

# Grid Data Distribution

DAIS F2F, Manchester November 2003  
Joint DAIS-OREP Session

Cecile Madsen (IBM)  
Dieter Gawlick (Oracle), Vitthal Gogate (IBM),  
Shailandra Mishra (Oracle), Inderpal  
Narang (IBM), Mahadevan Subramanian  
(IBM)

# Topics

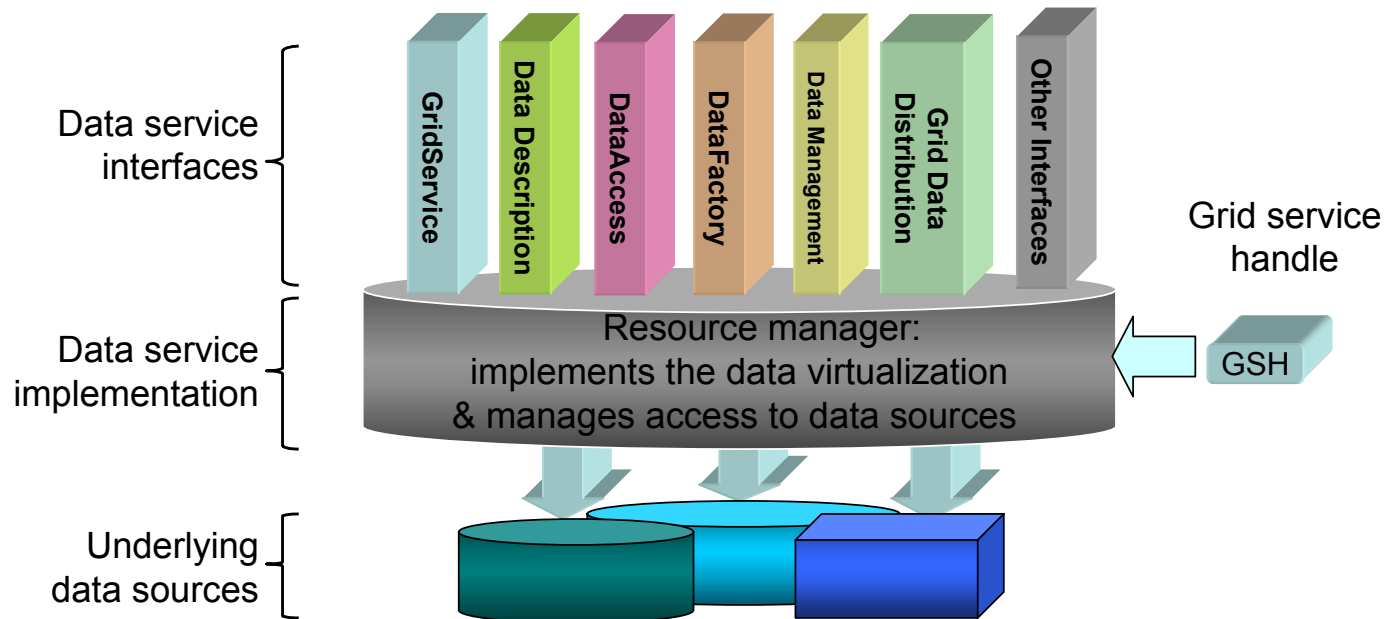
- Grid Data Distribution (GDD) model summary
- Issues raised in DAIS October F2F at ANL, Chicago
- GDD's solution to sample DAIS Scenarios
- GDD Simplification
- What's Next

# GDD

Asynchronous data and event distribution model

- 3<sup>RD</sup> party data delivery, data replication
- based on pub/sub
- dynamic operations (publication)
- administrative tasks and operational tasks with authorization and rules (secure, flexible)
- reliable, only-once delivery semantics
- consistency requirements (transactional)
- tracking and auditing of data
- support of all open data transport protocols

# Grid Data Service (GDS)



1. The GDD is an independent port type defined at the same level as Data Access and other port types
2. Data Service is source and may be a target in GDD model

# GDD Interfaces

- Publication
  - publishing rules (what/who), publisher info, implicit/explicit
  - dynamic publication (no materialization)
- Subscription
  - interest in future data, events (changes to data)
  - filtering rules, subscriber info
- Propagation
  - defines target: may be a Data service
  - distribution/delivery rules (1 subscription, n propagations), may include scheduling, retention, authorization rules

# GDD Interfaces

- Consumption
  - transformation, filtering by consumer at target
  - consumer may be different from subscriber
- Publish at a source
  - publishData
- Deliver at target (push)
  - deliverData, deliverEvents
- Retrieve from source (pull)
  - getData

# Issues raised in DAIS F2F

- GDD portType - Data Access or Data Management ?
  - Proposal: Define GDD as a separate portType
    - Has two sub-portTypes
      - GDDProducer
      - GDDConsumer
    - GDD is defined at the same level as Data Access and Data Management port types.
- Should GDD be decomposed to sub-portTypes ?
  - Please see above.
- How is data access done by GDD ?
  - GDD and Data Access – not related even on same GDS
  - GDD may use Data Access to define publication, subscription etc.

# Issues raised in DAIS F2F

- How does GRAP fits in GDD ?
  - GDD requires negotiated capabilities e.g. Version, Type, Charset, form etc.
  - GDD used GRAP to negotiate capabilities through DataDescription portType
- What kind of monitoring facilities are provided by GDD ?
  - GDD offers monitoring capability through views:
    - Administrative views – PubSub, Propagation rules etc.
    - Security views – User privileges etc.
    - Statistical views - #(Bytes transferred), Last error etc.
  - The views are accessed through the DataManagement portType

# Issues raised in DAIS F2F

- How does GDD handle transactional issues?
  - GDD is message oriented – needs transaction support from GDS for consistency, high performance & scalability
  - For improved control GDD needs recoverable read and fast commit for better performance.

