www.bsc.es

Barcelona Supercomputing Center
Centro Nacional de Supercomputación

Data sharing in the Big Data era

Anna Queralt and Toni Cortes
Storage System Research Group

Introduction

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

Why sharing data is important?

- Cooperation is the way to success

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

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

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
  - Flexible

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

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 objects** and **enrichment** by 3rd parties

Key technology: self-contained objects

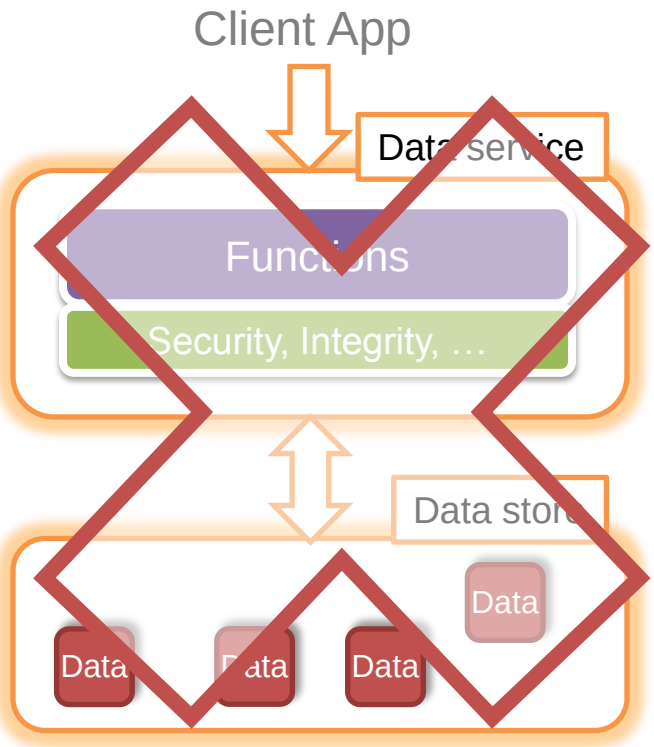
- **Self-contained objects**
  - Data
  - Code
  - Behavior policies

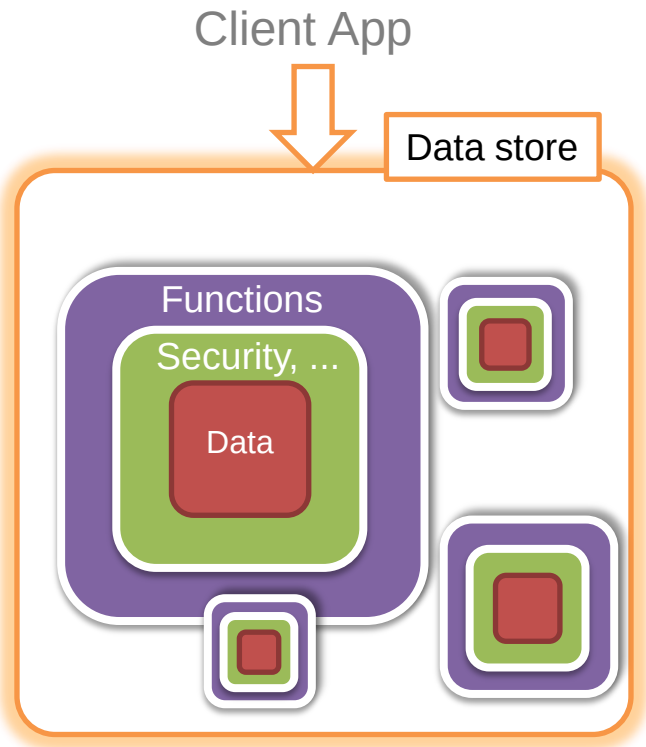
- But …

… this looks much like a data service

Self-contained objects

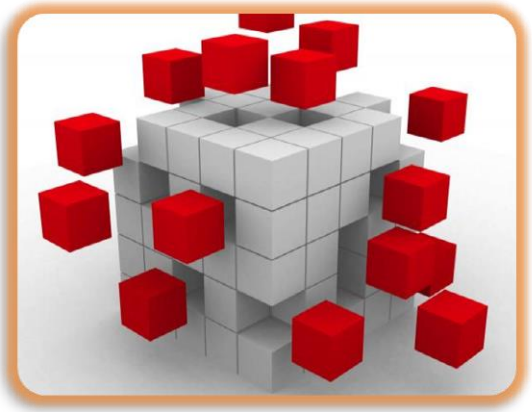
- Push the idea of data services to the limit





