www.bsc.es

Barcelona Supercomputing Center
Centro Nacional de Supercomputación

Data sharing in the BigData era

Toni Cortes and Anna Queralt

BigDataCloud 2014, Europar 2014, Porto

Agenda

- **Motivation and background**
- Vision
- Technological details
- Conclusions

Before everything started …

- What ignited our research “bigbang”
  - Different data models: Persistent vs. non persistent
  - New storage devices: byte addressable
  - Sharing is what really matters

- And then **dataClay** came to life …
  - (more details on how all fits together
  - In the next 50 minutes)

BigData problems

- **Big data problems (3Vs)**
  - **Volume**: money
  - **Velocity**: money (replication HW, model, …)
- **Variety**: more a data-model problem than storage technology

I know I am being over simplistic

But …

… where is the problem then?

Let me be a bit more controversial

- If we store the weigh of all rain drops

- Is this big data?

  big volume,
  velocity (many drops per second),
  and variety (info from many sources)

- Can we extract any **valuable** information?
- No value, no big data
- **Real value comes from sharing and combining**

Why sharing data is important?

- Cooperation is the way to success

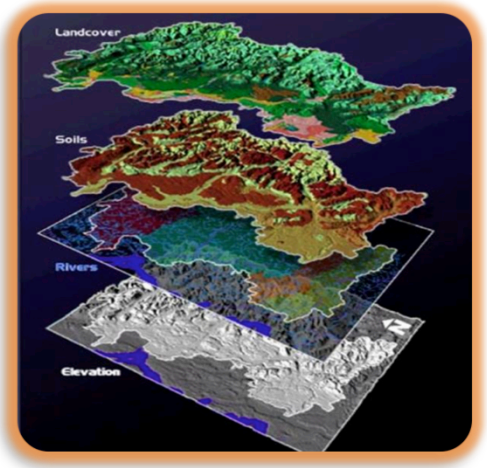
- Key information comes from **merging data from different sources**

- Data sources: public and open or private (not shared)

Some examples of what we want to achieve

- Geospatial

  Share and enrich data



- Newspaper archival

  Control over data

- Health

  Privacy and flexibility

How is data shared today?

- **Real sharing**: all actors have full access to infrastructure
  - Huge trust alliances or irrelevant data
  - Very flexible

- **Data copies:** owner decides what can be copied
  - Unnecessary data movement
  - Stale data
  - Owner loses control over data
  - Very flexible

- **Data services**: owner decides what and how data is shared
  - Very restrictive
  - Changes imply data provider involvement
  - Owner keeps full control

Agenda

- Motivation and background
- **Vision**
  - **Self containment and enrichment**
  - “New” programming model
- Technological details
- Conclusions

Our vision

- Enable all actors to
  - "Share" an infrastructure
  - Merge all data in a "single " data set
  - Upload computations to be shared
  - See different "views" of the data

- Key idea: **self-contained and** data **enrichment** by 3rd parties

Key technology: self-contained objects

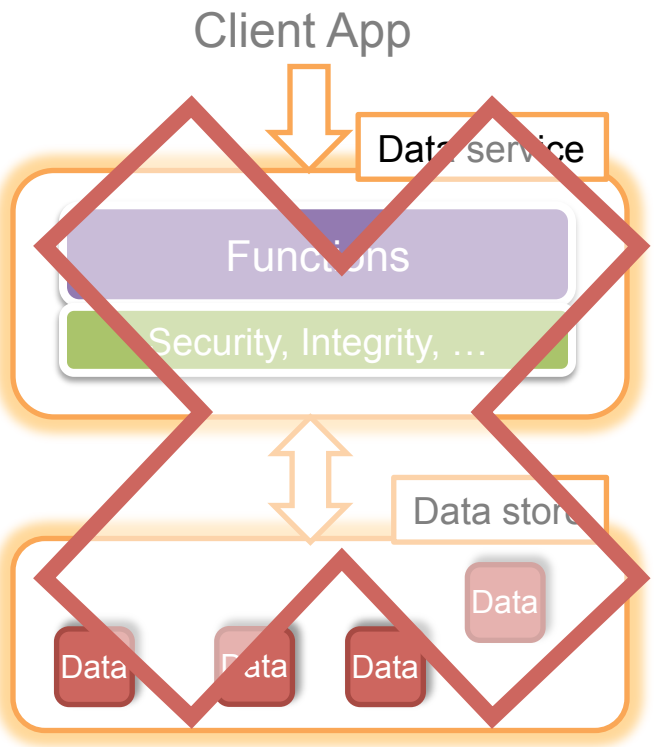
- **Self-contained objects**
  - Data
  - Code
  - Behavior policies

