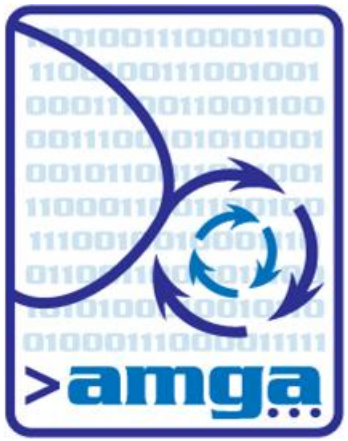
Design and Implementation of WS-DAIR for AMGA

N. KIM, S. Ahn, S. Hwang (KISTI)
A. Calanducci (INFN)
B. Kobliz (CERN)



Index

I. Overview of AMGA

II. Integration of WS-DAIR

III. Performance Study

IV. OGSA-DAI WS-DAIR vs. AMGA WS-DAIR

V. Issues

VI. Future Work

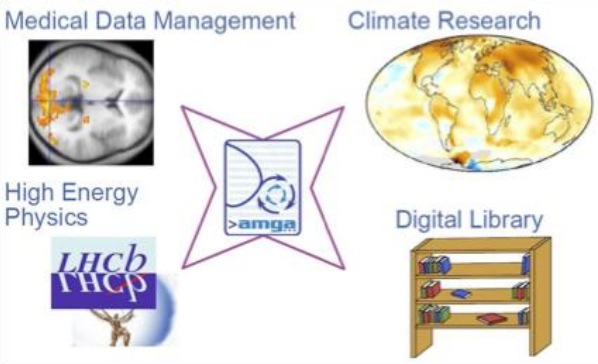
I. Overview of AMGA

What is WS-DAIR ?

- **AMGA(ARDA Metadata Grid Application)**
- **Began as prototype to evaluate the Metadata Interface**
  - Evaluated by community since the beginning:
  - Matured quickly thanks to users feedback
- **Metadata Catalogue of EGEE's gLite 3.1 Middleware**
- **Currently AMGA is used**
  - metadata catalog
  - simplified DB access service on the Grid

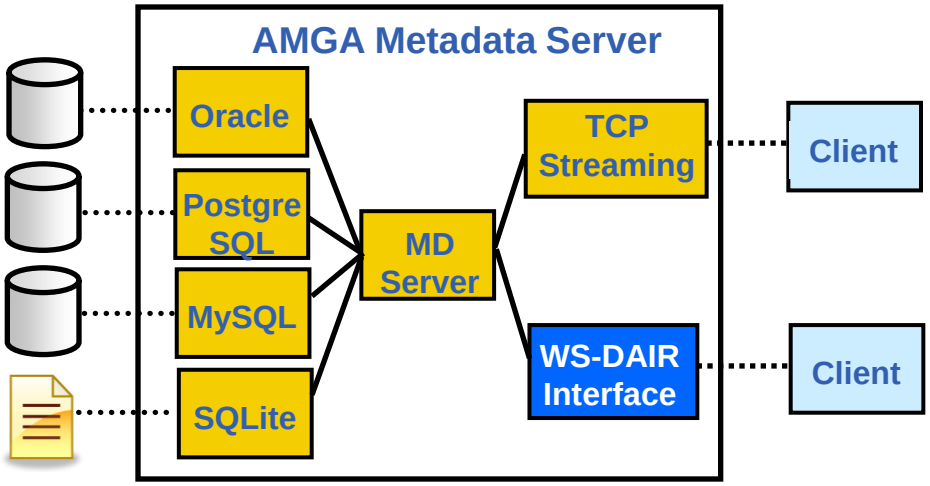
I. Overview of AMGA

- **Requirements from HEP community**
  - Millions of files, 6000+ users, 200+ computing centres
  - Mainly (real-only) file metadata
  - **Main concerns : scalability, performance, fault-tolerance, Support for Hierarchical Collection**
- **Requirements from Biomed community**
  - Smaller scale than HEP
  - **Main concerns : Security**
- **AMGA in preproduction with several projects**
  - LHCb, GANGA, gLibrary, MDM, WISDOM