3rd-party enrichment

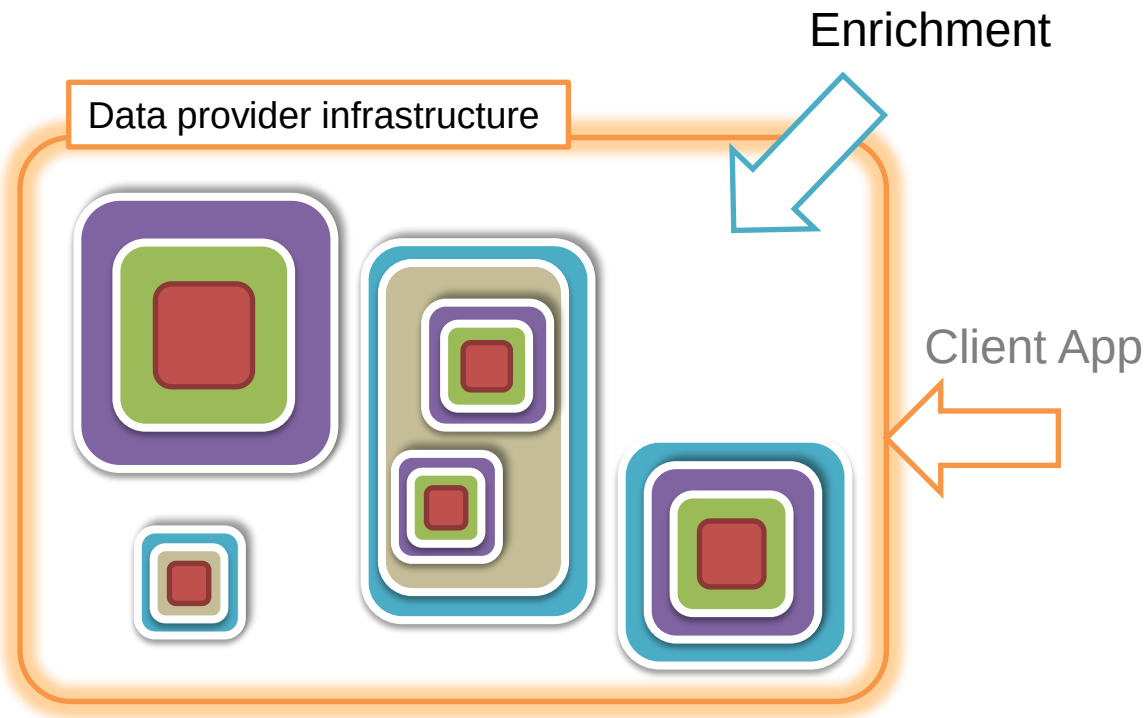
- By enrichment we understand:
  - Adding new information to existing datasets
  - Adding new code to existing datasets



- 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
  - 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

