Comparative evaluation of tools providing access to different types of data resources exposed on the Grid

A joint test program of INFN, SPACI-UNILE and INAF

Giacinto Donvito

INFN-Bari

Outline

- Motivations
- The tools under evaluation
- The testbed
- The test plan
- Issues
- Some Results
- Example of use
- Conclusions

Motivations

- Provide access in GRID to Bioinformatics data stored in relational DB's
  - BioinfoGRID EU project (http://www.bioinfogrid.eu/)
  - LIBI Italian MIUR project (http://www.libi.it/libi)
- Provide access to multiple astronomical databases (archives, Astronomical Catalogues, etc.) for the Astrophysical community.
  - The Virtual Observatory (http://www.euro-vo.org/pub/ http://www.ivoa.net/)
- Provide access to population data for porting "public administrations" applications to the GRID
  - EGG Italian MIUR project
- Diffuse inside the Italian community the knowledge and the expertise to access relational DB's from Grid.

The tools under evaluation

- GRelC: Grid Relational Catalog
  - *Developed by SPACI Consortium and University of Salento, Lecce*
    - Giovanni Aloisio (giovanni.aloisio@unile.it) , Supervisior
    - Sandro Fiore (sandro.fiore@unile.it), Project P.I.
    - Massimo Cafaro (massimo.cafaro@unile.it), Team Member
    - Alessandro Negro (alssendro.negro@unile.it) , Team Member
    - Salvatore Vadacca(salvatore.vadacca@unile.it), Team Member
  - Project site: *http://grelc.unile.it*

- G-DSE (INAF + INFN)
  - *Developed by INAF and INFN*
    - Edgardo Ambrosi (amborsi@cnaf.infn.it)
    - Giuliano Taffoni (taffoni@oats.inaf.it)
    - Andrea Barisani (lcars@infis.units.it)
  - Project site: *http://wwwas.oats.inaf.it/grid/G-DSE*

- OGSA-DAI
  - *Developed as part of the Open Middleware Infrastructure Institute UK (OMII-UK) project.*
  - Project site: http://www.ogsadai.org.uk/index.php

- AMGA
  - *Developed as part of gLite by:*
    - Birger Koblitz: Initial design, project responsible.
    - Tony Calanducci: RPM building and testing. User support.
    - Salvatore Scifo: Java API maintainer
    - Claudio Cherubino: PHP client
  - Project site: http://amga.web.cern.ch/amga/

The GRelC Project

- ***Grid Relational Catalogue*** **is a project which aims at designing and developing a set of efficient, secure and transparent Data Grid Services (Starting date, January 2001).**

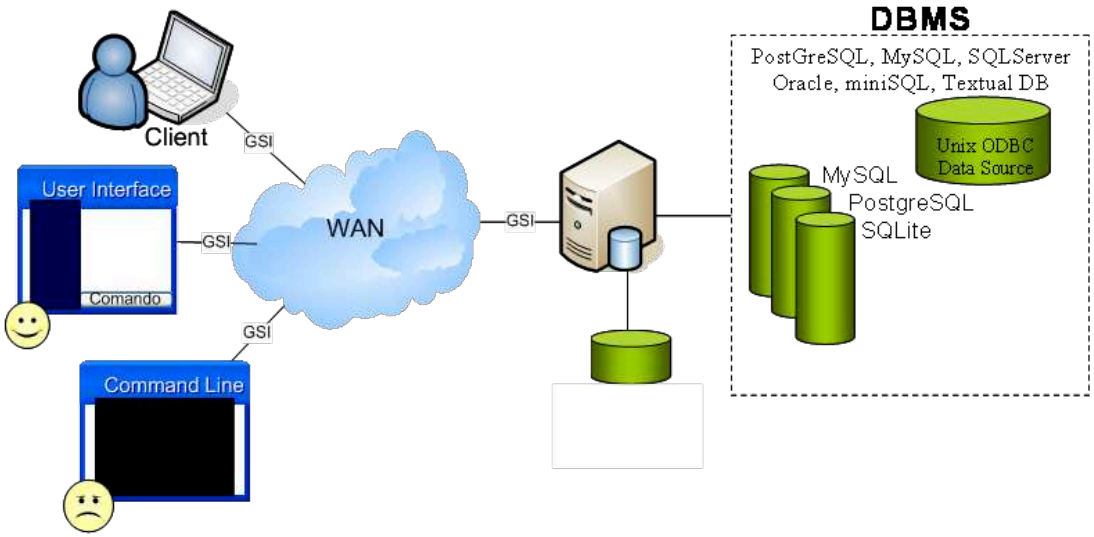
- ***Developed by SPACI Consortium and University of Salento, Lecce***

- ***GRelC Data Access Service*** **aims at providing a large set of functionalities to access both relational and non relational Databases in a grid environment**

http://indico.cern.ch/contributionDisplay.py?contribId=334&sessionId=28&confId=18714

