Middleware Adaptation with the Delphoi Service

Jason Maassen, Rob V. van Nieuwoort, Thilo Kielmann, Kees Verstoep

kielmann@cs.vu.nl

Vrije Universiteit, Amsterdam
Adaptive Applications

- Application needs to adapt itself to Grid resources
- Middleware is needed both for
  - getting performance information
  - guide adaptation decisions
- Problem:
  - Monitoring data is low-level (resource centric)
  - Information is scattered among many sources
- Solution: The Delphoi service
Building a Grid Application Toolkit (GAT):

- a simple, high-level, application-oriented API
- independent of underlying middleware (Globus 2, 3, 4, Avaki, Unicore, ...)
- using a set of services (on top of “core” middleware)
- many services need performance information, e.g.
  - data movement and replication
  - remote visualization
  - resource broker
Scope of the GAT API:

- Files
- Resources (CPU)
- Event/Information Exchange
- Utility classes
  (error handling, security, preferences...)

- NOTHING ELSE
GAT Architecture

Application Layer

Application

GAT API
GAT Engine
GAT Adaptors

GAT Layer

User Space

Globus
Sun Grid Engine
GRMS
GRMS
Data Movement
Replica Manager

Capability Space

Delphoi
Globus
iGrid
Mercury
Pythia
Pythia

iGrid
Mercury
Unicore
Delphoi Use Cases

- remote file transfer optimization
  - protocol selection (GridFTP, scp, ...)
  - protocol parameter settings (buffers, TCP streams)
- replica selection
  - like file transfer, plus transfer time estimation
- remote data visualization
  - trading image quality for waiting time
  - based on network characteristics
- job waiting time estimation
  - GRMS scheduler needs available queues and expected waiting times
Delphoi Functionality

- Meta information
- Low-level resource and network information
- High-level network information
- Queueing information
- Logging
Meta Information

- String[] getActiveSites[]
- MetricInfo[] knownMetrics(String hostName)
  - MetricInfo contains metric name and parameters
  - Example: freeDiskSpace and /dev/hda

- These calls are essential to find out what is available in a Grid (VO).
String estimateMetric(
    String hostName,
    MetricInfo metric,
    String operation,  // min, max, mean
    Calendar startTime, // past, present, future
    Calendar endTime)

String[] estimateMetricForMultipleHosts()

String[] getRawMeasurementData()
Network Metrics

Currently supported: (according to GFD.023)

- path.delay.oneway
- path.delay.roundtrip
- path.bandwidth.available
- path.bandwidth.utilized
- path.bandwidth.capacity
- hoplist
TcpOptions estimateTcpOptions(
    String sourceHostName,
    String destinationHostName,
    long dataSize,
    String transferMethod, // e.g. GridFTP
    int maxTcpStreams,
    Calendar startTime)

Returns TCP Options (send buffer size and parallel streams) to optimize data transfer
double estimateTransferTime(
    String sourceHostName,
    String destinationHostName,
    long dataSize,
    String transferMethod,
    Calendar startTime)

double[] estimateTransferTimeOneToMany()
    // e.g., selecting scheduling target

double[] estimateTransferTimeManyToOne()
    // e.g., replica selection
void logDataTransfer(
    String source,
    String destination,
    long dataSize,
    String transferMethod,
    TcpOptions options,
    Calendar startTime,
    Calendar endTime)

- Application can give timing feedback
- Optional, only for improving predictions
Queueing Information

Queue[] getQueues()
   // host name, schedulers, queue names

QueueConf getQueueConf(Queue queue)
   // hosts, CPUs, limits,...

QueueWaitingTime getQueueWaitingTime(
   Queue queue,
   int jobSize, // number of CPUs
   Calendar startTime,
   Calendar endTime)

For prediction, job sizes are put in four categories:
single (1), small (2-4), medium (5-16), large (17+)
ResourceUtilization getResourceUtilization(
    Queue queue,
    Calendar startTime,
    Calendar endTime)

- Average number of free hosts available to a queue
- Measure for machine utilization
Logging

Applications and services can log messages
- using the Mercury monitor
- service/application, component, origin (user/host), severity, message

```java
String[] getLogs(String service,
    String component,
    String origin,
    int severity,
    Calendar startTime,
    Calendar endTime)
```

- Regular expression matching on parameters
The Delphoi Service

[Diagram of Delphoi Service architecture with various components and data flow]

- GRMS
- Data Movem.
- Replica Manager
- OGSA Frontend
- Web Service Frontend
- WSRF Frontend
- RMI Frontend
- Delphoi Information Logic
- Net Viz. Client
- Pythia
- Mercury
- PathRate
- PathChirp
- TopoMon
- Historic data storage
- Network measurements

Jason Maassen, Rob V. van Nieuwpoort, Thilo Kielmann, Kees Verstoep
Adaptive Grid Middleware (AGridM), Juan Les Pins, September 30, 2004
Example using Delphoi

File transfer optimization (from litchi.zib.de)

Delphoi automatically predicts the optimal GridFTP settings
Conclusions

- Applications need middleware to
  - get performance information
  - guide adaptation decisions
- The Delphoi service provides
  - a unified interface to various information sources
  - application-level information with prediction

available from www.gridlab.org/delphoi/