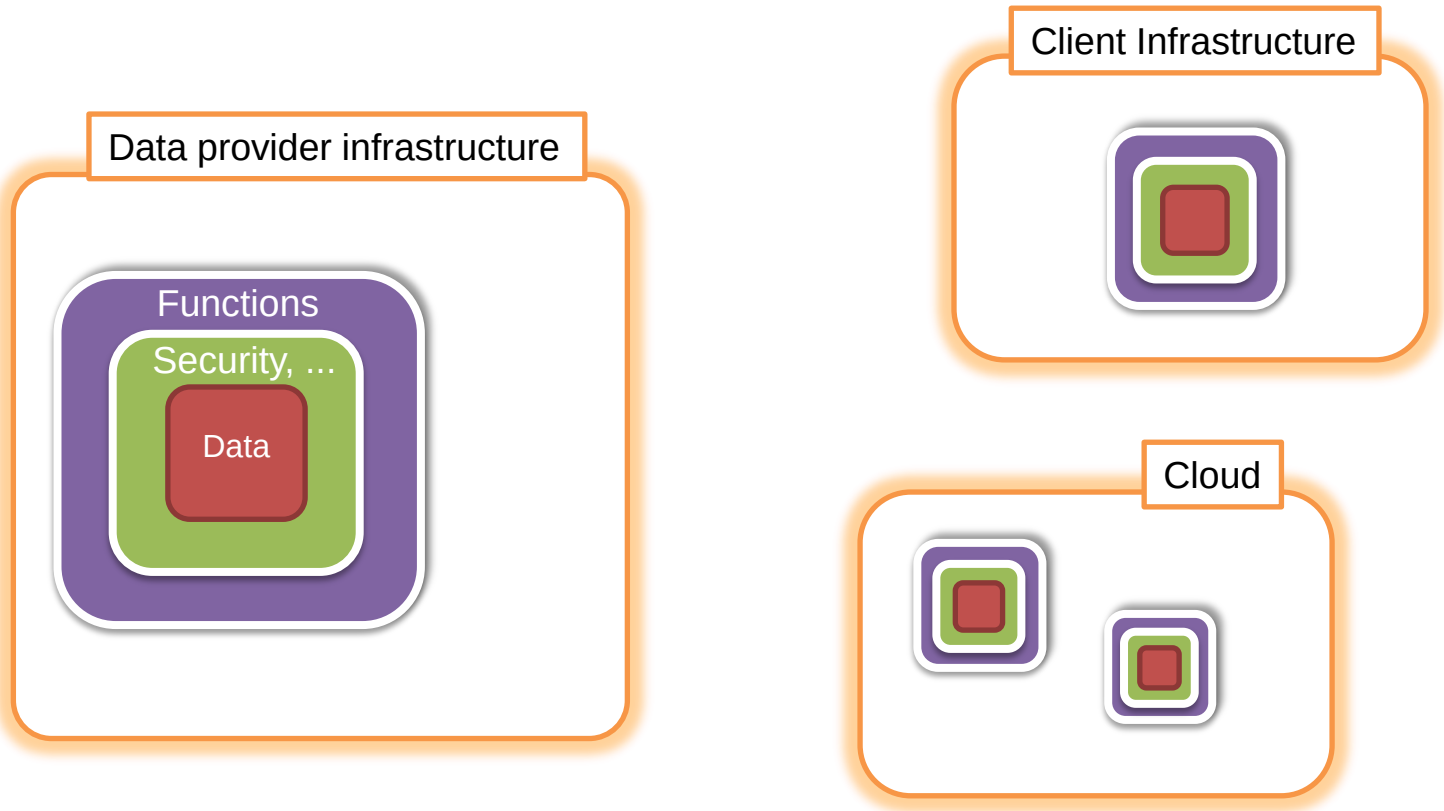
- Security and privacy policies should be part of the data
- Thus, data could be offloaded to other infrastructures

