1. Integration framework for Map Services and Geo-Science Grids

Geographic Information Systems (GIS) and specifically Map Services are very crucial for the spatial decision making and situation assessment. Map Services play the key role in the situation assessment through accessing and rendering the data to create comprehensible representations. Maps are composed of layers created with spatial data sets which are mostly kept in databases or in plain text files. Geo-Science applications are earth related scientific applications and using spatial and temporal data for their simulations.

There are several motivations for us. First of all, since Geo-Science applications require quick response times, the GIS services must enable accessing and processing these large data sets in a reasonable time period. Secondly, the distributed Geo-data provided by different vendors are heterogeneous. The users need to access and query these heterogeneous data remotely and seamlessly. Therefore, Geo-data should be interpreted and human comprehensible data representation should be created on the fly to reduce the complexity of the usage of these differently formatted Geo-data.

Therefore, we propose an architecture integrating Map Services with Geo-Science Grids, shown in **Figure 1**. The architecture utilizes a 3-layered structure. The bottom layer is to show the underlying maps such as satellite images. The middle layer is to display the features of the map by using vector data. For example, earthquake data or state boundaries can be illustrated in this layer. Finally, the third layer is to associate the previous two layers coming from Map Servers with Geo-Science outputs in the same bounding box. For more detailed information about the architecture look at the following section 2 and Figure 2.

Figure 1: output structure of the integration framework

There are some issues and challenges we will be investigating resulting from Geo-Science grids' requirements and spatial data characteristic. We group these research issues into two. These are (1) interoperability and extensibility issues and (2) performance issues. For the detailed information please see Section 3.

2. Components of the Architecture

WFS (implemented by Aydin G.) provide geographical information as GML feature collections. WMS interact with a Web Feature Service by submitting database queries encoded in OGC's Filter Encoding and in compliance with the OGC Common Query Language. WMS Enables visualizing, manipulating and analyzing geospatial data through maps shown on browser based interactive GUI. Map Servers typically compose maps in layers. Aggregator WMS is actually a WMS with some extensions providing high performance mapping services by using innovative pre-fetching, load-balancing and caching techniques (they will be explained in research issues section). User Portal and smart map+map animation tools enable end-users to interact with the proposed system. Resources are organized into projects and they are compliant with some resource schemas. The client portal is capable of supporting capabilities metadata of Web Map and Web Feature Services. The client portal provides both map-based and project based user interfaces. The two interfaces co-exist and are synchronized. Sci-Plotting service is developed in our lab. We define the service descriptions and interfaces in order to overlay Geo-Science applications' output as layers in 3-layer structured displays. Its core functionality is provided by Dislin scientific-data plotting libraries. We wrap them as Web Services and integrate into the proposed system as a layer providing service such as WMS. Job Manager (implemented by Gadgil H.) is actually a part of HPSearch project developed at CGL. It is simply a scripting technique for managing distributed workflows. Different Geo-Science applications (such as PI and Virtual California) require different set of parameters for the application users to trigger the job manager. This set of parameters and their order are defined earlier by the Job manager and user portal knows how to invoke it. WS-Contex's (implemented by Aktas M.) specifications defined by OASIS. When multiple Web services are used in combination, the ability to structure execution related data called context becomes important. WS-Context provides a definition, a structuring mechanism, and a software service definition for organizing and sharing context across multiple execution endpoints. We use it for implementation of asynchronous service runs. In our framework we take Geo-Science Grids as a black box. All the communication with this black box is handled by the job manager. As Geo-Science Grids we have been using ServoGrid projects.