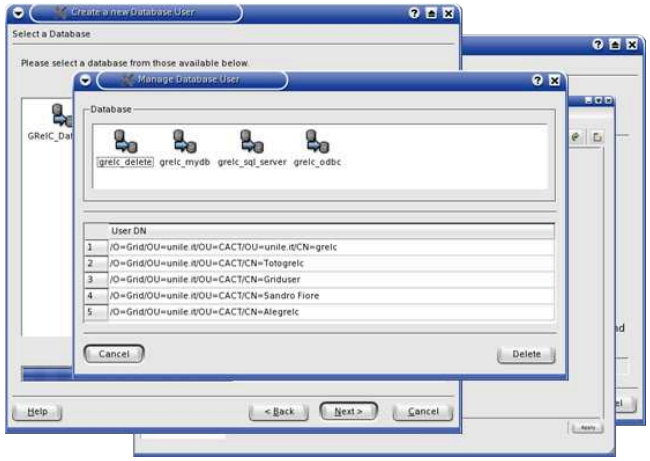
GRelC Data Access Service

- GRelC DAS is a Grid Database Access Service
- Web Service interface GSI enabled and WS-I compliant
- Mutual authentication based on GSI (X.509v3 digital certificates)
- Authorization Framework leveraging local ACL and VOMS



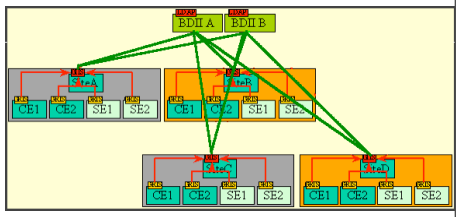
GReIC DAS: Main Features

- **Protection against SQL Query Injection**
- **Single Query: prefetch, dime, memory, stream etc.**
- **Dinamic binding to heterogeneous DBMSs:**
  - Postgres, MySQL, SQLite
  - DB2, Oracle
  - Microsoft SQL Server
  - UnixODBC



- **Graphical User Interface (Qt based)**
- **Web Interface (For Admin tasks and User Query)**

GRelC DAS and gLite



- Information System Support (BDII compliant)
- Porting on LCG-2-4-0, LCG-2-7-0 and gLite 3.x
- It also runs on the following platforms:
  - **Linux**
  - **MAC OS X**
  - **FreeBSD**
  - **Both IA64 and IA32 platforms are supported**

- Wide deployment on GILDA t-Infrastructure
  - **A tutorial is also available on Grid CT Wiki (GILDA,**
  - **https://grid.ct.infn.it/twiki/bin/view/GILDA/GRelCDataAccessService**

Data Source Engine Concepts

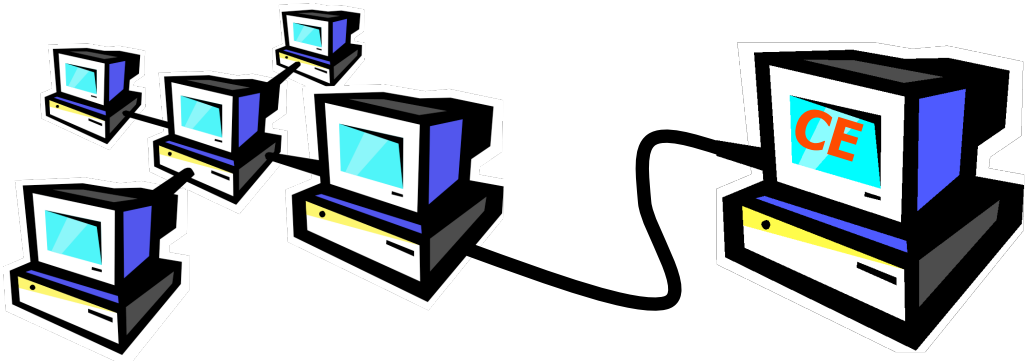
- The Grid Resource Framework Layer, Information System **and** Data Model **is extended so that a software virtual machine as a Data Source Engine becomes a valid instance for a Grid computing model.**

- **A** new Grid component **(G-DSE) that enables the access to a Data Source Engine and Data Source, totally integrated with the Grid Monitoring and Discovery System and Resource Broker is defined**

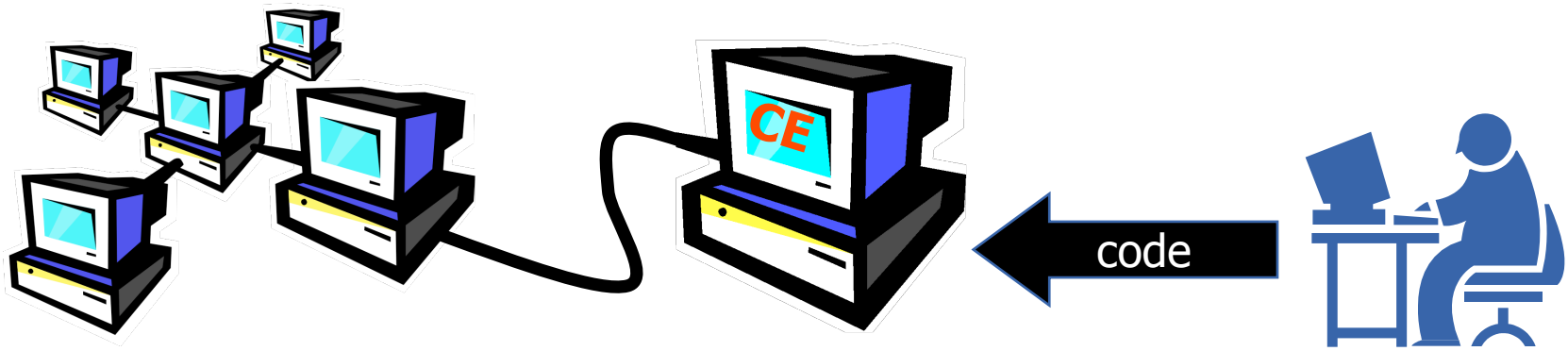
- **A new** Grid Element, the Query Element, **can be built on top of the G-DSE component.**

http://indico.cern.ch/contributionDisplay.py?contribId=333&sessionId=28&confId=18714