I. Overview of AMGA

Main Feature of AMGA

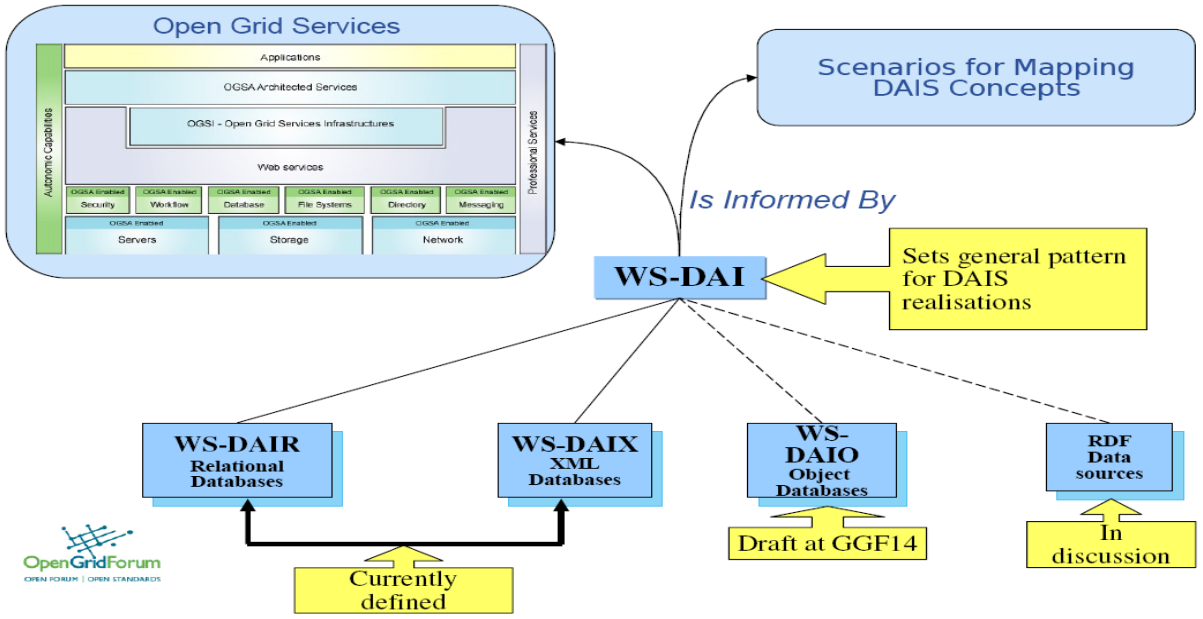


- **Modular back-end**
  - Oracle, PostgreSQL, MySQL, SQLite
- **Modular front-end**
  - TCP Streaming, **WS-DAIR**
- **Hierarchical Organisation**
  - Metadata organised in a tree-like structure
- **Security : SSL, GSI, VOMS, and ACL**
- **Replication : Master/Slave Model**
- **Import existing databases to AMGA**
- **Language**
  - AMGA Metadata Language, SQL-92 Direct Data Statement

II. Integration of WS-DAIR

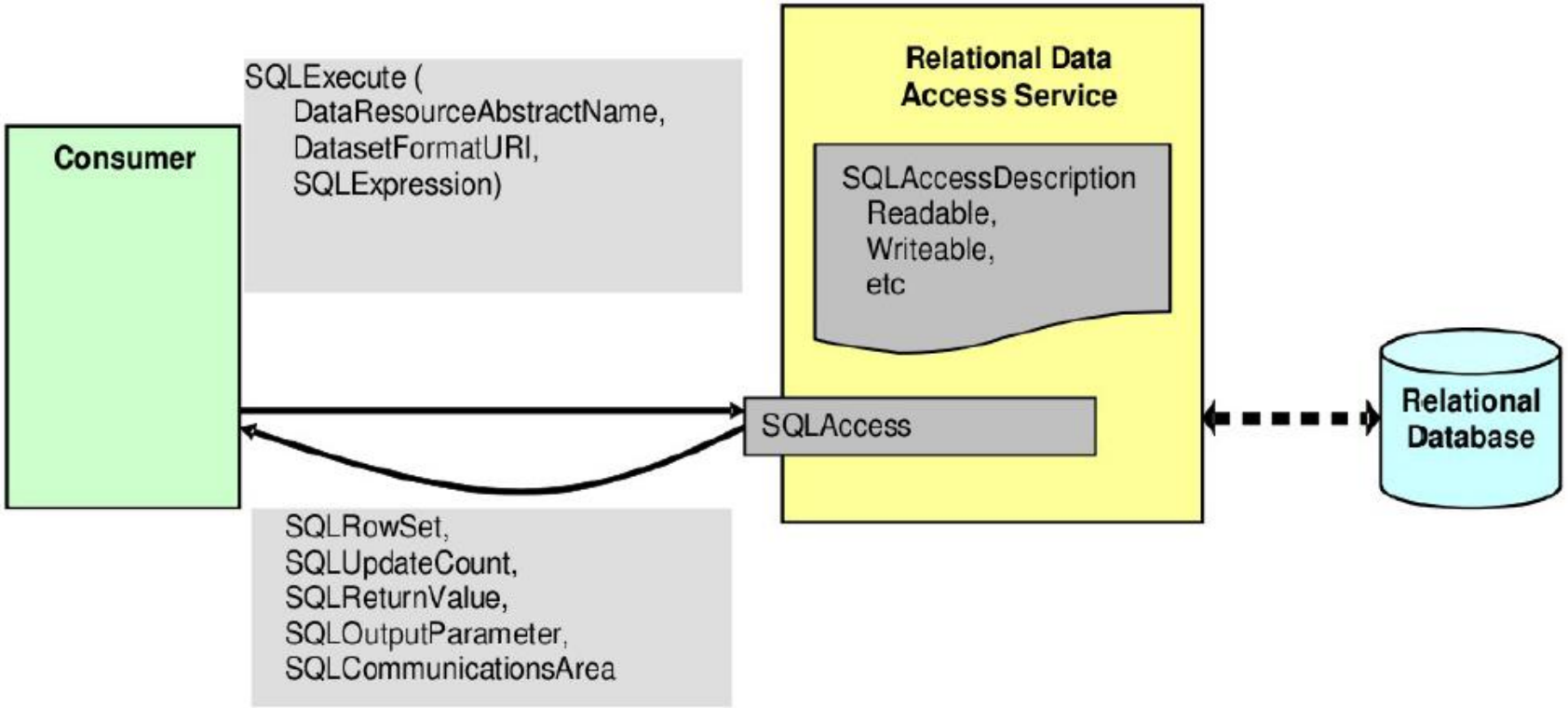
What is WS-DAIR ?