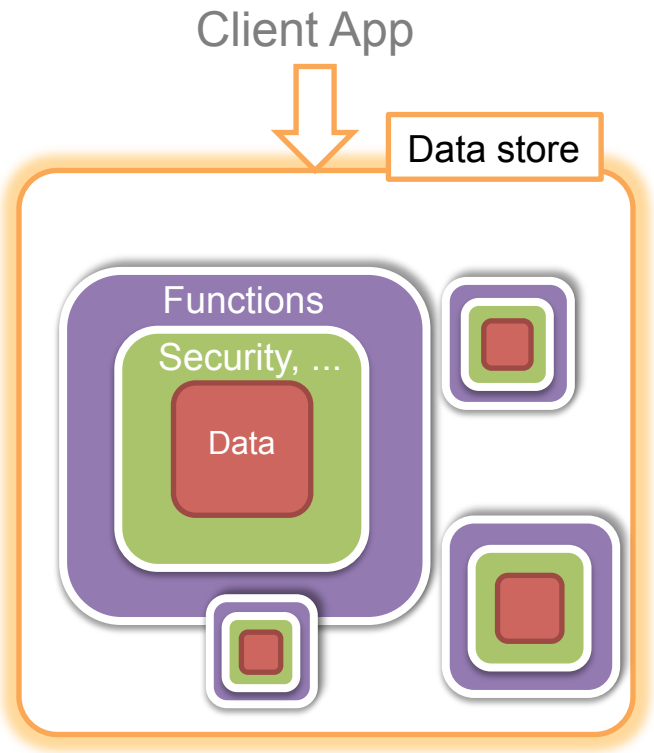
- But …

… this looks much like a data service

Self-contained objects

- Push the idea of data services to the limit





Key-technology: 3rd party enrichments

« Self-contained objects
seem to be a new technology to offer
data services in a different way

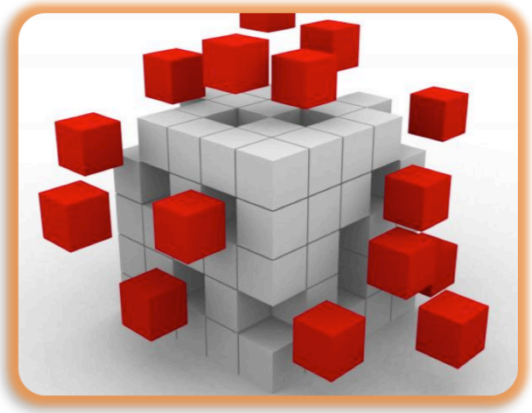
« Then…

… we need something else …

… something to **make it** really **flexible**!

