Service-Based Distributed Query Processing on the Gird: OGSA-DQP

M. Nedim Alpdemir
Department of Computer Science
University of Manchester

THE UNIVERSITY of MANCHESTER

places, people, funding, projects

Manchester	Newcastle upon Tyne
M Nedim Alpdemir	Arijit Mukherjee
Anastasios Gounaris	Jim Smith
Norman W Paton	Paul Watson
Alvaro A A Fernandes	
Rizos Sakellariou	

acknowledgement

- Much of the content in many of the slides has been authored co-workers, especially A A A Fernandes, A Gounaris and N W Paton.

- Errors are mine, of course.

outline

- Motivation, Context, Background
- Issues, Challenges, Innovation
- OGSA-DQP:
  - A brief tour, Example
  - Design choices
  - Design consequences
  - OGSA-DQP as a service aggregator
- Summary

motivation

- Pull by applications:
  - overwhelming amounts of semantically complex data in
  - very diverse, structurally dissimilar, and autonomous, geographically dispersed data sources
  - requiring computationally demanding analysis.
- Push from context and infrastructure:
  - Web service impetus combined with
  - Grid abstractions and protocols that enable,
  - not just dynamic resource discovery but also,
  - dynamic resource allocation and use.

context

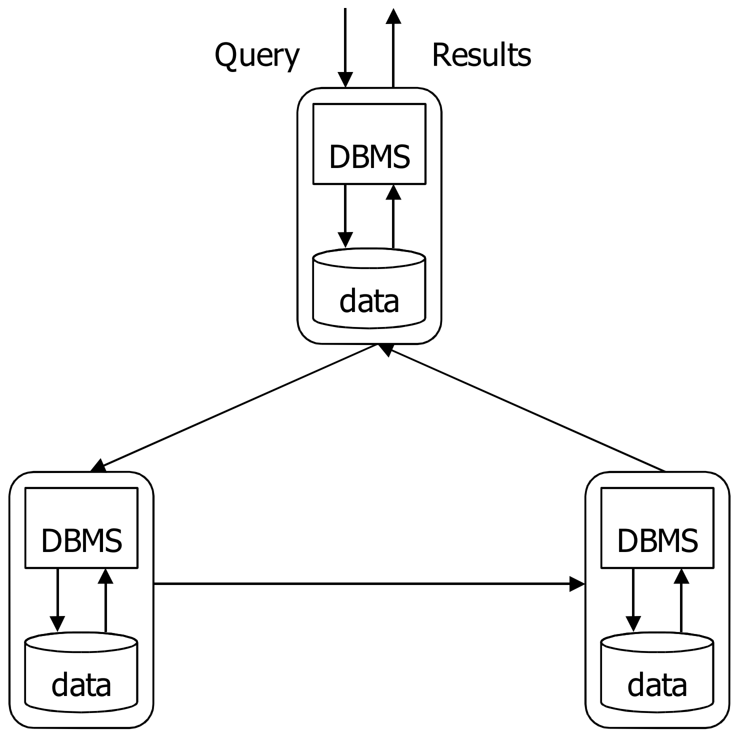
1. High-level data access and integration services are needed if applications that have data with complex structure and complex semantics are to benefit from the Grid.
2. Standards for data access are emerging, and middleware products that are reference implementations of such standards are already available.
3. Distributed query processing technology is one approach to delivering (1.) given the availability of (2.).
4. Declarative service orchestration falls out.

service-based in what sense?

- Service-based in two orthogonal sense:
  - Supports querying over data storage and analysis resources made available as services
    - Hence resource virtualisation via SOA
  - Construction of distributed query plans and their execution over the grid are factored out as services

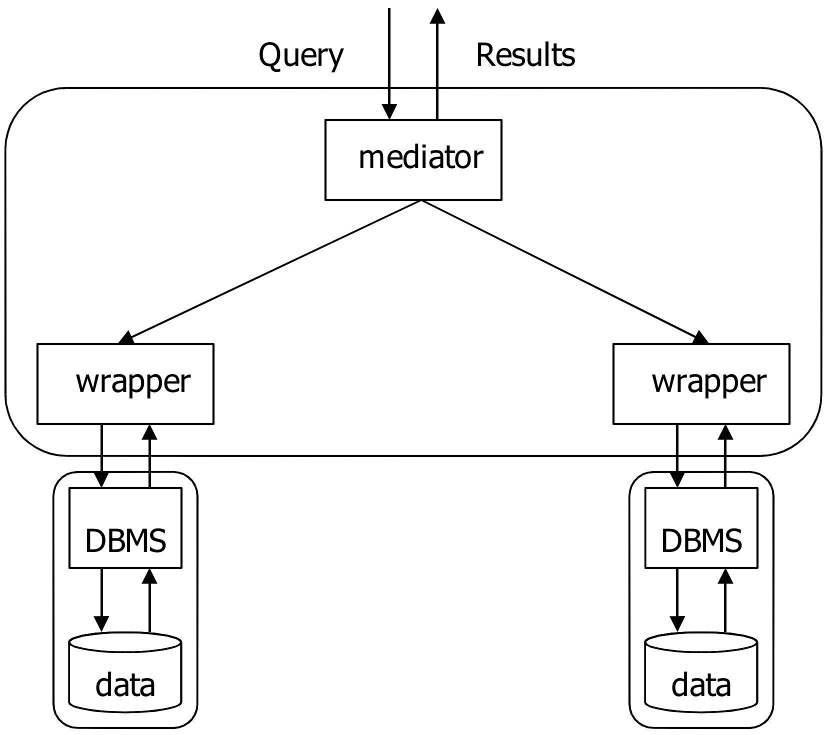
THE UNIVERSITY of MANCHESTER

background: federation



- One DQP approach is to federate.
- This is complex to manage because there always is an autonomous DBMS at every node.
- Resource allocation is static.
- Query optimisation cannot take evolving circumstances into account.

