Gridwire -
A Peer-to-Peer Grid Database
for Service and Resource Discovery

Wolfgang.Hoschek@cern.ch
CERN IT Division

Motivation: Web Services Vision

- Programs no longer configured with static information
- Programs more flexible, adaptive and powerful
  - by querying Internet databases (registries) at runtime
  - to discover services and related metadata
- Services can publish (advertise) themselves and related metadata
  - enabling the assembly of higher-level components
- Example (European Data Grid)
  - data-intensive High Energy Physics analysis application
  - looks for remote services
  - that exhibit a suitable combination of characteristics, including
  - service interfaces, operations and protocols,
  - network load, available disk quota, access rights, and perhaps Quality of Service (QoS) and monetary cost

Examples for Content - Service Description & Host Info

```
<service
  <interface type = "http://edg.org/interface/scheduler-1.0">
    <operation>
      <name> void submitJob(String jobdescription) </name>
      <allow> http://cms.cern.ch/everybody </allow>
      <bind:http verb = "GET"
                 URL  = "https://sched.cern.ch/submit"/>
    </operation>
  </interface>
</service>
```

```
<hostInfo>
  <host name="fred.cern.ch" os="redhat 7.2"  MHz="800"/>
  <host name="carl.cern.ch" os="solaris 2.7" MHz="400"/>
</hostInfo>
```


Discovery XQuery

- *Find all services*
  - *that implement a replica catalog interface*
  - *and that CMS members are allowed to use,*
  - *and that have an HTTP binding for the operation "XML getPFNs(String LFN)".*

```
LET $cat := "http://gridforum.org/interface/replicaCatalog-1.0"
FOR $tuple IN /tupleset/tuple[@type="service"]
WHERE SOME $op IN $tuple/content/service/interface[@type=$cat]/operation
      SATISFIES ($op/name="XML getPFNs(String LFN)" AND
                 $op/bindhttp/@verb="GET" AND
                 contains($op/allow, "http://cms.cern.ch/everybody"))
RETURN $tuple
```

- Everything SQL can do, XQuery can also do
- But XQuery is even more powerful
  - e.g. hierarchical navigation, regular expressions

Problem

- Grids are large distributed systems characterized by
  - heterogeneity, scale, lack of central control, multiple autonomous administrative domains, unreliability and frequent dynamic change
- Not obvious how to maintain information
  - from a large variety of unreliable, frequently changing, autonomous and heterogeneous remote data sources
  - Predictable, timely, consistent and reliable global state maintenance is infeasible
  - The information to be aggregated and integrated may be outdated, inconsistent, or not available at all
  - Failure, misbehavior, security restrictions and continuous change are the norm rather than the exception
  - → classic DBMS architectures do not apply ☹
- Not obvious how to allow for powerful queries
  - over time-sensitive dynamic information
  - while avoiding sacrificing reliability, predictability and simplicity

Tuple from Dynamic Data Model

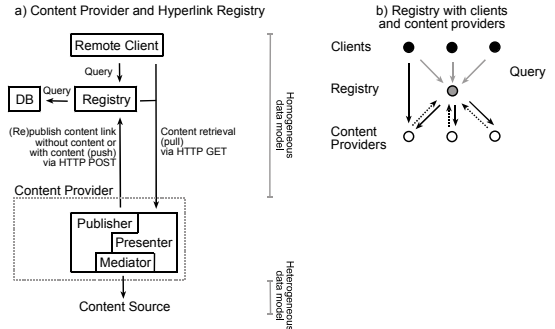
- A tuple is an...
  - annotated multi-purpose soft state data container
  - that may contain a piece of arbitrary MIME content
  - and allows for refresh of that content at any time
  - (default content-type is XML)

Tuple :=

Link	Type	Context	Timestamps	Metadata
Content (optional)				

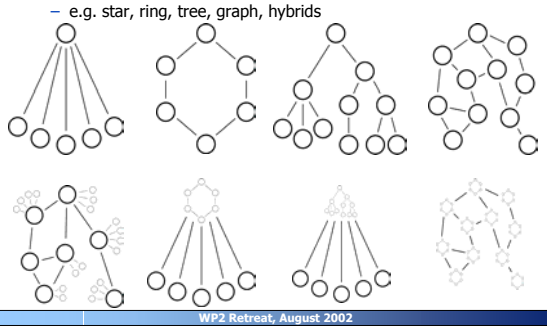
Semantics : HTTP GET(tuple.link) --> tuple.content
type(HTTP GET(tuple.link)) --> tuple.type

Client, Registry and Content Providers



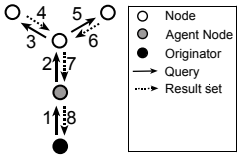
Tuples Partitioned over (Registry) Nodes Linked by Topology

- for autonomy, scalability, availability, performance, security, etc.
- e.g. star, ring, tree, graph, hybrids



P2P Query Flow

- Query Flow
  - An *originator* (e.g. HTML GUI)
  - sends a query to an *agent node* (i.e. gateway),
  - which evaluates it,
  - and forwards it to select neighbor *nodes*,
  - which in turn also evaluate it and forward it to select neighbor nodes, and so on…
- Four Query Response Modes
  - Routed Response
  - Direct Response
  - Routed Metadata Response
  - Direct Metadata Response



Permitted Message Exchanges

- `MSG_QUERY --> RPY_OK | ERR`
- `MSG_RECEIVE --> RPY_SEND | (ANS_SEND [0:N], NULL) | ERR`
- `MSG_INVITE --> RPY_OK | ERR`
- `MSG_CLOSE --> RPY_OK | ERR`

- Supports synchronous (pull) and asynchronous (push)
- Supports batched iterators
  - RECEIVE/SEND batches of at least N and at most M results from the (remainder of the) result set

Demo of early Prototype

- Production quality release: end of December 2002
- Several betas before that
- Current prototype supports…
  - Tuple set backed by JDBC/MySQL
  - Access via Peer Database Protocol over BEEP
  - Publish (with soft state maintenance of JDBC table)
  - Query
  - Receive (only synchronous)
  - Send

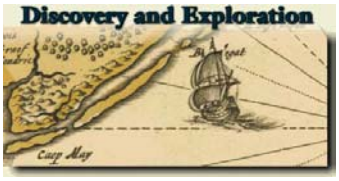
Let's see…

Todo

- Convenient command line tool and user-level interface

```
TupleSet tupleset = node.query(query, options);
Iterator iter = tupleset.iterator();
while (iter.hasNext()) {
       Tuple tuple = iter.next();
       doSomething(tuple);
}
```

- Axis SOAP between originator and agent node
  - Nodes still use PDP among themselves
- Multi-hop interactions
- Asynchronous Receive and Invitation
  - Listener (server) notifies initiator (client) when tuples become available → e.g. publish-subscribe patterns
- And much more…

Questions?

- For more information
  - Several Journal/Conf./Tech. papers on WP2 website
  - http://cern.ch/grid-data-management/