# Issues raised in DAIS F2F

- Can the GDD publications, subscriptions etc. be ‘services’ of their own ?
  - Depends on the implementer, Grid being highly scalable this is not prohibited
- If these are not spawned as services how would a client know about existing publication, subscription identifiers ?
  - The understanding is that these will be supported via the DataDescription portType or through external discovery mechanisms

# GDD – DAIS Scenarios

- Focus of GDD is to cover scenarios with Data distribution with wide range of operational characteristics
- GDD is not interested in scenarios already covered by DAIS

# GDD – DAIS Scenarios

<u>Scen.</u>	<u>GDS</u>	<u>Greg's Ext</u>	<u>GDD</u>	<u>Remarks</u>
1	Yes			Synchronous Query
2		Yes	Yes	
3	?		Yes	
4	Yes			Synchronous Update
5		?		Pull from Non.Svc.
6			Yes	
7			Yes	
8			Yes	

# GDD – DAIS Scenario – (2)

- Analyst locates Global Dataservice:
  - `lookup(global_registry GDS)` returns DSGDH
- Analyst subscribes expressing interest in the data through a query
  - `GDDProducer::createSubscription([implicitname=QueryPublication, SQL Query, scheduleat = 3PM], Analyst)` returns SubslD.
- Analyst specifies that result of the query be delivered to 3<sup>rd</sup> party, this is done through propagation rules
  - `GDDProducer::createPropagation(ConsumerURI, [subscription=SubslD, scheduleat = 9PM, protocol=SMTP, deliveryFormat=WebRowSet])` returns propagationId2.
- At 9 PM the DSGSH uses SMTP to deliver data to the consumer

# GDD – DAIS Scenario – (3a)

- Analyst locates Global Dataservice:
  - `lookup(global_registry GDS)` returns DSGDH
- Analyst subscribes expressing interest in the data through a query – note the `implicitname` clause in the subscription rule.
  - `GDDProducer::createSubscription([implicitname=QueryPublication, SQL Query, scheduleat = 3PM], Analyst)` returns SubslD.
- The analyst asks 3<sup>rd</sup> party consumer to get result data from DSGSH by passing the handle to the customer.
- The consumer specifies the consumption rules and uses `getData` to retrieve the result of the data.
  - `GDDConsumer::createConsumption([subscription=SubslD, dataConsumptionFormat=WebRowSet], Consumer)` returns `consumptionId`.
  - `GDDProducer::getData(consumptionId)`

# GDD – DAIS Scenario – (3b)

- The first three steps are same as (3a)
- The 3<sup>rd</sup> party consumer would specify a schedule to the data service (DSGSH)
  - GDDProducer.createPropagation(ConsumerURI, [subscription=SubsID, scheduleat = 11PM, protocol=FTP, deliveryFormat=WebRowSet]) returns propagationId.
- At 11PM, DSGSH, would use the protocol mentioned for propagationId to send result data to the consumer at consumerURI.

# GDD – DAIS Scenario – (3c)

- The first three steps are same as (3a)
- In this case at G1, we do createPropagation to G2
  - GDDProducer::createPropagation(G2GSH, [subscription=SubsID, scheduleat = 11PM, deliveryFormat=WebRowSet]) returns propagationId.
- At 11PM, data gets pushed to G2
- Also, the other variation here is C subscribes to G2
  - GDDProducer::createSubscription([implicitname=QueryPublication, SQL Query], Analyst) returns SubsID.
- The consumer specifies the consumption rules and uses getData to retrieve the result of the data.
  - GDDConsumer::createConsumption([subscription=SubsID, dataConsumptionFormat=WebRowSet], Consumer) returns consumptionId.
  - GDDProducer::getData(consumptionId)

# GDD Simplification

- The following additional elements are assumed available to provide:
  - A name for a request
    - Provides reference for Alter, Start and Stop
  - The time or conditions of the executions(s) of a request
    - **AT\_TIME | ON\_DEMAND | SCHEDULE | EVENT]**
    - AT\_TIME implies there is one execution
    - ON\_DEMAND and SCHEDULE provides the ability for continuous execution e.g. for time = t1 to time = t2 execute forever
  - Specification determining the delivery

# GDD Simplification

- DELIVERY [{RECIPIENT, INFORMATION, D\_SCHEDULE, QOS}, ...]
  - RECIPIENT [REQUESTOR, ADDRESS, EXPRESSION]
    - REQUESTOR identifies the issuer of the request; and needs to be explicitly specified if other recipients are named
    - ADDRESS identifies the address of a recipient along with a protocol, e.g., SMTP: [Joe@company.com](mailto:Joe@company.com)
    - EXPRESSION [directory reference, expression] identifies all recipients who are listed in the named directory and meet the expression.
  - INFORMATION [DATA | STATUS | FUNCTION]
    - INFORMATION identifies what is provided to specified recipient(s), data and the status, status only, or a function to allow transformations DATA is the default
  - D\_SCHEDULE allows the specification of a delivery schedule.

# What's Next + Reference

- What's Next:
  - agree on priority of to-do items
  - deliver new version of GDD informational paper
  - any volunteers for some topic ?
- GGF9 Data Distribution Informational paper:
  - <http://www.cs.man.ac.uk/grid-db/documents.html>