background: mediation

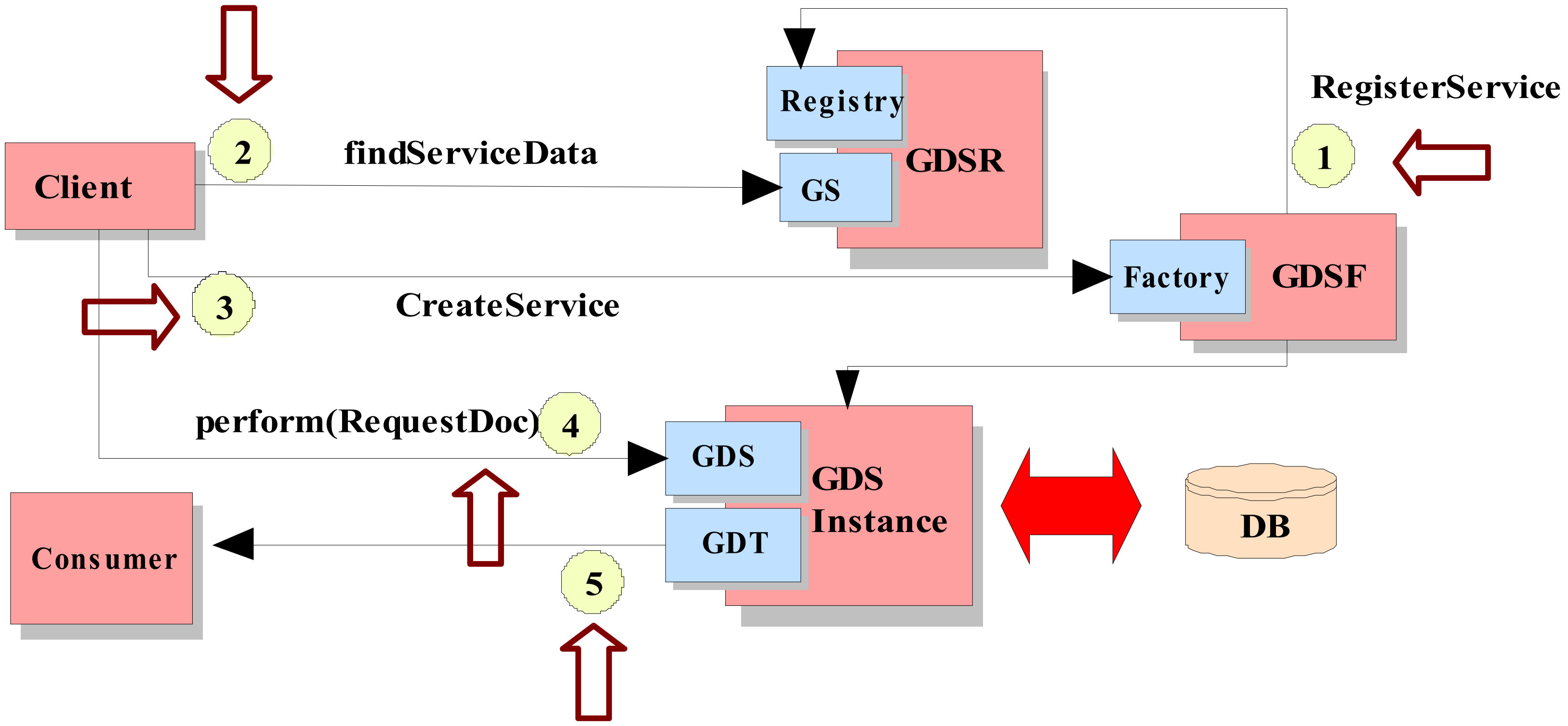


- Another approach is to use mediator/wrapper middleware.
- Autonomy is now less of an issue because the wrappers reconcile differences and impose a global schema.
- Often there is only one mediator in one fixed location, which is limiting.
- Query optimisation is not as hard.

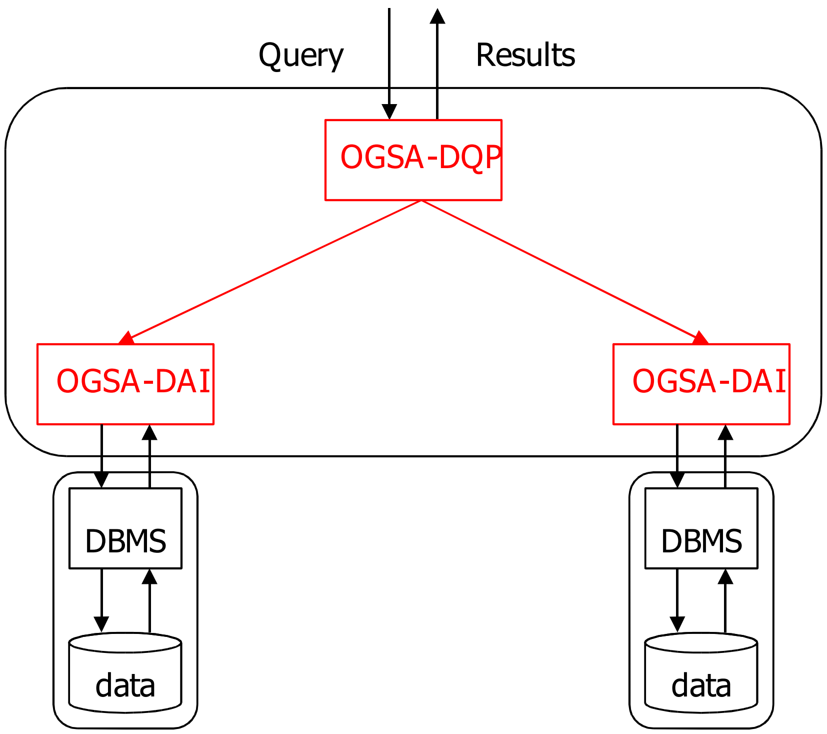
Background:OGSA/OGSI

- The Open Grid Services Architecture (OGSA) integrates key Grid technologies with Web services mechanisms to create a distributed system framework based around the Open Grid Services Infrastructure (OGSI).
- OGSI version 1.0 defines a component model that extends WSDL and XML Schema Definition to incorporate the concepts of:
  - stateful Web services (Service Data Elements (SDEs)),
  - inheritance of Web services interfaces,
  - asynchronous notification of state change,
  - references to instances of services (GSH),
  - collections of service instances (Service Groups)