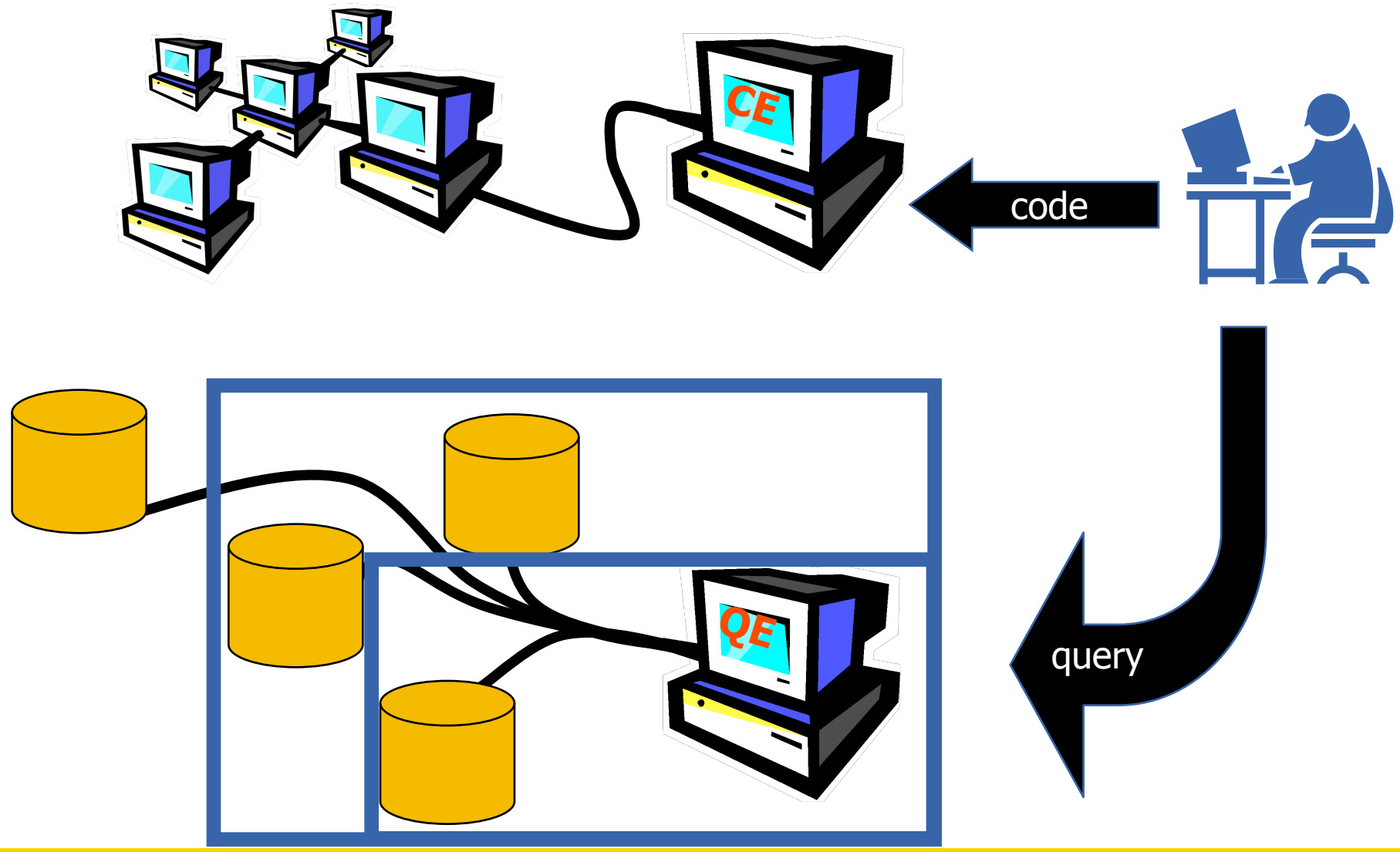
The Query Element



The Query Element



The Query Element



QE implementation

- Runs on any linux/unix flavor: GT>=2.4.3
- Backends: any DB vendor (MySQL, Oracle, PostgreSQL, etc…) + flat files
- Two protocols: GRAM or WS
- Authentication based on GSI
- Authorization based on VOMS
- API: C, C++, python, Java, perl

OGSA-DAI can support the following:

- Different types of data resources - including relational, XML and files can be exposed via web services. A number of popular data resource products are supported.
- Data within each of these types of resource can be queried and updated.
- Data can be transformed (using XSLT), compressed and decompressed (using ZIP and GZIP compression).
- Data can be delivered to clients, other OGSA-DAI web services, URLs, FTP servers, GridFTP servers, or files.
- Requests to OGSA-DAI web services have a uniform format irrespective of the data resource exposed by the service. (though the actions specified within each request may be data resource-specific).
- Information on the data resources exposed by an OGSA-DAI web service and the functionality supported by the service can be accessed by clients.
- OGSA-DAI users can extend OGSA-DAI web services to expose their own data resources and to support application-specific functionality, in addition to that provided by OGSA-DAI.
- Can execute data-centric workflow and data transformations
- http://indico.cern.ch/contributionDisplay.py?contribId=397&sessionId=28&confId=18714