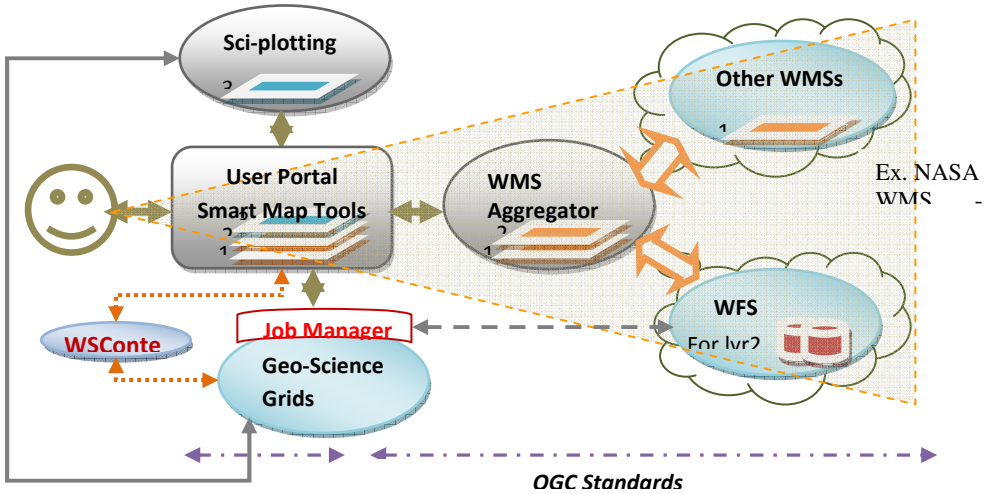

Figure 2: *Map Services and Geo-Science Grids integration framework. WFS and Job Manager (interface to Geo-Science Grids) are other CGL-Lab research projects*

3. Research Issues

***Interoperability and extensibility*:** Interoperability and extensibility are related to adding new services and data into the system seamlessly and enabling inter-service communication among the services within the system. In order to make the system interoperable and extensible, the services and data types should be standardized. Service standardization includes defining standard service functionalities and corresponding service interfaces. After defining the service standards we need to define common request formats and expected responses. Since each science domain has different needs and requirements (and attributes for the data), we initially focus only on GIS domain and possibly propose how GIS relates to other domains by generalizing standard services and data model we define in GIS.

We merge two important software worlds: GIS and Web Service Architectures. In order to achieve interoperability we use OGC and ISO/TC211 standard specifications in accordance with the Web Services principles. We use standards for online geo-services and common geo-data models. For online geo-services we develop WMS and WFS in accordance with OGC standard specifications. Inter-service communications are done based on capability exchanges and integration.

Regarding the data model, we use Geographic Markup Language (GML) for seamless data access and querying. Raw geo-data is kept in relational databases and provided by WFS in feature collections and features are encoded in GML. GML is a semi-structured XML based common data representation separating content from the presentation. Content part is related to data attributes which are queried interactively and presentation part is related to display of the data.

Performance: Geo-Science applications demand high-performance and high-rate data transfers and, require quick response times. Therefore, the GIS services must enable accessing and processing these large data sets in a reasonable time period. In most cases the amount of collected data reaches to an amount in the order of gigabytes or even terabytes; handling this data becomes a challenge for most users and organizations. Since the proposed GIS system is distributed architecture and due to the limited bandwidth and network speed it is impossible to make these kinds of large scale applications feasible. For the map animation applications, the situation becomes even much worse.

Since our proposed Information Grid architecture is a distributed system, the performance is limited by the network bandwidth. Distributed map services (WMS) and data sources (WFSs) are connected to each other through the network. Since network bandwidth issues are out of scope of this thesis, we work on improving the system performance through software solutions summarized as below.

Data Transfer: In the system there are only two types of data. One is vector data encoded in GML common data model and others are binary data in image types. The bottleneck is transferring vector data encoded in GML. We propose a streaming data transfer solution by using publish-subscribe based messaging middleware (NaradaBrokering).

Data handling in general: We propose using **pull parsing** to parse GML encoded data. XML Pull Parser is a recent development that demonstrates a different approach to XML parsing. It does not provide any support for validation. This is the main reason that it is much faster than its competitors. If you are sure

that data is completely valid and validation is done at the server side (as in our case) or in a database, then using XML Pull Parsing gives the highest performance results.

Load balancing is done through capability federation of services. The links to worker services are defined in capability metadata with the required parameters and attributes enabling remote invocation of worker nodes. The query partitioning and composition of the responses are other issues which are done by again using capability metadata and standard service interfaces and request structures.

Because of the characteristics of the spatial data (variable sized and un-evenly distributed) it is not easy to implement and get efficient performance results by using classic load balancing techniques. We propose a load balancing technique used together with the cached data extraction and rectangulation processes.

Regarding **caching**, we need to figure out caching algorithm based on what to cache (map images or GML data), how long to keep and how to utilize from it. We plan to cache map images and keep them until next request comes. We utilize from cached data by extracting overlapping region from the cached map and prepare requests for the remaining parts through *rectangulation* processes. Each rectangle obtained from the rectangulation process represents a worker node for the implementation of load balancing.

Finally, **pre-fetching** is getting the data before it is actually needed and keeping it in local server for the successive requests committed for the same data. Data fetching is done for the whole data by defining the queries in their widest ranges (no criteria). In our framework we use archived scientific data. Archived data does not change often. So, it is not reasonable transferring and rendering the same data again and again for every request coming from the different or even the same users. In order to solve this problem we plan to use pre-fetching.