Background:OGSA-DAI Interfaces &Typical interaction patterns



OGSA-DQP approach



- OGSA-DQP uses a middleware approach.
- It can be seen as a mediator over OGSA-DAI wrappers.
- It promises bottom-lines regarding:
  - efficiency: “leave to it to schedule in parallel”;
  - effectiveness: “leave to it to orchestrate your services”;
  - usability: “use it as a Grid data service”.

where we are

- Motivation, Context, Background
- Issues, Challenges, Innovation
- OGSA-DQP:
  - A brief tour, Example
  - Design choices
  - Design consequences
  - OGSA-DQP as a service aggregator
- Summary

OGSA-DQP goals

1. To benefit from homogeneous access to heterogeneous data sources [OGSA-DAI].
2. To benefit from Grid abstractions for on-demand allocation of resources required for a task [OGSA/OGSI/GT3].
3. To provide transparent, implicit support for parallelism and distribution. [Polar*]
4. To orchestrate the composition of data retrieval and analysis services.
5. To expose this orchestration capability as a Grid data service.

DQPE design issues

- A DQP engine (DQPE) is a distributed data-flow engine.
- How to map distributed query processing concepts and techniques to a service-based infrastructure?
  - What are DQPE components internally realized as?
  - What is the DQPE externally exposed as?
- How to partition functionalities and responsibilities among components?
- What interaction discipline to impose on components?
- Which control flows and which data flows among components?
- What component lifetime management policies to adopt?

DQPE design challenges

- To take decisions dynamically, on an as-needed basis;
- To automate complex, expert configuration decisions on behalf of users;
- To obtain the right grain when partitioning and scheduling computation loads;
- To keep interoperation overheads and context switches low;
- To achieve efficient throughput in data movements;
- To allocate resources lazily and free them up as early as possible.

OGSA-DQP innovations

- OGSA-DQP dynamically allocates evaluators to do work on behalf of the mediator.
- This allows for runtime circumstances to be taken into account when the optimiser decides how to partition and schedule.
- OGSA-DQP uses a parallel physical algebra: most mediator-based query processors do not.

OGSA-DQP provides two grid services

Exposes to clients

- Grid Distributed Query Services (GDQSs) that:
  - interact with clients;
  - find and retrieve service descriptions;
  - parse, compile, partition and schedule the query execution over a union of distributed data sources.
- The query plan is an orchestration of GQESs

Coordinates transparently

- Grid Query Evaluation Services (GQESs) that:
  - implement the physical query algebra;
  - implement the query execution model and semantics;
  - run a partition of a query execution plan generated by a GDQS;
  - interact with other GQESs/GDSs/WSs but not with clients.

where we are

- Motivation, Context, Background
- Issues, Challenges, Innovation
- OGSA-DQP:
    - A brief tour, Example
    - Design choices
    - Design consequences
    - OGSA-DQP as a service aggregator
- Summary

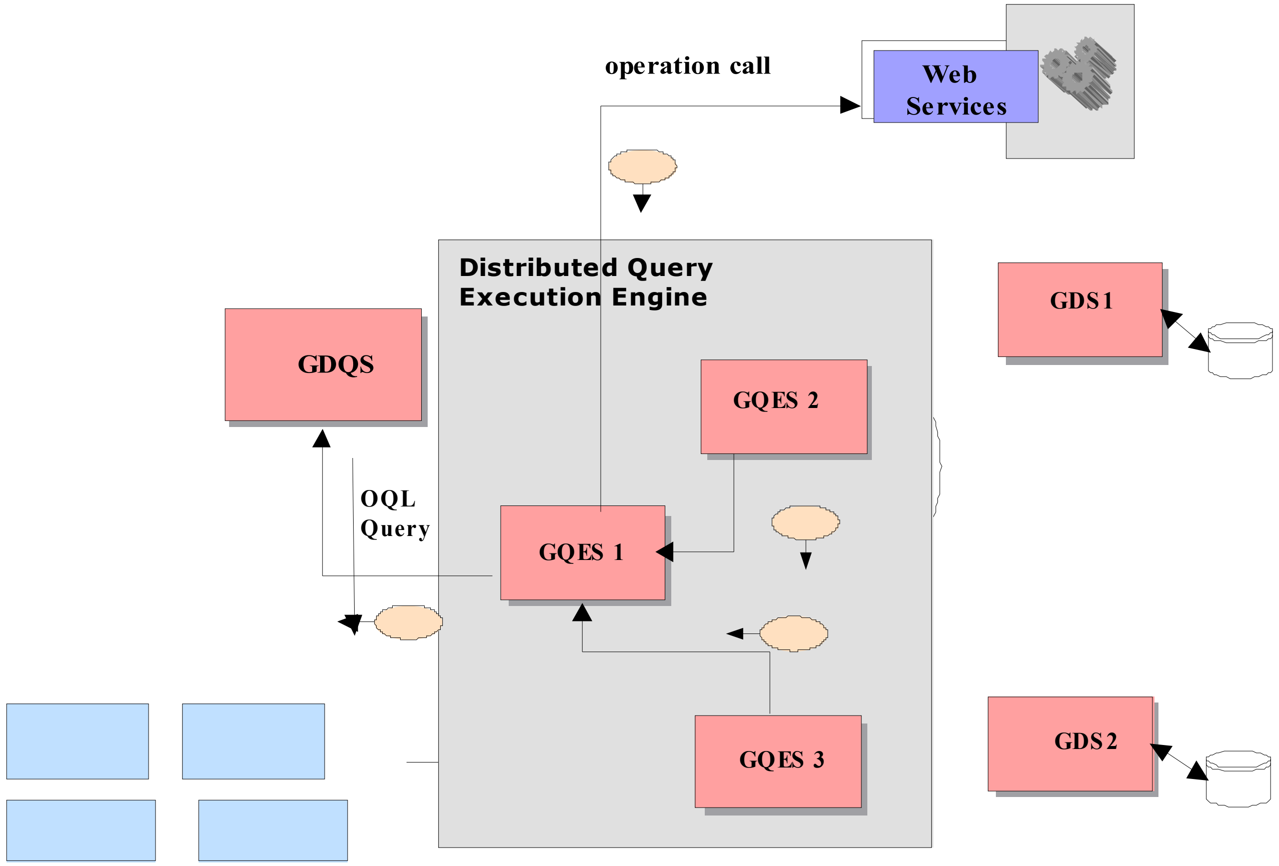
OGSA-DQP
a brief tour:initialisation and setup (1)