3rd-party data enrichment

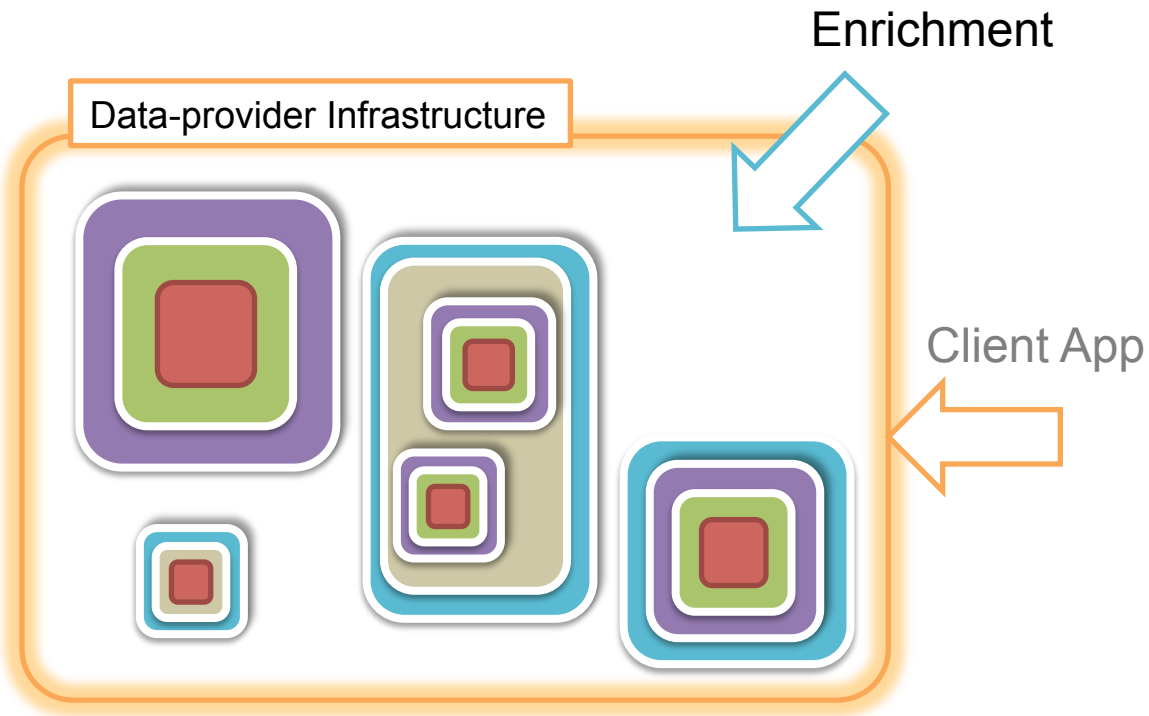
- By enrichment we understand:
  - Adding new information to existing datasets
  - Adding new code to existing datasets



- This enrichment should
  - Be possible during the life of data
  - Not be limited to the data owner
  - Enable different views of the data to different users/clients
  - Not everybody should be forced to see the same enrichments
  - Several enrichments should be available concurrently

Then…

- Data can be enriched both with **data and code**, in provider infrastructure
  - Code can be executed in the provider infrastructure



Data integration in a single infrastructure?

- Using a “single” infrastructure may become a bottleneck

- Security and privacy policies should be part of the data
- Thus, data could be offloaded to other infrastructures

  Without breaking the data policies
- Data owner enables 3rd party enrichment and …
- … does not lose control

And now … behavior policies

- Behavior policies include
  - Privacy
  - Security
  - Integrity
  - Life cycle
  - Etc.

- Who manages them
  - Traditionally: platform and/or infrastructure
  - Self-contained objects: the object itself
    - If each object manages its own behavior, the system is far more scalable