- **Web Services Data Access and Integration - The Relational Realization**
- **Proposed OGF standards Recommendation for access to relational DB's on the Grid**
  - Allows AMGA a seamless integration into the OGF standardized Grid Data Access Services



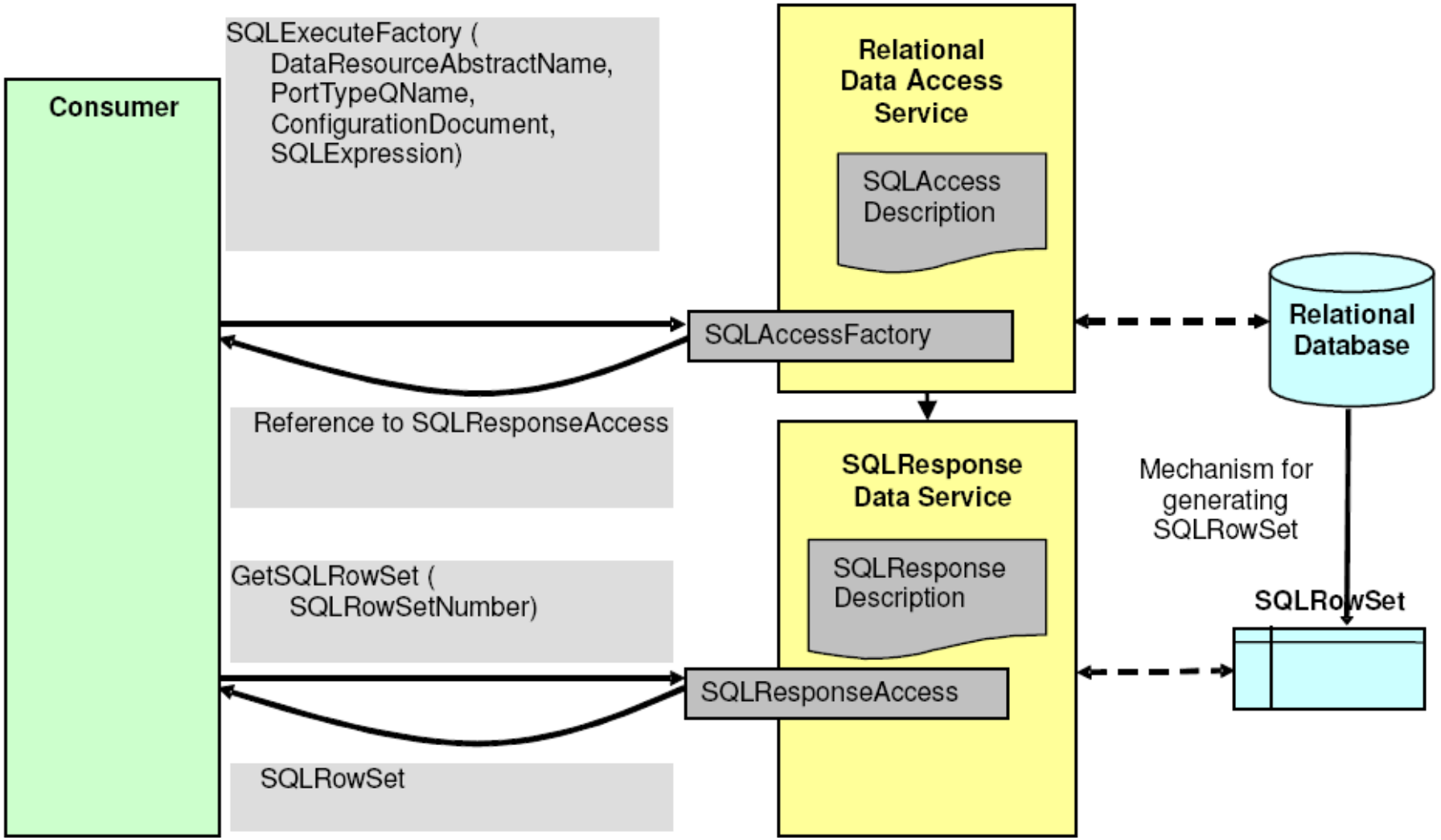