- To generate an execution plan, a GDQS:
  - retrieves information about the data and computational services deemed of interest by the requestor;
  - Interacts with GDSFactories, obtains WSDL docs of analysis services and (in future) Index Services to acquire relevant metadata;
  - Compiles, optimises, partitions and schedules the query execution.

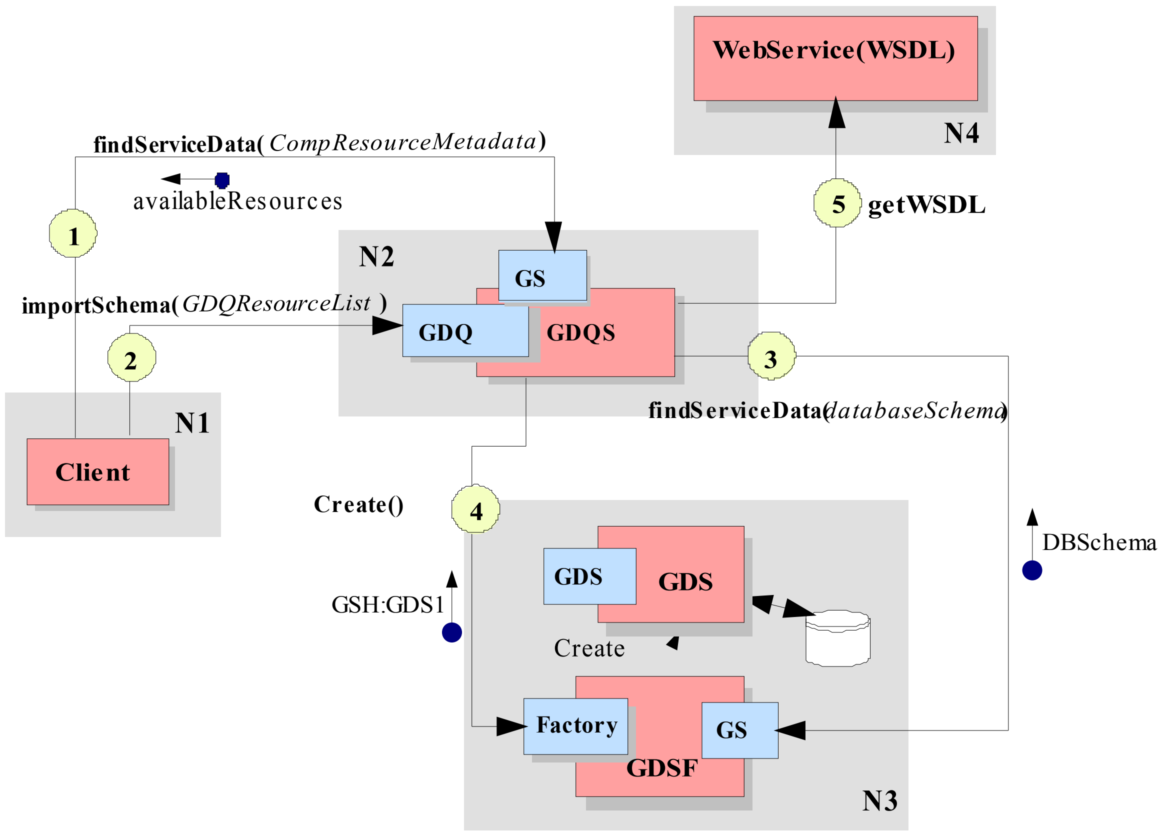
OGSA-DQP a brief tour (2)

- Given a distributed query plan, a GDQS:
  - Interacts with GDS factories to create the leaf services (I.e. data source wrappers) in the plan;
  - Commands the creation of GQESs as stipulated by the partitioning and scheduling decided on by the query optimiser;
    - Instances of GQESs are created as specified by the query execution plan
  - Coordinates the GQESs into executing the plan.
    - Partitions are allocated to service instances (i.e., computational nodes) for execution according to schedule.