Implementing behavior policies

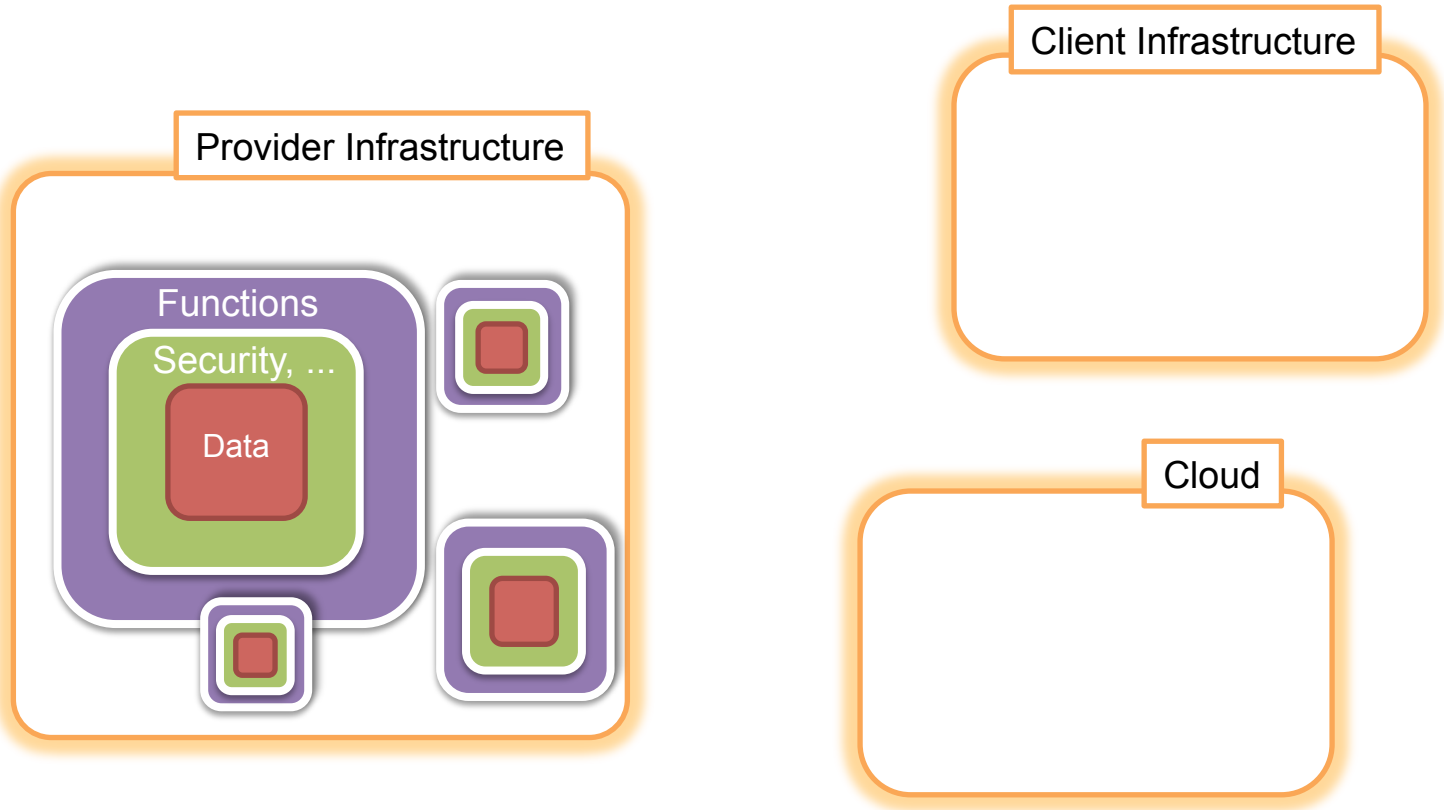
- How it is implemented
  - Policies implemented as part of object methods
- How are policies defined
  - Defined using traditional declarative languages (rules)
  - Compiled and injected into class methods

Moreover…

Efficient usage of resources

Data and code can be offloaded to resources not accessible by the data provider

Also for storage ➔ reduce data movements if data is placed close to client



Agenda

- Motivation and background
- **Vision**
  - Self containment and enrichment
  - **“New” programming model**
- Technological details
- Conclusions

Why persistent data is different than volatile?

Today

We have one data model for volatile data

Traditional data structures and/or objects

We have a different data model for the persistent data

Relational database, NoSQL database, files

Future

Store data in the same way as when volatile

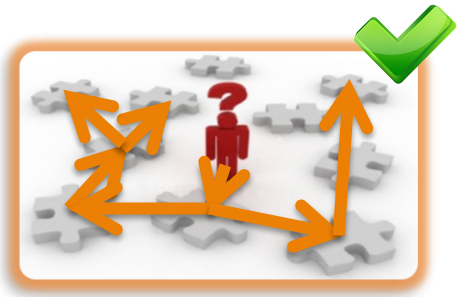
Store objects and relations

Data selection (Queries make no sense anymore)

- Enrichment enables accessing persistent data as if in memory

- In memory:
  - Data “never” queried
  - Data linked according to needs of program
    - Next data item found by following a link, not a query

- Persistent data should behave in a similar way
  - Following a link is faster than a query over the whole dataset
  - Programs do not need to make any differences whether
    - Data is in memory or in persistent storage
  - Enrichments enable data to be linked in different ways

