ABSTRACT

- Grid Middleware is founded upon a service-oriented architecture and Web Services technologies.
- However, Grid Middleware remains limited in a number of key areas.
- Our approach seeks to integrate sophisticated interaction types and advanced network services in order to better support advanced E-Science application scenarios.

Limitations of Grid Middleware (1)

- OGSA has emerged as a 2nd Generation Grid Middleware.
- Layers the Grid environment on top of existing web services platforms.
  - e.g. Apache Axis, Sun’s ONE and IBM’s WebSphere.

Limitations of Web Services

- Web Services are limited, compared to object-based middleware platforms (e.g. RM-ODP, CORBA, Java RMI, Enterprise JavaBeans, DCOM).
  - lack of generic (horizontal, or breadth-oriented) services. CORBA supports generic reusable services like fault tolerance, persistent state, automated logging, load-balancing, transactional object invocation, event distribution...
  - scalability and performance. EJB and the CORBA Component Model have sophisticated support for the automated activation/persist of stateful services, and natively support services that span multiple machines & networks. The web services platform developers haven’t focused on performance optimization compared to CORBA platform developers.

Over-reliance on SOAP as a communication engine

- Although flexible and general, SOAP is limited when used exclusively:
  - Unsuitable for large-volume scientific datasets. The use of XML as an on-the-wire data representation is highly demanding in terms of bandwidth, memory and processing cycles.
  - SOAP is not transparent to the application programmer, who often have to explicitly build and extract SOAP envelopes and message bodies, and perform manual marshaling and unmarshaling.
  - Limited support for different interaction types (e.g., choice of request-reply or one-way messages), it can never support a comprehensive range of interaction types (e.g. workflow, group, peer-to-peer, publish-subscribe etc.)

Configurable Middleware

- Web Services do not include recent results from advanced middleware research:
  - Highly configurable and dynamically reconfigurable middleware built upon the techniques of components and reflection, supports the custom building of platforms for PDAs, embedded systems and large-scale servers.
  - Support a range of programming APIs (e.g. CORBA, or web services, or APIs for media-streaming, or message-oriented middleware).
  - Alternative policies (e.g. security, replication, service (de)activation, priority- assigned invocation paths, thread scheduling) and components (e.g. protocols, buffer managers, loggers, debuggers, demultiplexers) are configured in or out at deploy-time, and reconfigured at run-time.

DCV applications exhibit properties such as: high levels of heterogeneity in terms of both networking and end-systems; real-time interactive collaboration employing multiple media-types; large scale, complexity and dynamic (re-)configuration; QoS-sensitivity, and adaptability to changes in environmental conditions.

Such applications over-stretch the state-of-the-art in existing Grid support. Our analysis is that current platforms have three major areas of deficiency in terms of advanced application support.

Integration with advanced network services

- One of the attractions of OGSA is its simple SOAP-based model.
- However, advanced applications require more sophisticated communications services in terms of:
  - QoS management,
  - different ‘interaction types’
    - RPC, asynchronous RPC, reliable/ unreliable messaging, publish-subscribe, tuple-space, peer-to-peer, gossip, event notification, request-response, service invocation, distributed voting or auction protocols, and various transactional styles

Architectural framework

- OGSA focuses on interoperability through the use of ‘ubiquitous’ Web protocols and associated abstractions (e.g. WSDL).
- However, this focus on interoperability needs to be complemented with a strong internal platform architecture that supports the integration of diverse system elements in terms of:
  - breadth (e.g. generic, ‘horizontal’ distributed services such as persistence, visualization, conferencing)
  - depth (e.g. underlying ‘vertical’ communication services in intimate contact with the network; and with other, end-system based, resources).

Complexity management

- Large, complex, and long-lived Grid applications need sophisticated management.
- The scale and complexity of such systems demand a self-managing or autonomic approach.
- It is crucial that such self-management is applicable to the architecture of the whole system including communication services.
- We argue that this implies an open and programmable approach to system construction.

Next generation Grid middleware must leverage the results of wider middleware research, retaining key web services characteristics (loose coupling, XML-based data structuring, reliance only on ubiquitous Internet standards), and in doing so maintain the availability of generic services, scalability and performance engineering, and the increased flexibility and configurability promised by the advanced middleware research.