Scientific applications are becoming highly dependent on Wide Area Network environments to access data and computing resources. Traditional database middleware systems have succeeded in integrating heterogeneous data sources but fail on scaling to WANs because they are focused on centralized architectures with static characteristics.

Our goal is to design a distributed database middleware system that takes into consideration the dynamic characteristics of each host on the WAN to efficiently run queries on it. We call this system NetTraveler.

Our goal with NetTraveler is to design an efficient query execution environment for distributed and parallel query execution that improve response time based on the characteristics of the hosts on the WAN.

Conventional DBMSs are usually modelled as a centralized architecture with several pipelined steps. One important step is the Query Optimizer (QO). A QO is the process that determines the needed actions to solve a query. Image below represents the most important aspects of a QO and its output.

Centralized Query Optimizer

Due to several reasons as cited on [5] centralized QO fails to scale to WAN environment. With that in mind we proposed the idea of a decentralized QO that could create plans that exploits parallel and distributed execution on WAN environment (see image below).

Decentralized Query Optimizer

Both the query optimizer and the query parallelization step would need information of the available resources on each host.

Our basic assumption is that data sources are replicated, we just need to select the better host for each step of the query execution. The result is shown on the next figure.

Centralized Query Optimizer

Current system provides an environment for remote query execution and the orchestration of the services to bring up the result for the client in the way explain by the graphic shown below.

Actual system support for parallel query execution of operation and static distribution of the load across participating sites.

A set of experiments will be performed to compare the number of queries per time unit between similar systems in order to validate our proposed solution.

Use of different Java technology for implementing our proposed architecture:

- Web Services, Axis
- Data access using JDBC, Hibernate

References