- AMGA (ARDA Metadata Grid Application):
  - It is designed to provide metadata access to grid application
  - It allows the use of several back-ends:
    - Oracle, MySQL, PostgreSQL, SQLite
  - It is oriented to files (Metadata describes files on the grid)
  - It is possible to use it in order to access DB using a dedicated Languages.
    - This languages allows a lot of functionality provided by SQL but it is not SQL standard
    - Some very complex query are not possible in an easy way
    - Pre-existent application should be modified
  - It provides both Web Service and socket connection
  - It gives the possibility to exploit advanced replication functionality
    - Also between server using different back-end
    - and with “streaming” behavior
  - It has a lot of good facilities for supporting ACL (Unix like) on tables (or entries), based on GSI authentication
  - API available in several languages: C++, Java, Python, Perl, Ruby
  - It is available also with standalone Python library (for single user)

TESTBED

- OGSA-DAI
- G-DSE
- AMGA
- GRELC-DAS

Test Database: Bioinformatics database containing just a "molecule table" with about 500.000 tuples (350MB, PostgreSQL).

Other Databases:

Sakila (MySQL 23 tables)
World (MySQL 6 tables)
Dellstore (PostgreSQL 8 tables)
uniutrdb_test (MySQL 35 tables)
go_5_06 (MySQL 18 tables)
homo_sapiens (MySQL 74 tables)
2MASS (PostgreSQL)
Population db (PostgreSQL)

The test plan

- Sequential tests
  - extraction of zero tuples - Estimate GSI services overhead
  - extraction of 10, 100, 1000, 10000, 100000 simple tuples
  - extraction of result sets of increasing dimensions: O(kb), O(MB), O (100MB)
  - extraction of 10, 100, 1000, 10000, 100000 with increasing table complexity
  - Submission of complex queries (join, multiple queries, etc)
  - Submission of **INSERT, UPDATE,** and **DELETE** queries
    - Performance evaluation for a single action
  - Evaluate the differences between LAN and WAN queries

The test plan

- Concurrent tests
  - Whit O(10) concurrent clients extract
    - **Zero, 10, 100, 1000 tuples**
  - Repeat the extractions with O(100) concurrent clients

- Use a common working environment for the three tools
  - Only GSI authentication
    - VOMS authentication not supported by some of them
  - Same type of output format
    - No date post or pre processing allowed (avoid translation in a more user friendly format (XML,….))

Security

Tools	GSI	VOMS Authentication	Transport Layer Security	Data Encryption
OGSA-DAI	Yes	No	Yes	Yes
GRELC-DAS	Yes	Yes	Yes	Yes
G-DSE	Yes	Yes	Yes	No
AMGA	Yes	Yes	Yes	Yes

Tools installation

- GRelC:
  - Easy to install and manage via the grid data portal interface (GRelC Portal)
  - Postgresql and Mysql drivers currently available from the website.
  - New drivers related to Oracle, DB2, etc. are in a preproduction phase.
- G-DSE:
  - The installation and configuration procedure was not straightforward.
  - Installation based on yaim (glite tool)
  - Found problems in almost all the sites
- OGSA-DAI:
  - The installation and configuration of the tool was not straightforward although a vaste documentation is provided on the web site.
  - Greater variety of data sources accessed
- AMGA:
  - The installation is quite easy and straightforward
  - It is not simple to use more than one DB at the same time