  Without breaking the data policies

Then…

- Efficient usage of resources

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

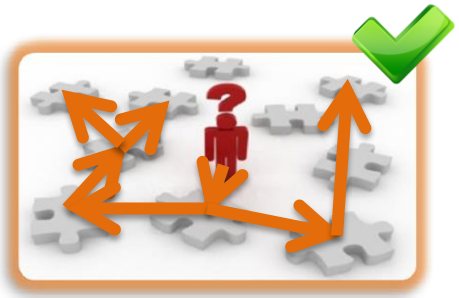


“NEW” PROGRAMMING MODEL

Data selection

- The platform 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

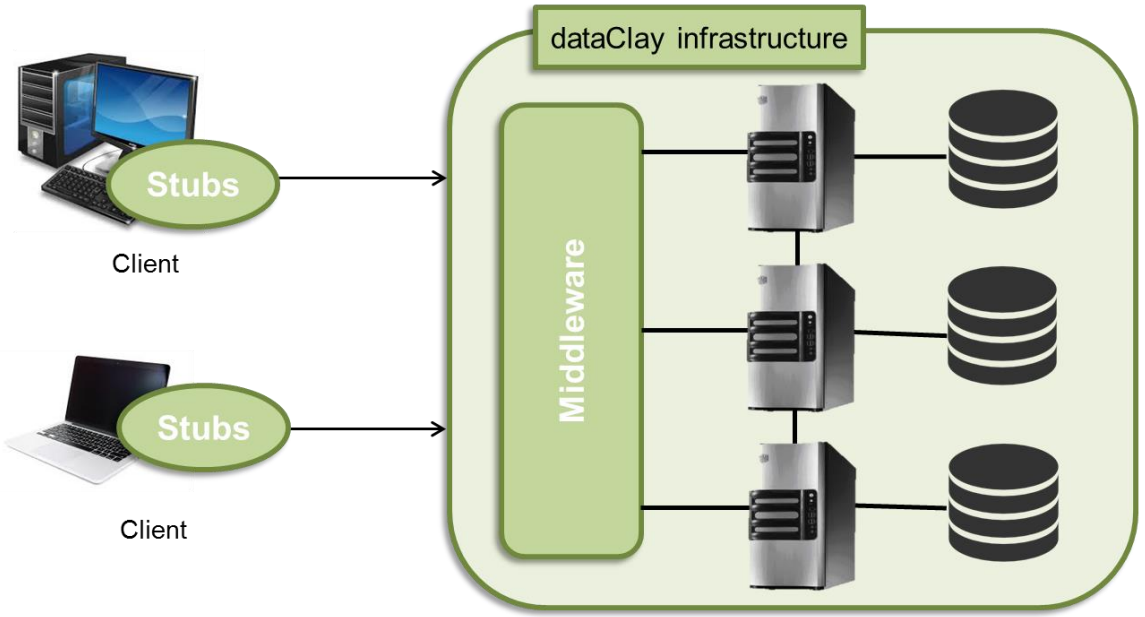
- Persistent data should behave in a similar way
  - Following a link is faster than a query over a dataset
  - Programs do not need to make any differences between persistent and volatile data
  - Enrichments enable data to be linked in different ways



THE PLATFORM

dataClay overview

- dataClay: Storage platform based on objects
  - Currently a prototype for Java applications (and Python soon)
- Main features in the current version:
  - Transparent persistence
    - Store objects directly as seen by applications → no transformations
  - Remote execution of methods
    - Execute methods in the resource where data is stored
  - Enrichment of existing classes
    - With new methods
    - With new fields



PRELIMINARY MEASURES

Experiment

- Goal
  - See how dataClay performs compared to other data management systems that are used today
    - Use this information to optimize performance (dataClay is still under development)

- We have chosen a popular representative for each of the following kinds of data stores:
  - Key/value: Cassandra
  - Object-oriented database: db4o
  - Graph database: Neo4j
  - RDBMS: PostgreSQL

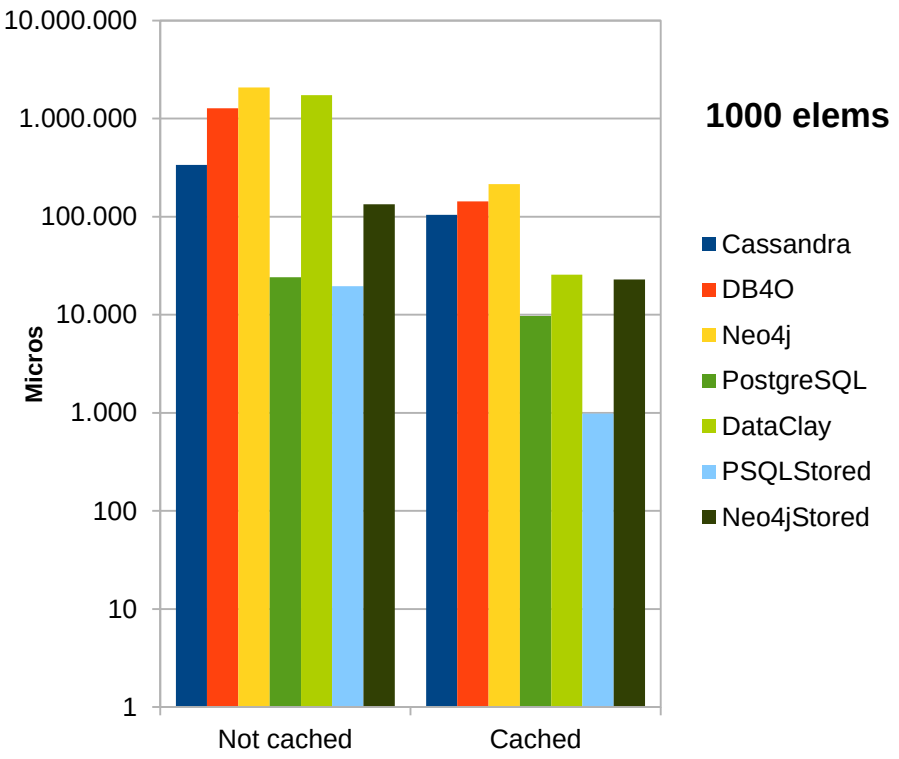
Application

- Find the maximum value in a list of 1000 elements
- 2 alternative settings for each element in the list:
  I. A single integer
  II. An array of 1000 integers, the average of which has to be calculated