II. Integration of WS-DAIR

Direct Data Access



II. Integration of WS-DAIR

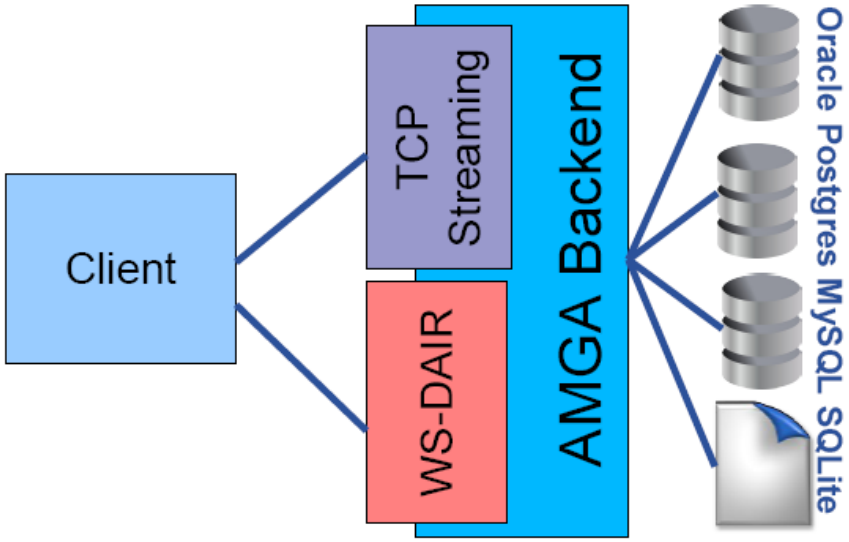
Indirect Data Access



II. Integration of WS-DAIR

AMGA-DAIR Integration

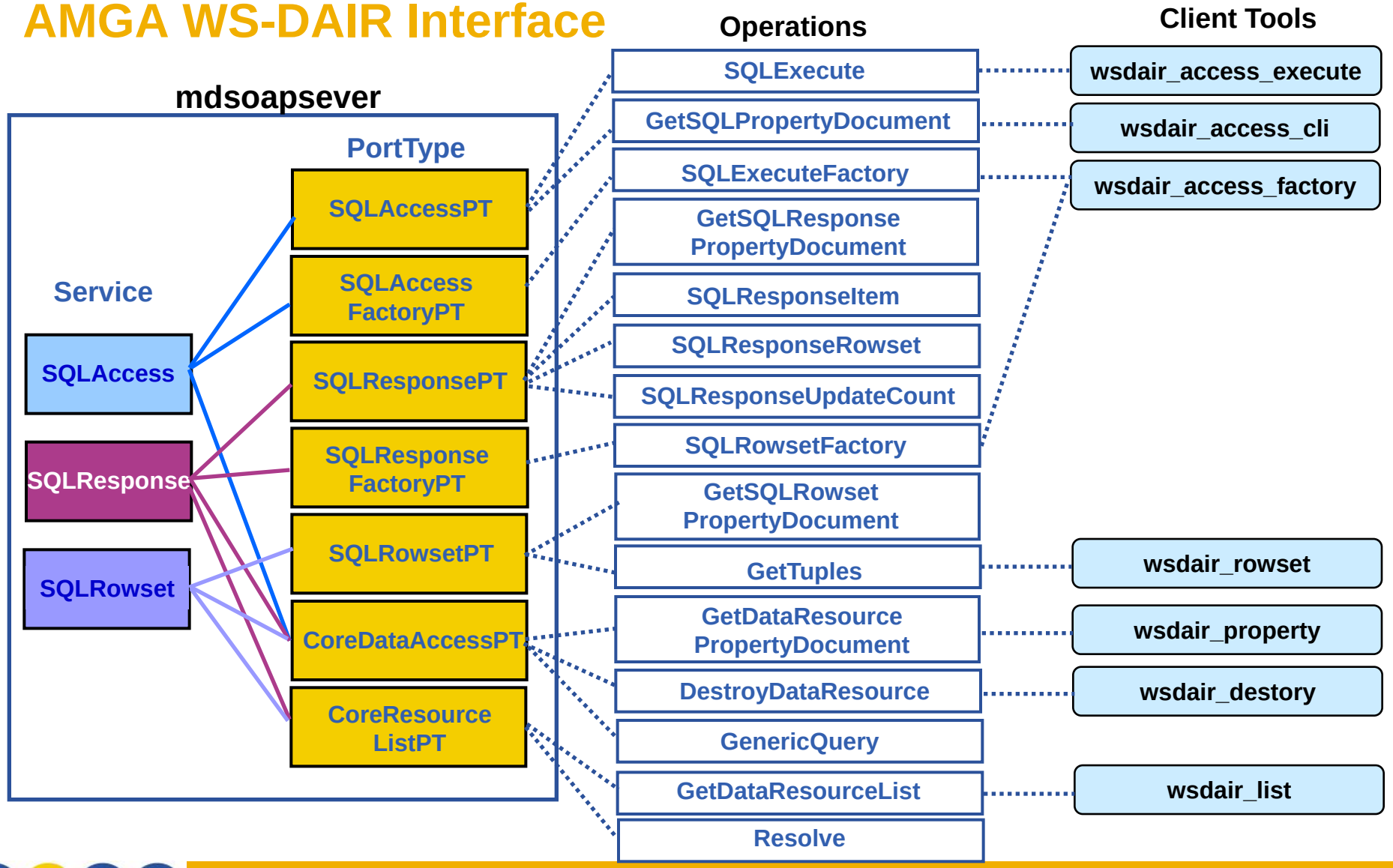
- **WS-DAIR interface replaces the existing SOAP interface, written in gSOAP:**