Population DB: Test Result

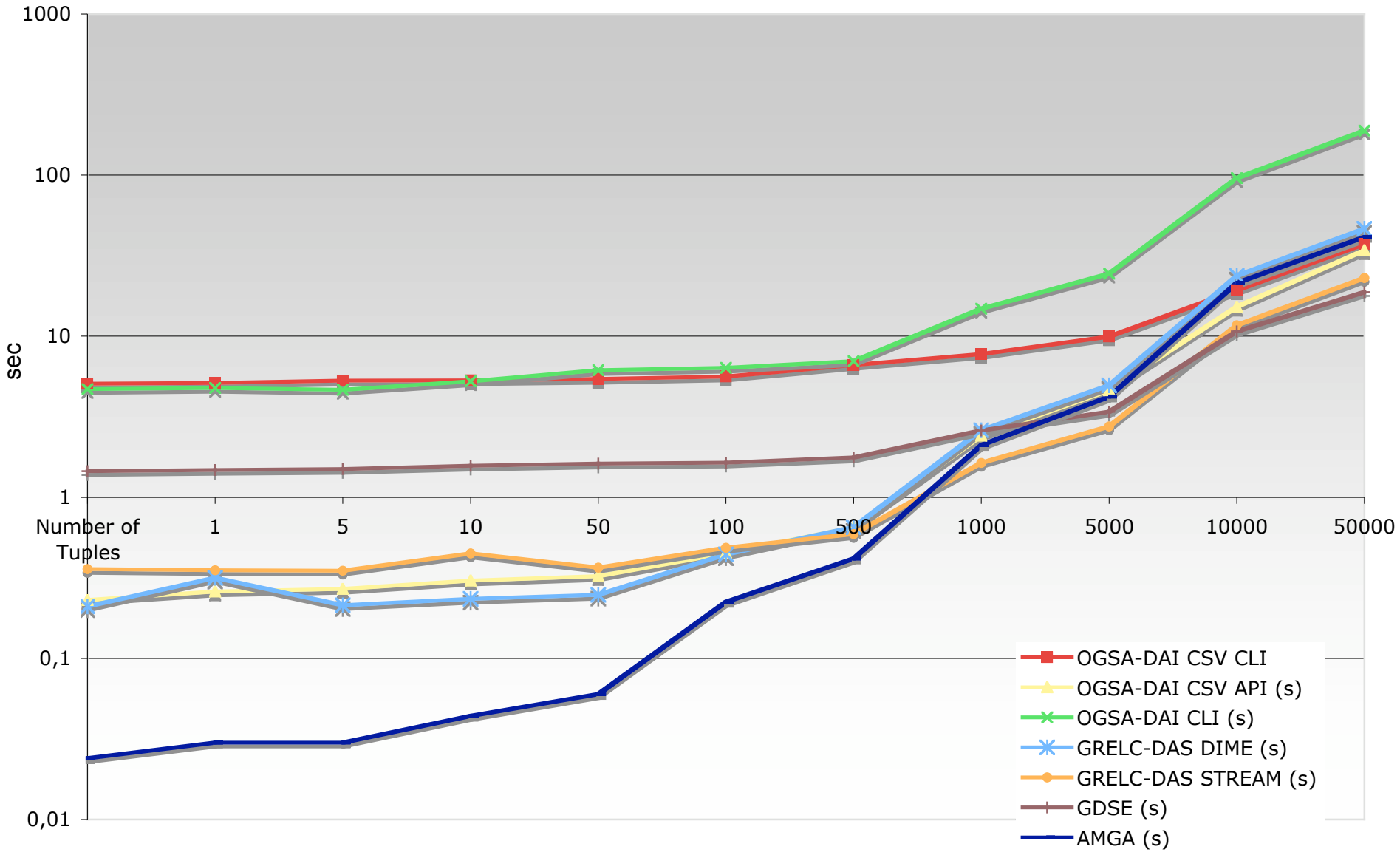
- Tests have been performed on the population database using the real use case query.
- Such query includes complex join query between two or more tables, inner join and outer join between tables.
- It has been found that the database itself is bottleneck in this case.
- No big difference found in the query response time between running query using database client and any of three tools.
- Current version of AMGA does not support such complex join query like inner join or outer join between tables and require a readjust of the DB.

… before numbers…

- For all the results:
  - These numbers are the result of a quite long process of optimization and interaction with the developers of each tools, but ... improvements are still possible.

- AMGA does not still support standard SQL query

Simple query: tests results



Simple query: tests results (2)