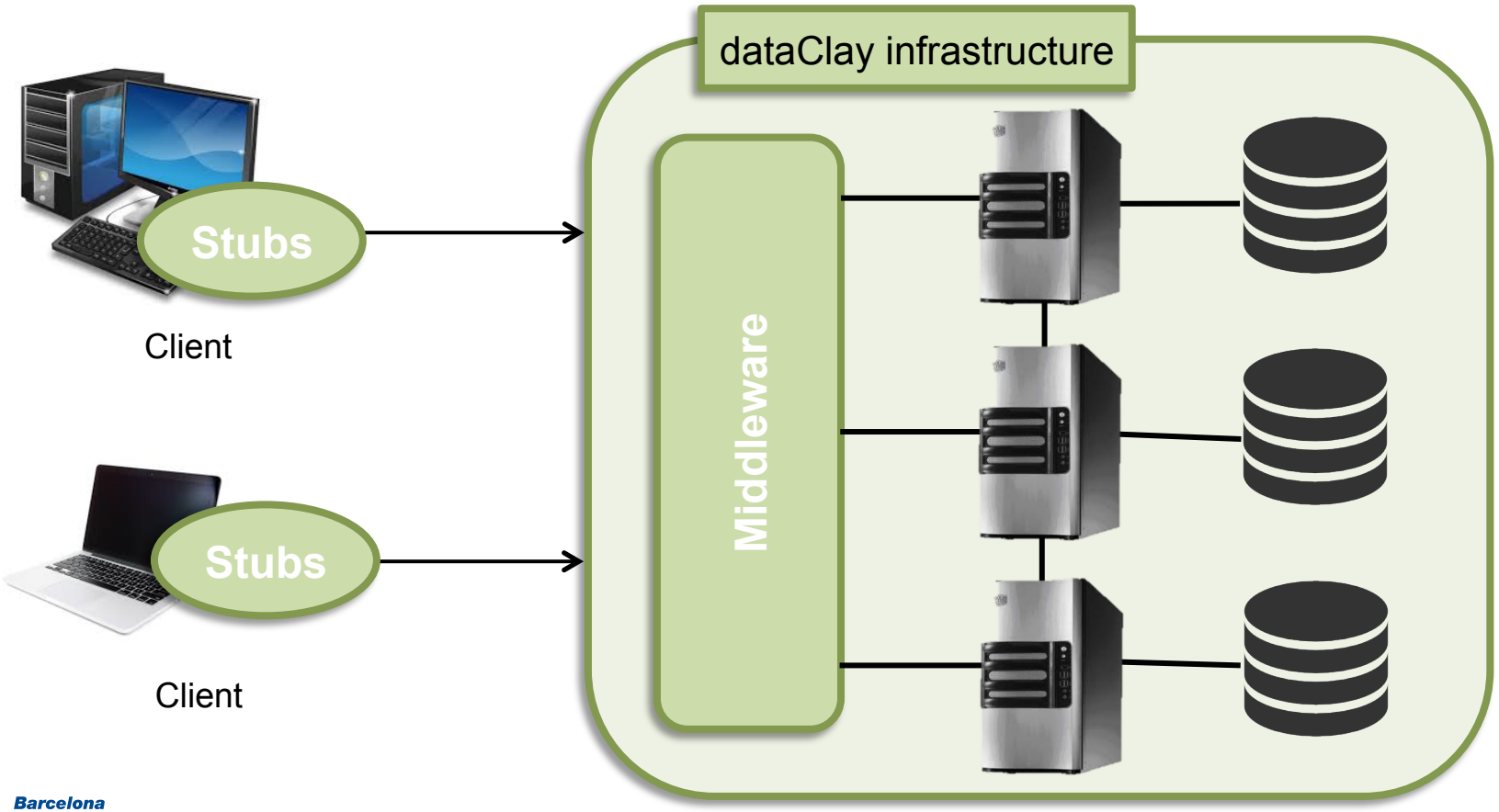


Agenda

- Motivation and background
- Vision
- **Technological details**
- Conclusions

Platform overview

- **dataClay**: Storage platform based on objects
  - Self-contained persistent objects (data and code)
  - Currently a prototype for Java applications