- **Implementation of direct access**
- **Implementation of indirect access**
- **Data returned in Sun's WebRowSet specification**
- **All queries use Native SQL language**

II. Integration of WS-DAIR

AMGA WS-DAIR Interface



II. Integration of WS-DAIR

AMGA WS-DAIR Interface Implementation

- **Written in C++ (gSOAP)**
  - SOAP Binding : document/literal
  - **Given WSDLs in WS-DAIR specification were used with few modification**
- **Features**
  - Supported Dataset Format : SUN JDBC WebRowSet (default)
  - Supported Language : SQL-92 Direct Data Statement, AMGA Metadata Language
- **Security : SSL, GSI, VOMS, and ACL**
- **Indirect Data Access Service**
  - **Data for a new indirect service is stored as a DB VIEW**

II. Integration of WS-DAIR

Client Tools

- **$ wsdair_access_cli “SELECT * FROM /simulation”**

  ….

- **$ wsdair_access_factory “SELECT * FROM /simulation”**

  http://....:8833/SQLResponse/responses487_19

- **$ wsdair_rowset –P 0 –C 1 http://....:8833/SQLResponse/responses487_19**

- **$ wsdair_list**

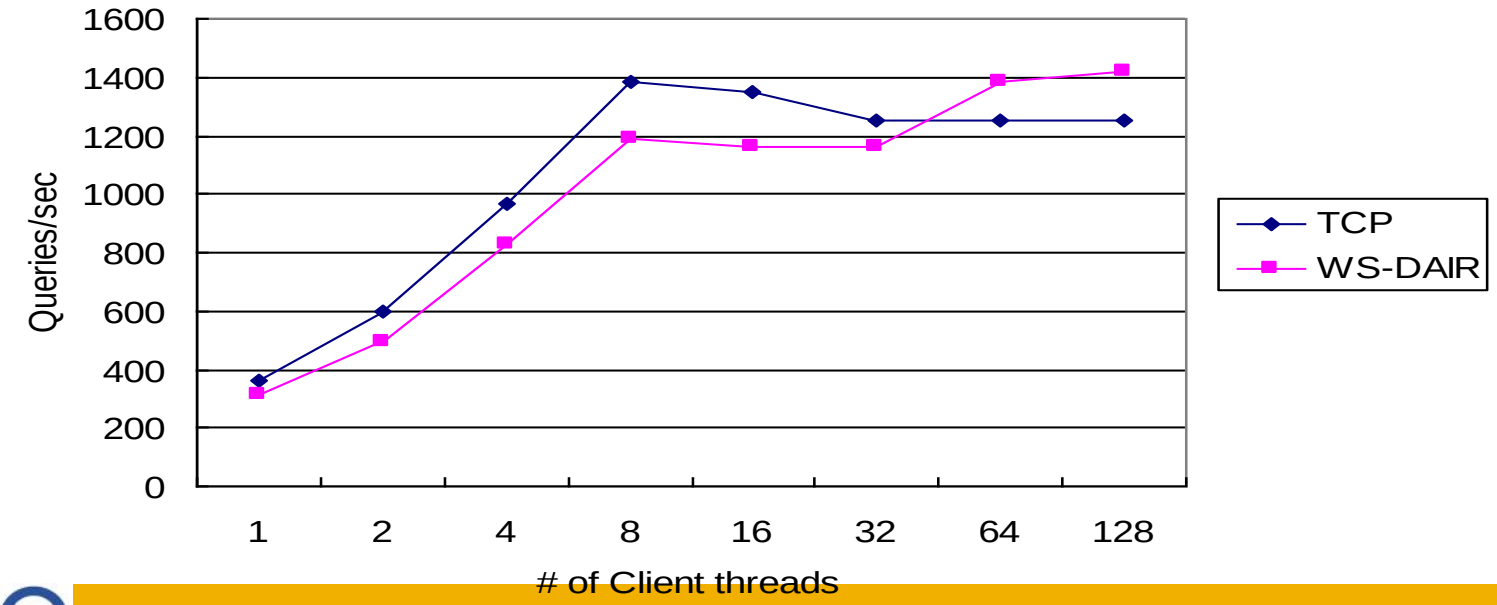
  http://....:8833/SQLResponse/responses487_19