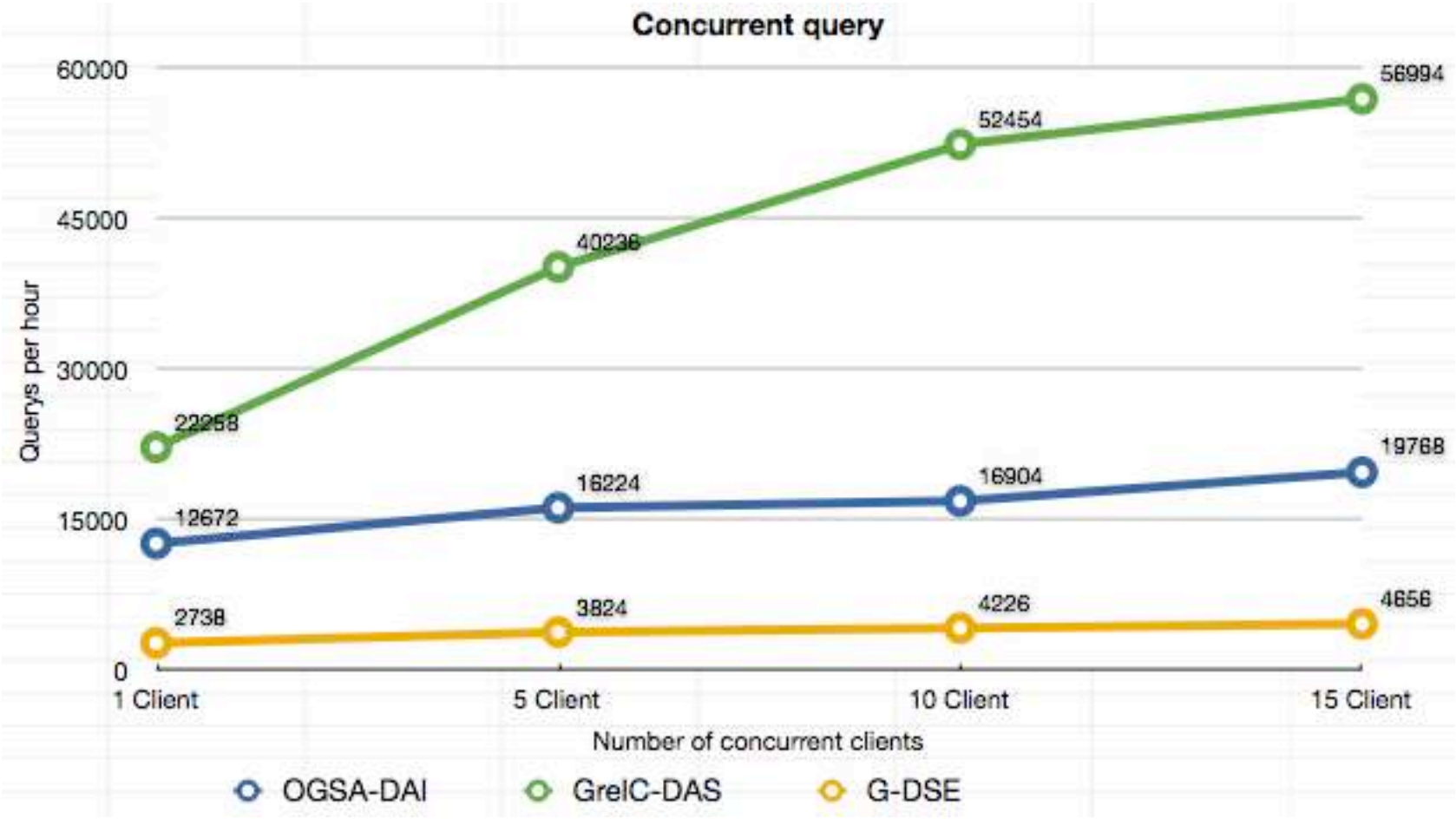
Number of Tuples	OGSA-DAI CSV CLI (s)	OGSA-DAI CSV API (s)	OGSA-DAI CLI (s)	GRELC-DAS DIME (s)	GRELC-DAS STREAM (s)	GDSE (s)	AMGA (s)
1	5,07	0,23	4,7	0,21	0,358	1,454	0,024
5	5,13	0,26	4,77	0,316	0,352	1,48	0,03
10	5,31	0,27	4,65	0,214	0,35	1,496	0,03
50	5,32	0,304	5,25	0,234	0,448	1,572	0,044
100	5,43	0,324	6,15	0,248	0,366	1,62	0,06
500	5,61	0,45	6,37	0,442	0,486	1,642	0,224
1000	6,63	0,65	7	0,652	0,592	1,77	0,416
5000	7,74	2,41	14,8	2,606	1,634	2,602	2,106
10000	9,96	4,61	24,46	4,962	2,754	3,398	4,208
50000	19,31	15,21	95,63	23,686	11,696	10,71	21,454
100000	37,42	34,21	188,86	46,486	23,002	18,764	41,336

- OGSA-DAI CLI is much slower than all other due to the time needed to start the JVM:
  - For this reason we report also the time spent for a query using the API:
    - We measure only the time needed to run just the query.

Concurrent test: description

- We use a fixed number of clients that run constantly the same query against the server.
- We measure the number of queries that each service is able to provide for a fixed time-window (30 min).
- This gives an idea of the performance that each software is able to reach under high load
- We strictly keep under control the load on both client and server side