Brief tour: an illustration

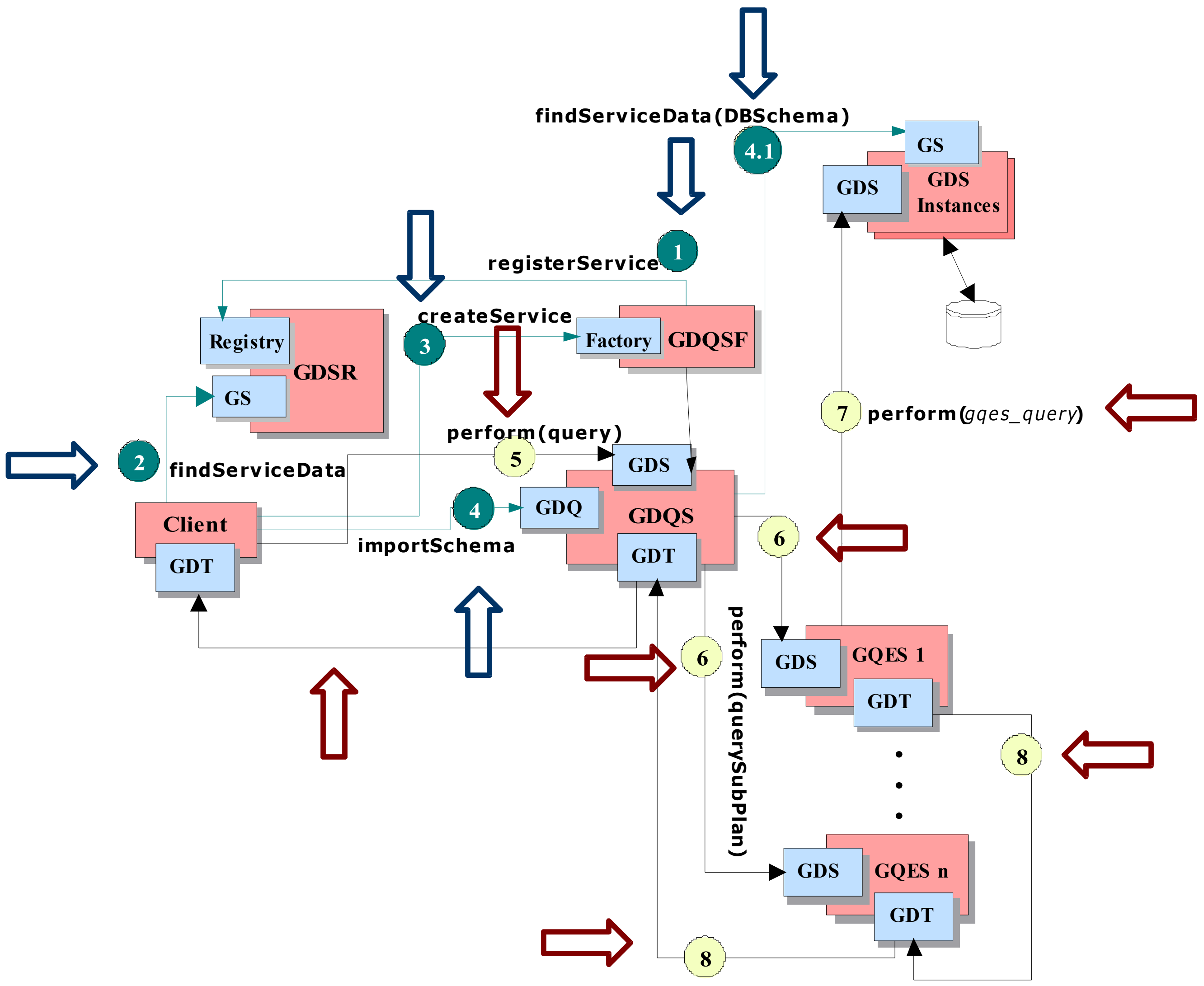


Importing data/analysis resource metadata



THE UNIVERSITY of MANCHESTER

what is going on behind the scenes (1)

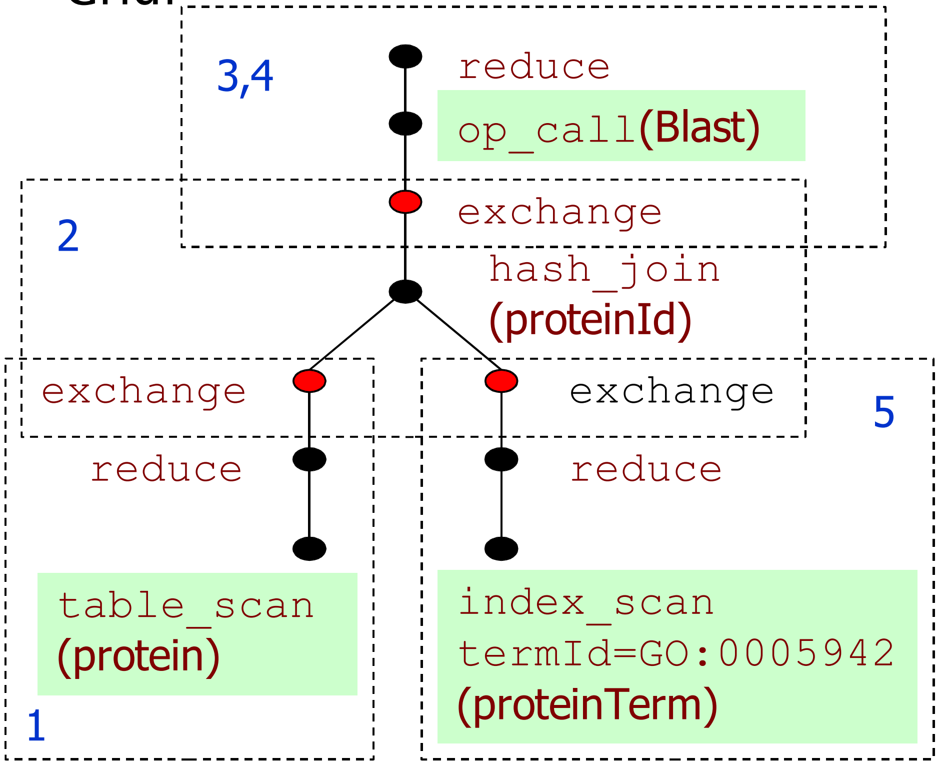


OGSA-DQP example

- Given two DBMSs and one analysis tool (e.g., a WS):
  - **proteinTerm** to a GO Gene Ontology running as a remote mySQL DB,
  - **protein** to a GIMS Genome Warehouse running as a remote ODMG-compliant DB,
  - **Blast** (sequence alignment scoring);
- We can obtain alignment scores for a sequence against proteins of a certain kind:

```
select p.proteinId, Blast(p.sequence)
from   protein p, proteinTerm t
where  t.termId = 'GO:0005942' and
       p.proteinId = t.proteinId
```