- **$ wsdair_property http://....:8833/SQLResponse/responses487_19**

- **$ wsdair_destroy http://....:8833/SQLResponse/responses487_19**

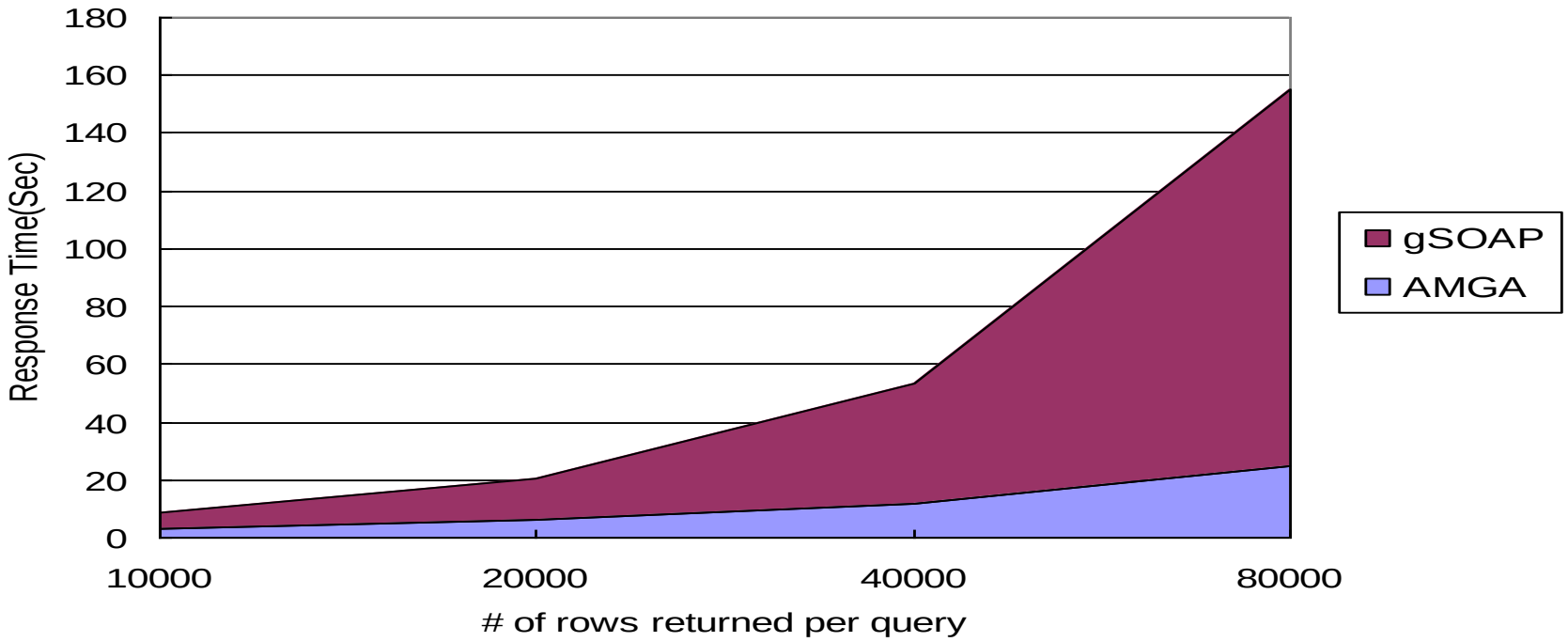
Performance Study : Throughput

- **Testing Environment (Direct Data Access)**
  - DB : the simulation table used for the WISDOM project
  - One row with two attributes was retrieved by each query
- **Results**
  - Little difference between TCP/IP streaming and WS-DAIR SOAP
  - Maximum Throughput : about 1,400 queries / sec



