OPTIMIZED DATA MIGRATION WITHIN A MEDICAL GRID

Jared Christopherson and Chun-Hsi Huang
Department of Computer Science and Engineering, University of Connecticut, Storrs, CT 06269, U.S.A.

Keywords: Caching, Database Optimization, HIV Databases, Data Grid.

Abstract: This paper focuses on creating an intelligent, scalable system that vastly improves the speed and efficiency of looking up medical data. The system automatically and meaningfully organizes the distributed medical data to allow fastest access. Additionally, this research seeks to further improve on the concept of a distributed database service by introducing caching across servers as a means to optimize data retrieval time. Instead of looking to many individual sources, researchers would be able to access data from a single source, which is optimized on a per-region basis to ensure the shortest access times.

1 INTRODUCTION

The focal point of this research is to connect multiple databases across a grid in such a way that they appear as a single data repository while optimizing data flow on the back end. More specifically, the goal in this case deals with simplifying the process of acquiring data for health researchers. For example, when a researcher needs to find data on cancer statistics or HIV drug resistance, that researcher needs to spend time connecting individually to many different data sources and then must manually compile the data. The possible data sources include information from hospitals, other researchers' laboratories, and any institution hosting related databases. This process is clearly time-consuming and would benefit from a centralized location to acquire all the research data. This project will provide a web application for researchers to search a collection of databases from participating institutions and present the data as if they came from a single, specific source. Our initial research targets at the HIV sequence, resistance, immunology and vaccine trials databases (www.hiv.lanl.gov/content/index).

Back-end. The initial problem with linking databases is that each source usually stores their data in completely different fashions. Related technical articles may be found in (Huang, *et. al.,* 2005). Though some of the basic data supplied will be the same, table names and field names within each database will almost all be different – there is no published standard for how to store this type of data, so each individual site decides their preferred method. Getting cooperation from each hospital to restructure their database to a standard format would be impractical. To solve this problem, this project introduces the concept of *Master Templates*, which form the basis for the entire interface. Refer to articles included in (Huang, *et. al.,* 2008).

Master Templates. Each linked database is given a unique ID and then their login information is stored in the administrative database. A Master Template keeps each database's ID as a reference and maps the desired fields from that particular database into a "virtual" field created by the administrator. For example, across several databases, a field with data about the researcher might be known as *source*, *site*, *source_data*, *etc*. Additionally, these fields are likely in tables with all different names, such as *hiv_data*, *gene_info*, *etc*. This makes it very difficult for a grid system to link the data unless it has the information on how the fields should be linked. The administrator can simply choose to call this field in the master template "*Source*" and then regardless of all the different table and field names, the information will be correctly linked and displayed to the user. Thus, the grid-level administrator is responsible for appropriately linking the database fields as each new database is added to the system, but this is a one-time process that saves a huge amount of time for researchers in the future. Another benefit of Master Templates is that different templates can be provided to focus on different research subjects. The idea is that some medical

Christopherson J. and Huang C. (2010).
OPTIMIZED DATA MIGRATION WITHIN A MEDICAL GRID.
In *Proceedings of the Third International Conference on Health Informatics*, pages 481-484
DOI: 10.5220/0002759904810484

databases will contain information that pertains to more than one research area, so a Master Template would help organize the data into the different subjects. In this manner, multiple templates could be set up for a single database source to pull information pertaining to cancer, HIV, drug interactions, etc.

Display Templates. This project also introduces the *Display Templates* as a means to simplify data retrieval. A Master Template would contain every bit of information possible on a certain research subject, which is undoubtedly more than necessary for most users. Oftentimes research focuses just on a specific area of interest rather than simply the subject of "vaccine trials" as a whole. Thus, users may choose Display Templates, which act as a subset of Master Templates, to only retrieve needed information. All of the extraneous information would be ignored and users now have a convenient way to get information from a wide variety of sources pertaining only to his or her interests.

Interface. The front end provides a basic search system that returns records from all the databases matching the search parameters. Users have the ability to choose a Display Template to only focus on the specific data needed. The user has the option of viewing the datasets individually; otherwise the program will attempt to compile the data into a single table. The interfaces use AJAX, which is a Javascript technology that allows a web page to update data without refreshing the entire page. Thus, AJAX allows for individual database results to be pulled in as soon as they are returned so that a user doesn't have to wait for all the results from the slow servers before the page is loaded.