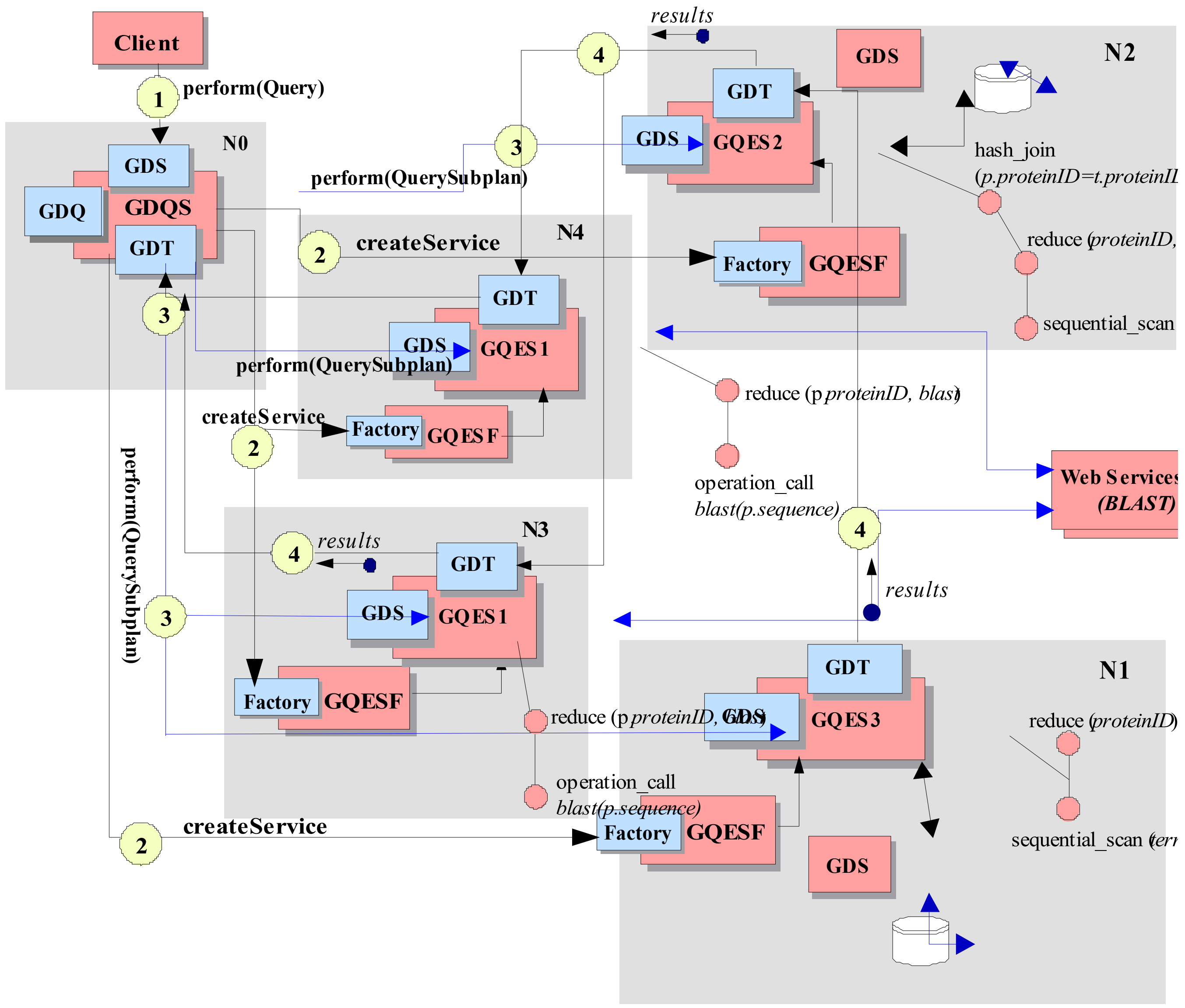
- Then, OGSA-DQP acts as an enactor of a declarative orchestration of services on the Grid:



how is a query evaluated?

- Query installation stage:
  - As many GQES instances are created as there are partitions specified.
  - Each partition is sent to the GQES instance it is scheduled for.
- Query evaluation stage:
  - Each GQES evaluates its partition using an iterator model.
  - Queries execute under pipelined and partitioned parallelism.
  - Results are conveyed to client.

what is going on behind the scenes (2)



can I ask more questions on that?

- When is a GDQS bound to a particular GDS?
  - When the schema of the GDS is imported.
- What is the lifespan of a GDS used by a GDQS?
  - The GDS is kept alive until the GDQS expires.
- Are GDSs shared by multiple GDQSs?
  - No.
- When is a GQES created?
  - When a query is about to be evaluated that needs it.
- What is the lifespan of a GQES?
  - It lasts only as long as a single query.
- Is a GQES shared among several queries or GDQSs?
  - No.

where we are

- Motivation, Context, Background
- Issues, Challenges, Innovation
- OGSA-DQP:
  - A brief tour, Example
  - Design choices
  - Design consequences
  - OGSA-DQP as a service aggregator
- Summary

design choices (1)

- Resource virtualisation through a service-oriented architecture:
  - Data Resource Discovery using service registries;
  - Computational Resource Discovery via Index Services (not implemented yet);
  - Reliance on GDSs for metadata and data access.

design choices (2)

- OGSA-DQP uses coarse-grained services with document-oriented interfaces.
- By acquiring and manipulating data in a data-flow architecture that is constructed dynamically, OGSA-DQP constructs, on-the-fly, a lightweight DQPE.
- Lightweight in the sense that the resources required for query plan execution are acquired at runtime, by instantiating GQESs, and disposed of on a per-query basis.

design choices (3)

- Similarly, metadata acquisition and management is performed per GDQS instance.
- Thus, there is no need for sophisticated mechanisms for maintaining consistency.
- It should be possible to layer on top of (or inside) OGSA-DQP any schema integration framework (e.g., global-as-view, etc.).
- OGSA-DQP performs its functions via an orchestration of loosely-coupled Grid services.

design consequences (1)