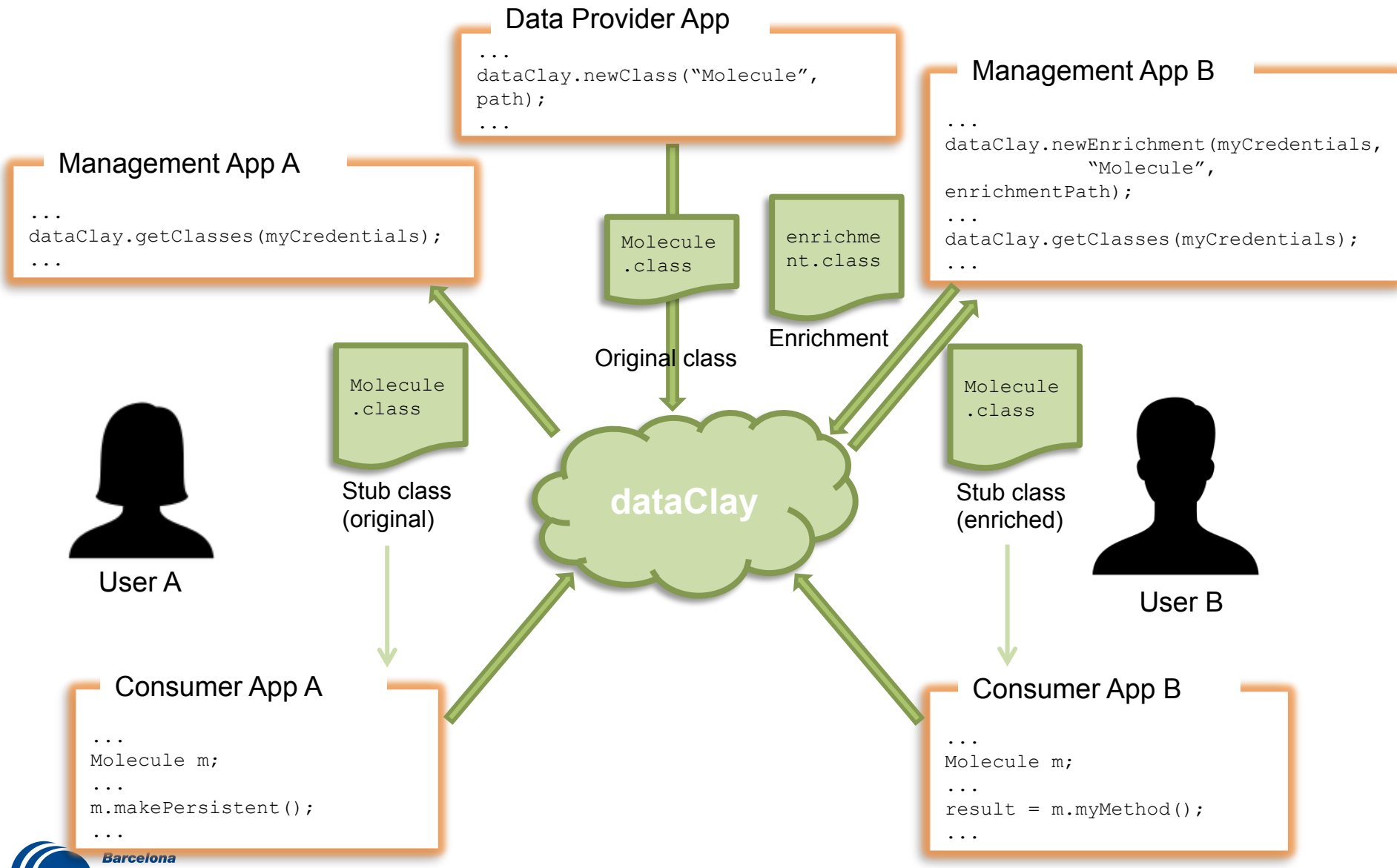


dataClay Interface

- Data API
  - Make object persistent
  - Delete persistent object
  - Retrieve objects
    - By oid
    - By query (simple)
  - "Execute method"