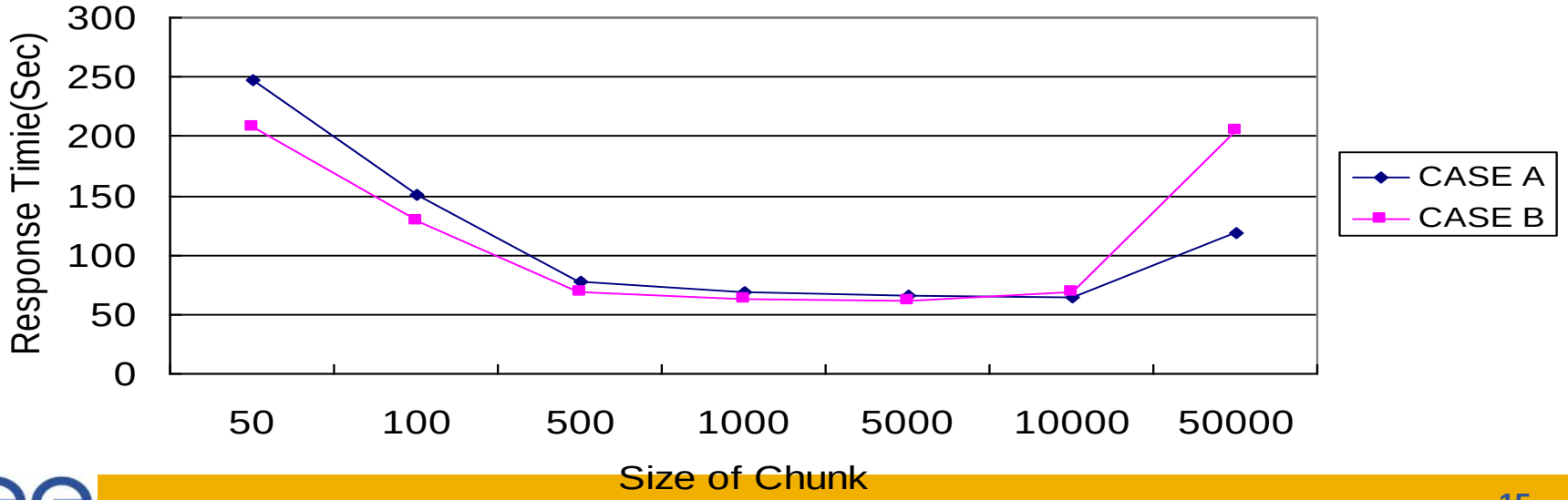
Performance Study : Response Time

- **Testing on response time taken to retrieve various # of rows per query**
  - DB : average 4.5Kbyte per row, 19 columns (ligand table in WISDOM)
- **Results**
  - Response time increases exponentially when # of rows increases



Performance Study : Response Time

- **Testing on response time taken to retrieve 100,000 rows with various chunk size (Indirect Data Access)**
  - Case A : 4.5Kbyte for each row, 19 columns (ligand table)
  - Case B : 250byte for each row, 32 columns (simulation table)
- **Results**
  - Minimum response time : in around a chunk of 1,000 rows
  - Response time depends on # of elements in a SOAP message rather than the size of data in case of a large size of chunk



IV. OGSA-DAI WS-DAIR vs. AMGA WS-DAIR

	OGSA-DAI WS-DAIR	AMGA WS-DAIR
Dev. Tools	Axis, Java	gSoap, C++
SOAP binding	rpc/literal	document/literal
Supported DBMS	MySQL, (tested) Oracle, DB2, PostgreSQL ..	PostgreSQL, MySQL, SQLite Oracle (partially)
Security Feature	NO	SSL, GSI, VOMS, and ACL
# of databases supported in a service	Many	1 (AMGA metadata DB)
Supported Language	SQL-92	- SQL-92 Entry Level Direct Data Statement - AMGA Metadata Language
Supported Dataset Format	WRS, CSV	WRS
Throughput	Tens of queries per sec	Hundreds of queries

V. Issues

1. Handling Large Size of data

- **Requesting very large size of data may cause the service crash.**
- **Possible Solutions**
  - returns error message on Requesting too large size of data
  - Streaming : not defined in SOAP 1.1 & WSDL 1.1
    - MTOM (SOAP 1.2)
    - DIME (extension of WSDL 1.1)