- OGSA-DQP design opts for a fairly thick data processing layer that aims to:
  - provide a generic framework,
  - effectively exploit Grid resources and dynamism;
- As opposed to a relatively thin, pre-configured data federation layer that:
  - delegates most of the data processing to data servers, and
  - only controls data flow.

design consequences (2)

- Greater benefits are possible if data is replicated, because only then parallel data access and parallel execution of joins show their true value.
- The existence of multiple instances of the same analysis service (or the ability to create instances of it dynamically) would also boost the performance because it would be possible to stream data simultaneously to multiple services for processing.
- So, the benefit of OGSA-DQP would be more sizeable in the presence of such replicated data sources and services and where large amounts of data are processed.

design consequences (3)

- OGSA-DQP assumes that computational resources can be acquired and used dynamically, which works best when application staging is possible.
  - In the case of true application staging, sophisticated policies may be required that would guarantee that the staged services are properly bound to service providers, both in legal terms and in technical terms (resource provision, security etc.)

design consequences (4)

- Since, currently, true application staging is not supported by Globus Toolkit 3, OGSA-DQP assumes that GQES factories already exist on Grid nodes.
  - Consequently, it is implied that whoever provides OGSA-DQP has either multiple servers with many data sources and evaluators installed on them (less likely), or service level agreements with providers of GDSs as well as of computational resources that host evaluator factories.
  - This implies B2B agreements are in place.

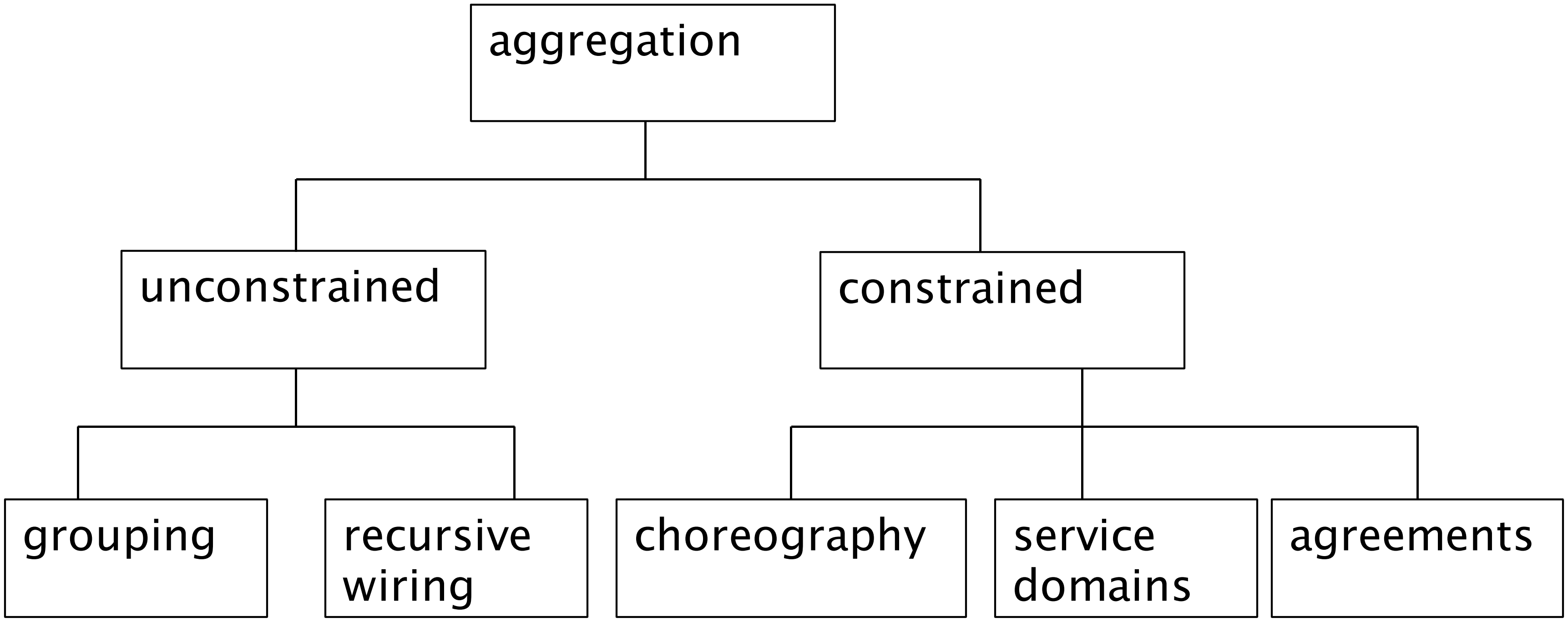
design consequences (5)

- One performance bottleneck is the lack of an efficient data transfer mechanism.
  - As OGSA-DQP is essentially a data flow system, efficient data movement is critical. Currently underlying technology is not mature enough to provide this capability.
  - For example, block delivery does not working efficiently, because:
    - the block size is not configurable;
    - the data is transferred as XML documents using SOAP over HTTP.

where we are

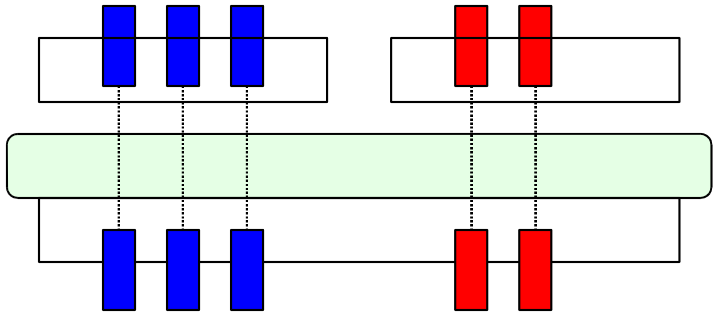
- Motivation, Context, Background
- Issues, Challenges, Innovation
- OGSA-DQP:
  - A brief tour, Example
  - Design choices
  - Design consequences
  - OGSA-DQP as a service aggregator
- Summary