Concurrent test: number of query

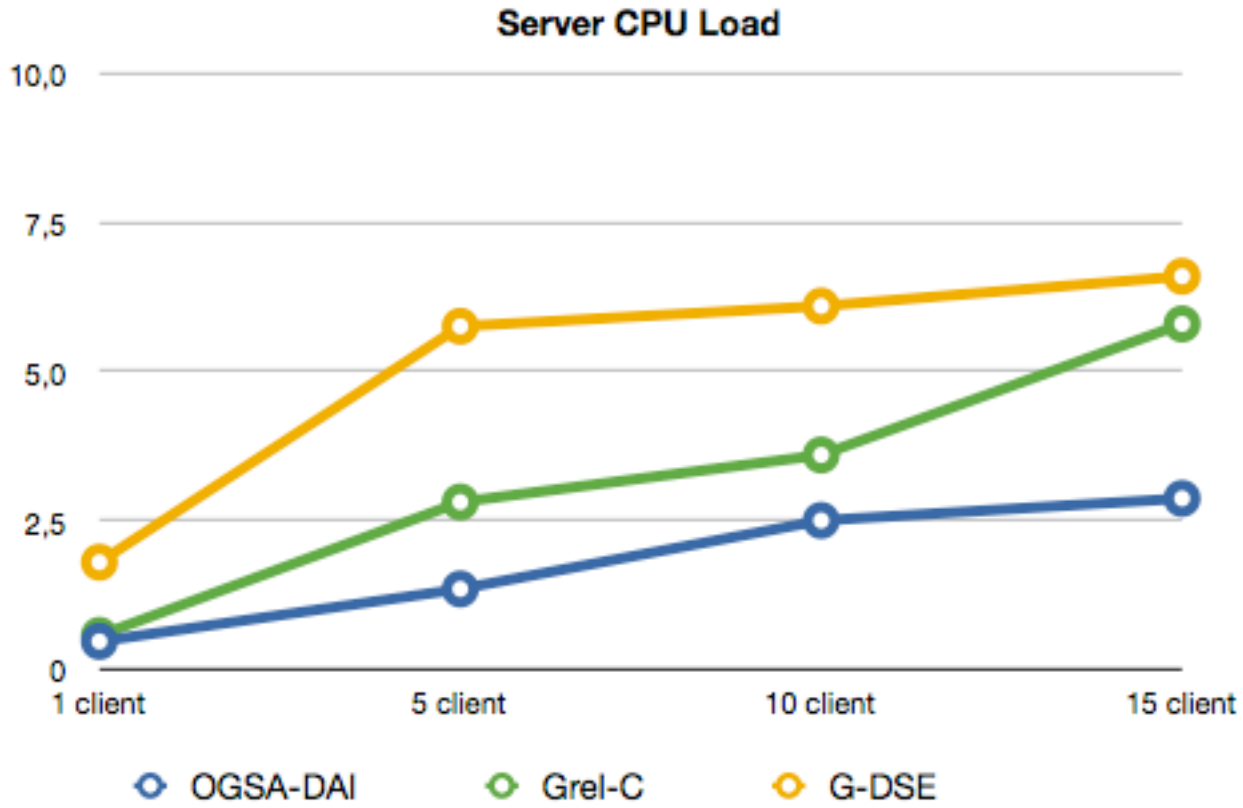


- In this graph are showed the amount of query server by each tools per hour, related to the given amount of concurrent client

Concurrent test: issues

- OGSA-DAI clients are loading much more the client machine instead of loading the server
- For each execution 4 seconds are spent to start the JVM and only 0.2 seconds in order to execute the query.
- We solved the problem building a program that runs an infinite loop instead of running many times the CLI
- In this way the client is not overloaded and a good scalability is achieved

Concurrent test: server load



- This graph shows the load average of each server when the concurrent test is executed:
  - It seems evident that for OGSA-DAI the number of client needed to saturate the machine is not reached yet… maybe we can achieve a better results increasing the number of concurrent client

BioinfoGRID

Examples:

- GDSE:
  - CLI interface:

```
>  globus-job-run  grpk005.oat.ts.astro.it:2119/jobmanager-odbc -queue dellstore2 "select products.title,categories.categoryname, products.actor from products,categories where products.category = categories.category and categories.categoryname='Sci-Fi';" -XML
```

Examples:

- GDSE:
  - CLI interface:

```
>  globus-job-run  grpk005.oat.ts.astro.it:2119/jobmanager-odbc -queue dellstore2 "select products.title,categories.categoryname, products.actor from products,categories where products.category = categories.category and categories.categoryname='Sci-Fi';" -XML
<table>
<row>
<title>ACADEMY ACADEMY</title>
<categoryname>Sci-Fi</categoryname>
<actor>PENELOPE GUINESS</actor>
</row>
<row>
<title>ACADEMY ACE</title>
<categoryname>Documentary</categoryname>
<actor>EWAN RICKMAN</actor>
</row>
….
```

Examples:

- GDSE:
  - CLI interface (Parallel Query):

BioinfoGRID

Examples:

- GDSE:
  - CLI interface (Parallel Query):

```
> globus-job-submit -: g.dse.host/dbmanager-ODBC -queue PSQL
“select a,b from table where a < 10;” -: g.dse2.host/dbmanager-
ODBC -queue PSQL “select a,b from table where a between 10 and
20;” -: g.dse.host3/dbmanager-ODBC -queue PSQL “select a,b from
table where a > 20;”
```

Examples:

- GreIC:
  - CLI interface:

BioinfoGRID

Examples:

- GreIC:
  - CLI interface:

```
> /opt/grelc/das/bin/grelc-das-query-stream  -s
firblibi03.ba.infn.it -D MOLECULE -Q  "select *  from molecule
where id<=2" -t
**********************************************
id: 1
id_name: 104K_THEPA
id_molecule_type: 1
id_data_class: 1
id_sequence_type: 1
seq: MKFLILLFNI LCLFPVLAAD NHGVGPQGAS GVDPITFDIN SNQTGPAFLT AVEMAGVKYL QVQHGSNVNI HRLVEGNVVI WENASTPLYT GAIVTNNDGP YMAYVEVLGD
PNLQFFIKSG DAWVTLSEHE YLAKLQEIRQ AVHIESVFSL NMAFQLENNK YEVETHAKNG ANMVTFIPRN GHICKMVYHK NVRIYKATGN DTVTSVVGFF RGLRLLLINV
FSIDDNGMMS NRYFQHVDDK YVPISQKNYE TGIVKLKDYK HAYHPVDLDI KDIDYTMFHL ADATYHEPCF KIIPNTGFCI TKLFDGDQVL YESFNPLIHC INEVHIYDRN
NGSIICLHLN YSPPSYKAYL VLKDTGWEAT THPLLEEKIE ELQDQRACEL DVNFISDKDL YVAALTNADL NYTMVTPRPH RDVIRVSDGS EVLWYYEGLD NFLVCAWIYV
SDGVASLVHL RIKDRIPANN DIYVLKGDLY WTRITKIQFT QEIKRLVKKS KKKLAPITEE DSDKHDEPPE GPGASGLPPK APGDKEGSEG HKGPSKGSDS SKEGKKPGSG
KKPGPAREHK PSKIPTLSKK PSGPKDPKHP RDPKEPRKSK SPRTASPTRR PSPKLPQLSK LPKSTSPRSP PPPTRPSSPE RPEGTKIIKT SKPPSPKPPF DPSFKEKFYD
DYSKAASRSK ETKTTVVLDE SFESILKETL PETPGTPFTT PRPVPPKRPR TPESPFEPPK DPDSPSTSPS EFFTPPESKR TRFHETPADT PLPDVTAELF KEPDVTAETK
SPDEAMKRPR SPSEYEDTSP GDYPSLPMKR HRLERLRLTT TEMETDPGRM AKDASGKPVK LKRSKSFDDL TTVELAPEPK ASRIVVDDEG TEADDEETHP PEERQKTEVR
RRRPPKKPSK SPRPSKPKKP KKPDSAYIPS ILAILVVSLI VGIL
seq_check: 289B4B554A61870E CRC64
mw: 103626
length1: 924
createddate: 1990-04-01
lastupdateddate: 1990-04-01
lastannoteddate: 1992-08-01
descr: 104 kDa microneme-rhoptry antigen
id_database_division:
id_note:
id_db: 1

**********************************************
```