2. Response time increases exponentially when # of elements in a SOAP message increases

- **Other very simple Dataset Formats other than WebRowSet will be examined (e.g. csv)**

3. Interoperability between AMGA WS-DAIR vs OGSA-DAI WS-DAIR

- **Due to different soap binding type between two S/W to communicate with each other is impossible up to now.**

Future Work

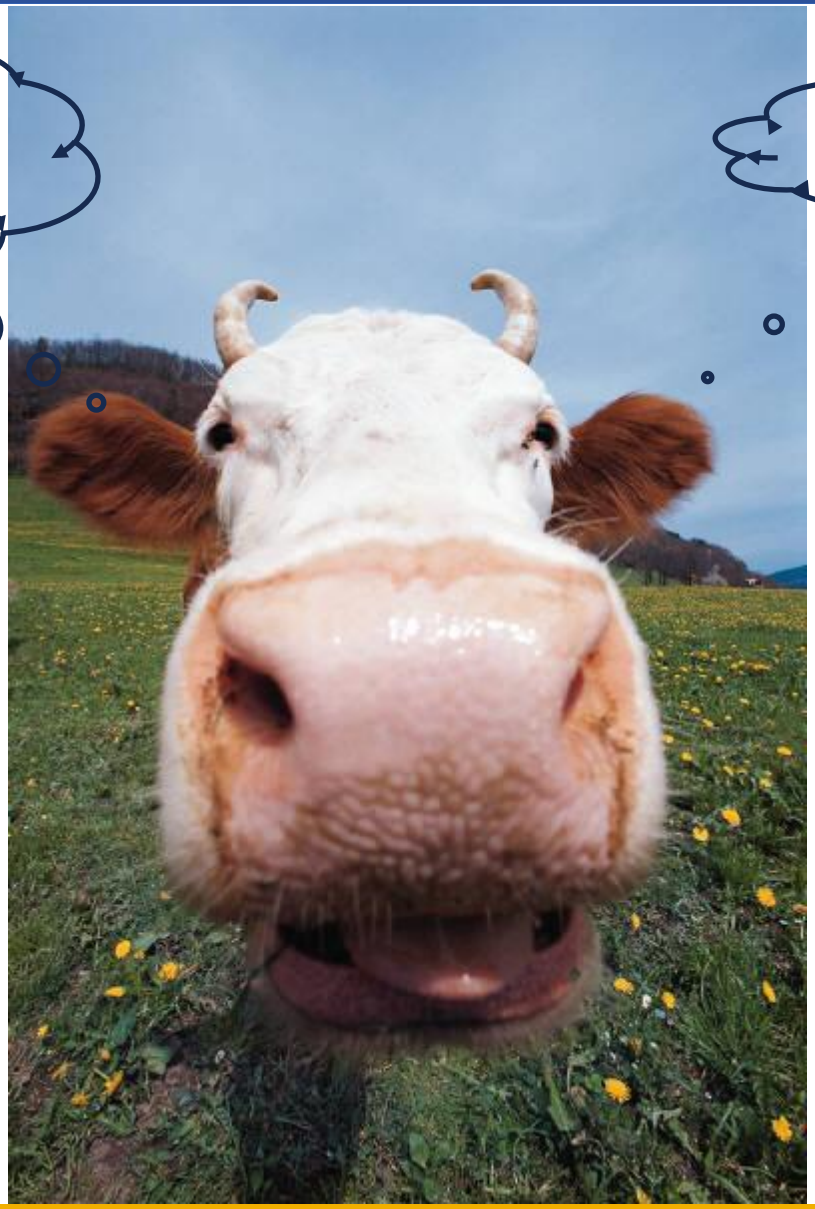
Future Work

- Handling very large size data
- Interoperability against OGSA-DAI WS-DAIR in OGF

Release

- 2.0 : WS-DAIR support (Spring 2009)

Thank You for your attention!



Questions?

Appendix

Modified WSDLs in WS-DAIR specification

- SQLExpression in <WS-DAIR Appendix A.2 SQLAccess WSDL>

```
<!--xsd:element abstract="true" name="SQLExpression"
type="wsdair:SQLExpressionType"/ -->
<xsd:element abstract="false"  name="SQLExpression"
type="wsdair:SQLExpressionType"/>
```

- DatasetDataType in <WS-DAI Appendix A.1 Core XML Schema>

```
<xsd:complexType name="DatasetDataType" mixed="true">
<xsd:sequence>
<!--xsd:any namespace="##any" minOccurs="0" maxOccurs="unbounded"
<xsd:element ref="wrs:webRowSet" minOccurs="0"  maxOccurs="1" />
</xsd:sequence>
</xsd:complexType>
```