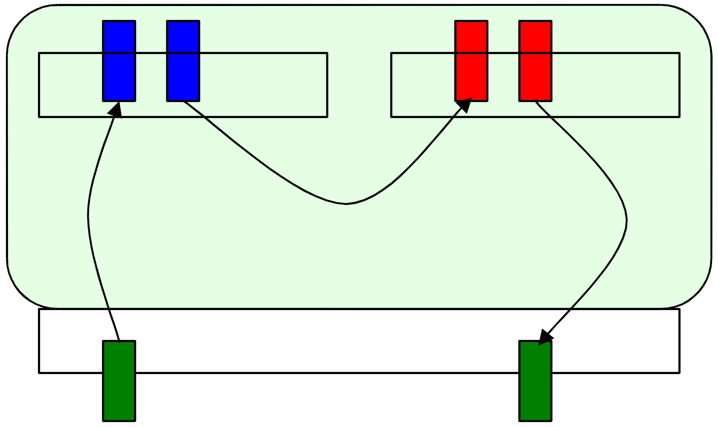
the Khalaf-Leymann taxonomy for web services aggregation



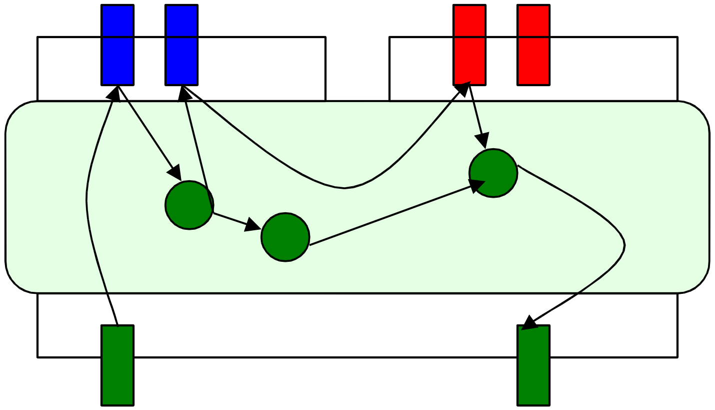
web service aggregation types

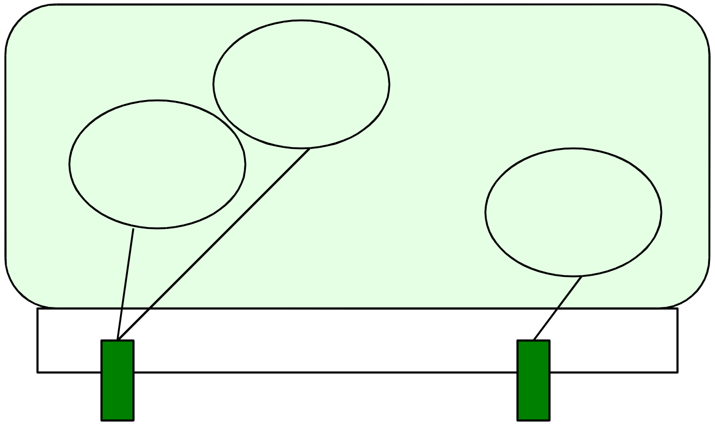
- Grouping
  - Typically is either interface inheritance or instance grouping.
  - It leads to a portal-style of aggregation: no structure, no flow, no synergy.
- Recursive wiring
  - Resembles behind-the-scenes choreography: leads to internal synergy from encapsulated structure and flow.





web service aggregation types





- Choreography
  - Amounts to published orchestration: leads to structure, flow and constraints being made explicit.
- Service domains
  - Are characterized by one-to-many mapping to instances: leads to competition to perform service requests.
- Agreements
  - Are opportunity-driven with short lif

OGSA-DQP as a service aggregator

- There is interface inheritance from GSs and GDSs.
- The query execution plan can be seen as encapsulating a **wiring of GQESs and external web services**,
- But **constrained**, and constructed on-the-fly, as in an **orchestration**.

THE UNIVERSITY of MANCHESTER

possible future work

- More functional:
  - Semi-structured data;
  - Streams.
- More dynamic:
  - Use Index Services;
  - Dynamically install factories.
- More application test-beds:
  - Metabolomics;
  - Sensor networks.
- More adaptive:
  - Queries may be long running, environment is constantly changing, and charging: static optimisation is likely to become stale fast.
  - So: monitor, assess and respond (e.g., switch operators/ algorithms, spawn more copies, relocate).
- More deployable:
  - As a web service, e.g..

summary

- OGSA-DQP is a service-based distributed query processor for the Grid that is:
  - Exposed as a service;
  - Implemented as an orchestration of services.
- OGSA-DQP is an enactor of declarative Grid service orchestrations that:
  - Improves on Grid portals when only retrieval and analysis is involved;
  - Fills the gap left by the lack of a service orchestration framework in the OGSA.

where to find out more: papers

1. M N Alpdemir, A Mukherjee, A Gounaris, A A A Fernandes, N W Paton, P Watson, J Smith. An Experience Report on Designing and Building OGSA-DQP: A Service Based Distributed Query Processor for the Grid. GGF9 Workshop on Designing and Building Grid Services, 2003.
2. M N Alpdemir, A Mukherjee, A Gounaris, N W Paton, P Watson, A A A Fernandes, J Smith. OGSA-DQP: A Service-Based Distributed Query Processor for the Grid. 2nd UK e-Science All Hands Meeting, 2003.
3. J Smith, A Gounaris, P Watson, N W Paton, A A A Fernandes, R Sakellariou. Distributed Query Processing on the Grid. GRID 2002, LNCS 2536

(papers available from http://www.cs.man.ac.uk/~alvaro/publications.html)

THE UNIVERSITY of MANCHESTER

where to find out more: software

OGSA-DQP

Grid middleware to query distributed data sources

www.ogsadai.org.uk/dqp

OGSA-DAI

Grid middleware to interface with data(bases)

www.ogsadai.org.uk/

Globus Toolkit

Open-source implementation of OGSA/OGSI

www.globustoolkit.org/