Examples:

- AMGA:
  - CLI interface:

BioinfoGRID

Examples:

- AMGA:
  - CLI interface:

```
> mdcli "selectattr /molecule:id  /molecule:id_name  /
molecule:id_molecule_type  /molecule:id_data_class  /
molecule:id_sequence_type  /molecule:seq  /molecule:seq_check /
molecule:mw    /molecule:length1  /molecule:createddate  /
molecule:lastupdateddate  /molecule:lastannoteddate  /
molecule:descr  /molecule:id_database_division  /
molecule:id_note  /molecule:id_db '/molecule:id<$1 and /
molecule:id>0'"
```

Examples:

- OGSA-DAI:
  - API in a simple application:

Examples:

Examples:

```
> import java.sql.ResultSet;
…
import uk.org.ogsadai.client.toolkit.Response;
import uk.org.ogsadai.client.toolkit.activity.sql.SQLQuery;
…
 String sql = "select * from molecule where id<=100";
       SQLQuery query = new SQLQuery(sql);
       WebRowSet rowset = new WebRowSet(query.getOutput());
…
 // Set the number of rows to fetch each time to 200
        ResultSet result = deliver.getResultSet(200);
       result.next();
       int rowNumber = 1;
       while (result.next()) {
       System.out.println("id: " + result.getString(1));
       System.out.println("id_name: " + result.getString(2));
       System.out.println("id_molecule_type: " + result.getString(3));
       System.out.println("id_data_class: " + result.getString(4));
       System.out.println("id_sequence_type: " + result.getString(5));
       System.out.println("seq: " + result.getString(6));
       System.out.println("seq_check: " + result.getString(7));
       System.out.println("mw: " + result.getString(8));
       System.out.println("length: " + result.getString(9));
…
```

Conclusions

- The test is still on going
  - Some input from OGSA-DAI developers has improved the reliability of the results
- From the first results it seems that each middleware has some specific field of excellence:
  - OGSA-DAI:
    - has lot of advanced features
    - Widely distributed
    - Fair good performances when used with API
  - GRelC:
    - Fast for small/medium/large datasets.
    - Fast DML query submission.
    - Easy to install. Already ported on gLite
    - Easy to manage via the grid data portal interface (GRelC Portal)

Conclusions (2)

- …
  - GDSE
    - Very efficient for huge query
    - Easy to manage
    - Good integration with gLite environment (BDII and VOMS enabled)
  - AMGA:
    - Very quick for small query
    - Fast DML query submission (very low GSI latency).
    - A lot of advanced functionality (replication master-multi slave, etc)
    - Good performance for large datasets
    - It will be part of the next release of gLite
- The “best tools” should be chosen taking into account the use case:
  - Java Application/Workflow -> OGSA-DAI
  - Not strict SQL bound application -> AMGA
  - General purpose -> GreIC
  - Huge query -> G-DSE
  - …
- If a standard unified interface is build: the user can use the same API/ interface to use different back-end in a transparent way (Like SRM for data access).

Acknowledgments

- TAFFONI, Giuliano (INAF)
- VUERLI, Claudio (INAF)
- BARISANI, andrea (INAF)
- PASIAN, Fabio (INAF)
- MANNA, Valeria (INAF)
- GISEL, Andreas (CNR-ITB)
- GIORGIO, Emidio (INFN)
- AIFTIMIEI, Cristina (INFN)
- ATUL, Jain (INFN+Politecnico Bari)
- BARBERA, Roberto (INFN+Università Catania)
- CAROTA, Luciana (INFN)
- DONVITO, Giacinto (INFN)
- GHISELLI, Antonia (INFN)
- LA ROCCA, Giuseppe (INFN)
- MAZZUCATO, Mirco (INFN)
- PIERRO, Antonio (INFN)
- VERLATO, Marco (INFN)
- FIORE, Sandro (Univ. del Salento, Lecce)
- ALOISIO, Giovanni (Univ. del Salento, Lecce)
- CAFARO, Massimo (Univ. del Salento, Lecce)
- VADACCA, Salvatore (Univ. del Salento, Lecce)
- NEGRO, Alessandro (Univ. del Salento, Lecce)
- DEL FREO, Federico (EGG project)
- RICCI, Giovanni (EGG project)

BioinfoGRID
Bioinformatics Grid Application for life science

- Work supported in part by BioinfoGRID and LIBI projects

LIBI International Laboratory of Bioinformatics