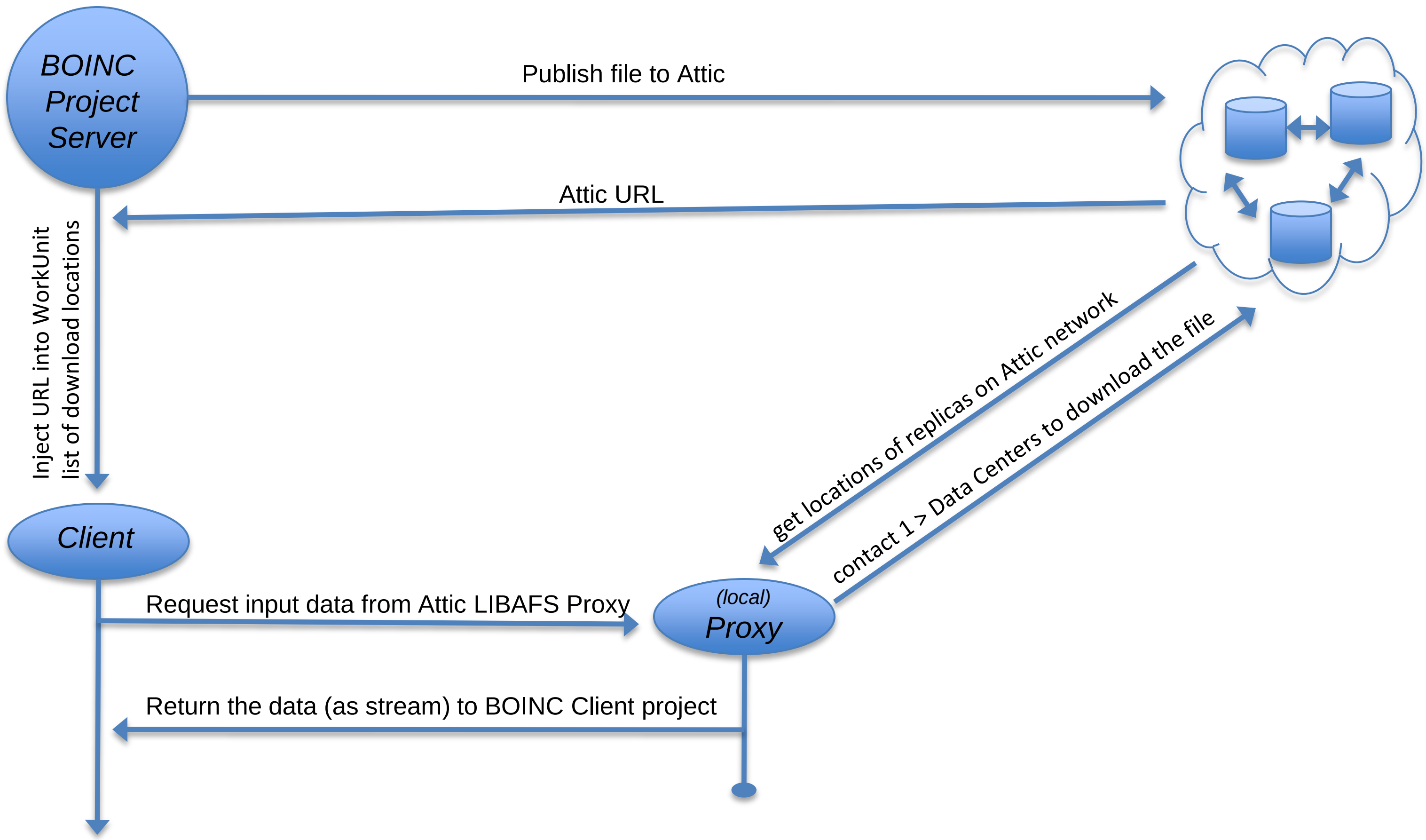
Project – BOINC (w/Proxy)

- Attic “libafs” BOINC proxy client
  - Native-C BOINC project
  - Runs a local web server to intercept URL requests. i.e., **http://localhost:port/<file-identifier>**
  - Required no additional (project) modification to the BOINC client code, and only minor modification to the server to inject work-unit endpoint and MD5s
    - Except subscription to the new project…
  - Did not “break” anything or endanger BOINC, as there can be automatic fall-over to the next replica URL.
  - Could easily be adapted to intercept attic:// protocol requests

(this would have required changes to BOINC code)

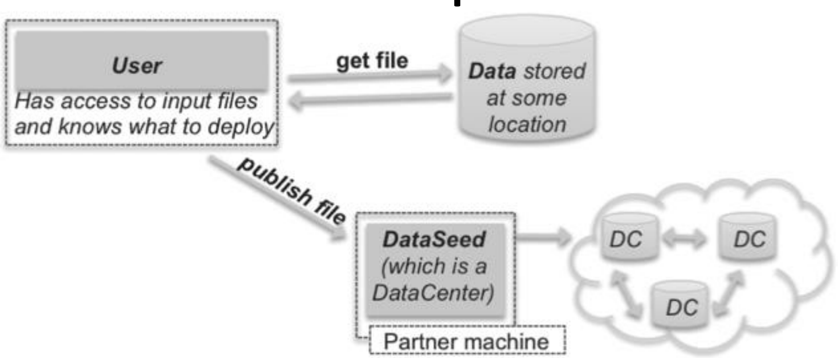
Project – BOINC (w/Proxy)

http://www.atticfs.org/libafs



EDGI Use-Case

- Deployed Attic within EDGI project as way to distribute Service Grid data
- Each project partner (total: 10) allocated a Data Center node
- Files coming from Service Grid users could be distributed to this Attic layer, giving DG clients a download endpoint.



Institute	Abbreviation	Country
Laboratory of Parallel and Distributed Systems	MTA SZTAKI	Hungary
University of Westminster	UoW	United Kingdom
University of Paderborn	UPB	Germany
University of Copenhagen	UCPH	Denmark
AlmereGrid	—	The Netherlands
University of Coimbra	FCTUC	Portugal
University of Zaragoza	UNIZAR	Spain
Cardiff University	CU	United Kingdom
National Center for Scientific Research	CNRS	France
National Institute for Research in Computer Science and Control	INRIA	France

Science Cloud Use-Case ?

- Use distributed architecture as a way to share research data (also in an ad-hoc manner)
- A collaborator might deploy one or more Data Center nodes
  - To serve "long term" data
  - Or, to distribute and load balance in a new environment
- Files coming from various resources could be first distributed *to* new layer and then *within* it
  - Providing local copy decoupled from original source
  - Ability to proxy information going out and coming in

My Thoughts

- Important attributes
  - Low barrier to entry as end-user (important!)
  - Ability to leave data “where it is” and still use it
  - Able to proxy data, isolating while bridging data providers and consumers
    - Useful for bridging data to closed-off systems (e.g., clusters), and/or leveraging network structure
  - Way for data to live “beyond” research projects?
    - b/c others can replicate
  - Way for low-resource projects to share
    - Heavy lifting can be done “by the network”

Your Thoughts?