- Implementation on top of
  - Cassandra and PostgreSQL:
    - A single table containing all the elements in the list
    - All the elements are retrieved at once by means of a SELECT *
  - Neo4j:
    - Each element is represented by a node with an edge to the next node
    - All the elements are retrieved at once by means of a SELECT *
  - dataClay and db4o:
    - Each object in the list has a reference to the next object
    - Objects are *accessed* one by one
  - PostgreSQL using stored procedures
  - Neo4j using server plugins

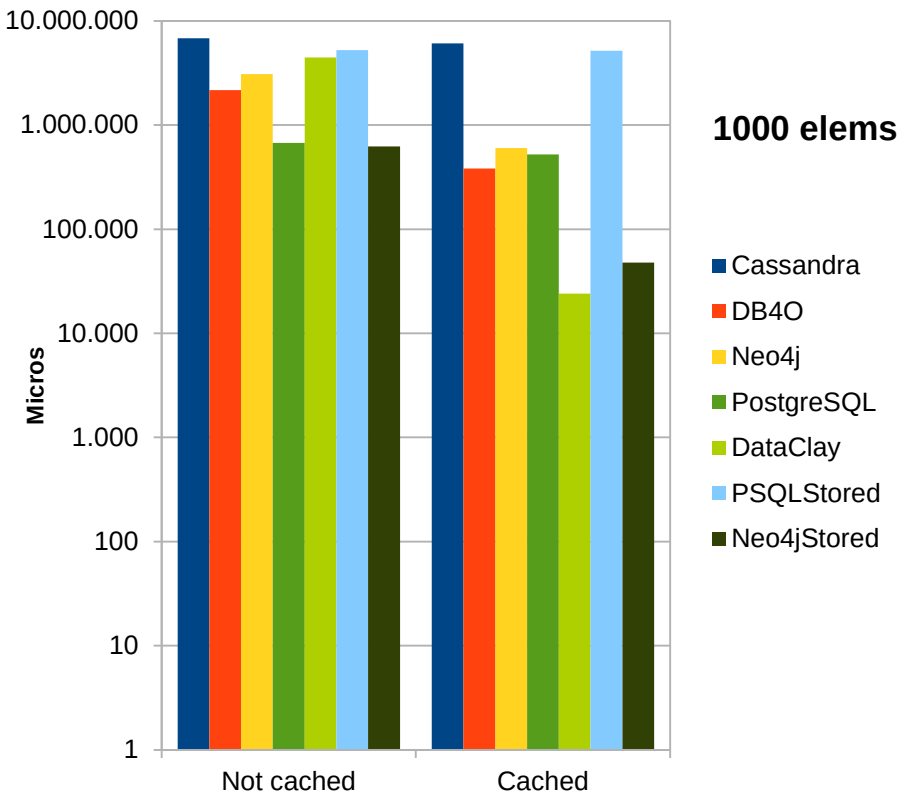
Results (I)

- Each element contains a single integer
  - PostgreSQL is much faster than the rest, especially with stored procedures
  - Db4o, Neo4j and dataClay do not perform well when each element is accessed once, but when elements are cached dataClay is much close to stored procedures than the rest



Results (II)

- Each element contains an array of 1000 integers, its average is calculated before calculating the maximum
  - PostgreSQL does not behave so well with arrays, either with or without caching, and with or without stored procedures
  - dataClay with cached objects outperforms the rest of solutions up to 2 orders of magnitude



CONCLUSIONS

dataClay

- Storage platform that provides flexible big data sharing
- Today
  - Store and retrieve objects
  - Execution of methods in the platform
  - Basic enrichment functionality
  - Reasonable performance
- Near future
  - Higher performance and scalability
  - Fault-tolerance
  - Security

Benchmarking

- Performance is essential in a big data platform
  - … But how can we choose between two different solutions with similar features and performance?

- Is it possible to measure other thinks like…?
  - Usability
    - e.g, easiness of building a new application on top
  - Flexibility
    - e.g., easiness of using the same data in different ways without affecting performance
  - Energy efficiency
    - e.g., energy consumed in the execution of an application


BSC
Barcelona
Supercomputing
Center
Centro Nacional de Supercomputación
THANK YOU