- Management API
  - New class
  - New enrichment
  - Get classes (stubs)

Using dataClay



Stubs

Original Molecule class

```
public class Molecule {
            …
            public void setNext(Molecule nextMolecule) {
                        this.next = nextMolecule;
            }
            public Molecule getNext() {
                        return this.next();
            }
            public Point getCenterOfMass() {
                        return this.centerOfMass();
            }
}
```

Molecule stub class

```
public class Molecule {
            public Molecule(ObjectID oid) {
                        this = (Molecule) dataClay.getObject(oid);
            }
            public ObjectID makePersistent() {
                        return dataClay.makePersistent(this);
            }
            public Molecule[] getAlike() {
                        return dataClay.queryByExample(this);
            }
            //original Molecule methods
            public void setNext(Molecule nextMolecule) {
                        dataClay.executeRemoteImpl(…);
            }
            …
}
```

Transparent persistence

```
//We create the atoms of a molecule
Atom[] atoms = new Atom[2];
atoms[0] = new Atom("H",0,1,0,1);
atoms[1] = new Atom("O",0,0,1,1);

Molecule newMolecule = new Molecule("NewWater", atoms);
objectID = newMolecule.makePersistent();


Molecule lastMolecule = new Molecule(previousID);

//Add the new molecule as a member of the list
lastMolecule.setNext(newMolecule);
```

New object and subobjects are stored

Get persistent object by OID

Updated object is implicitly stored

Remote execution

```
// Query for molecules
Molecule sampleMolecule = new Molecule();
sampleMolecule.setName("Water”);
Molecule[] molecules = sampleMolecule.getAlike();

Molecule currentMolecule = molecules[0];
while (currentMolecule != null) {

      //Calculate center of mass of molecule i
      Point centerOfMass = currentMolecule.getCenterOfMass();
      sumX += centerOfMass.getX() * centerOfMass.getMass();
      sumY += centerOfMass.getY() * centerOfMass.getMass();
      sumZ += centerOfMass.getZ() * centerOfMass.getMass();
      sumMassOfMols += centerOfMass.getMass();

      System.out.println("Getting next molecule...");
      currentMolecule = currentMolecule.getNext();

}
centerX = sumX / sumMassOfMols;
centerY = sumY / sumMassOfMols ;
centerZ = sumZ / sumMassOfMols;
```

Get persistent object by query

All methods of a persistent class are executed remotely

Get persistent object through a method

Enrichment

```
// Query for molecules
Molecule sampleMolecule = new Molecule();
sampleMolecule.setName("Water”);
Molecule[] molecules = sampleMolecule.getAlike();

Molecule currentMolecule = molecule[0];
while (currentMolecule != null) {

      //Calculate center of mass of molecule i
      Point centroid = currentMolecule.getCentroid();
      sumX += centroid.getX();
      sumY += centroid.getY();
      sumZ += centroid.getZ();
      numMols++;

      System.out.println("Getting next molecule...");
      currentMolecule = currentMolecule.getNext();

}
centerX = sumX / numMols;
centerY = sumY / numMols;
centerZ = sumZ / numMols;
```

Class enriched by user U

```
public class Molecule {
            …
            public Point getCentroid(){
                        …
            }
}
```

The new method is available (only for U)

Index

- Motivation and background
- Vision
- Technological details
- **Conclusions**

New business models to share data

- Today
  - In order to exploit data you need
    - To have the data
    - And the infrastructure to store and compute it
      - Or subcontract it, but may lose control over the data
- Our proposal enables to decouple
  - The data provider
  - The infrastructure provider
  - And the different service providers

dataClay

- Storage platform that provides flexible big data sharing
- Today
  - Basic enrichment functionality
  - Reasonable performance
    - Similar to state of the art OODB
- Near future
  - Higher performance
  - Security
  - Scalability

Thanks to …

- Original team
  - **Anna Queralt**
  - Jonathan Martí
  - Daniel Gasull



- Recently joined
  - Juanjo Costa
  - Alex Barceló


- Former team members
  - Ernest Artiaga