2 OPTIMIZATION

This research seeks to address the issue of data optimization so that researchers have the fastest access to data that they seek. Inter-region accesses to high-resolution image data such as MRI or CT-Scan images via a medical grid could incur a prolonged response time. The process would be much slower than if the data were stored on a more local server with sufficient bandwidth. In order to solve this problem, the web application tracks statistics on usage patterns and decides where to move the data so that it is best optimized for each grid-level data user. This project investigates the following options:

Moving to a Central Server - This option is impractical because it would require maintaining a single ultra-powerful system with prohibitive bandwidth costs. Additionally, this doesn't solve the problem of data access from different regions, because people in regions outside of the location of this main computer would still be at a disadvantage.

Moving Records around as they are Accessed - This option could run into legal issues with the actual deletion and moving of data between systems, since it could put the original owner of the data at a disadvantage in terms of access to the data.

Complete Caching - Clearly the perfect situation would be complete caching, where every institution has a complete cache of every other system. Unfortunately, this would be impractical because it assumes that every institution has the available space and bandwidth to host all the records.

Caching based on Usage - The best realistic solution would be to monitor usage patterns of each system and cache only the most highly requested databases. This solution provides the best and most realistic compromise to the problem.

What to Cache? Every time a query is performed, the system converts that user's IP address into a region ID to store statistics on searches from that region. The program keeps track of the region ID as well as the result count for each database that returns results from the query. In this manner, the system can build a list of the most highly accessed databases for each region. It is important to point out that the system is only concerned with databases that are outside of the region from which the search is being performed, since searches within the same region should already be relatively fast and are therefore considered already "optimized."

When the caching script is run, the program looks at each region individually and considers the result count for each database outside of that region. The script then converts the result count to a percentage of the overall relevant results, and inserts into a *Database Caching Queue* in order of highest result percentage. Thus, when the process is complete, every region has an ordered list of the most heavily accessed databases outside of that region.

Where to Cache? Since the question of what data needs to be cached has been answered, the remainder of the problem is simply an examination of where to cache the data. Several constraints on how to decide where to cache the data exist. The program must abide by these constraints while attempting to place data on a server that would result in the fastest response time for a particular set of users that need it. Here are the basic constraints:

allow_cache – Not every hospital/institution will want to volunteer to use their server for data caching, so this binary setting causes the system to take this server out of the caching determination.

supersite – This is a binary setting that gives special preference to a particular server that may be regarded as a more important institution or research facility. A supersite has more data cached locally and thus will have the fastest possible access.

bandwidth – This is a score of bandwidth available for a particular server.

cache_size – This setting tells the system how much space to allow for database and file caching.

All these settings are maintained by the grid-level administrator at the request of each institution. The variables are ordered in terms of importance for caching consideration, thus *allow_cache* and *supersite* trump any initial consideration of bandwidth or cache size on the other servers. The reason that *cache_size* comes after *bandwidth* in terms of importance is because otherwise there could be a server with a large amount of space but terrible bandwidth, and if data is placed on this system, nothing is being optimized in terms of speed.

Caching Script Conclusion. The script runs at an automatic interval determined by the system administrator. Alternatively, it may also be run manually. The caching process continues for each region with the program assigning data to servers with progressively lower *bandwidth* and *cache_size* scores until all the server space from that region is exhausted. Thus, when the process is complete, each region should have as many local copies of the most frequently requested databases as possible, and users will witness a very significant improvement in their data retrieval speeds, especially in retrieving locally cached high resolution image files. Finally, to avoid problems with data consistency, all the cached copies will be read-only so that they will always reflect the exact data on the original server.

3 EXPERIMENTAL PROCEDURE

We initially ran a simulation locally and assigned realistic values for data transfer speed. These values reflected the slow transfer rate from/to servers that were more distantly away. The values assumed the originating site was within the US region and ranged from transfer speeds of 450KB/s to a local server in the US to 25KB/s to a server in the Japan region. With the values in place, the script generated random requests to each of the 9 different HIV databases. The requests and transfer speeds from each database are shown in Figure 1 as follows.

Database ID	Region	Transfer Speed from US	Number of Results	External Region Result Percentage
DB1	US	400KB/s	1229	0.00%
DB2	US	450KB/s	1105	0.00%
DB3	JAPAN	25KB/s	856	19.10%
DB4	FRANCE	75KB/s	473	10.55%
DB5	SPAIN	85KB/s	764	17.03%
DB6	US	380KB/s	1439	0.00%
DB7	SPAIN	90KB/s	620	13.82%
DB8	JAPAN	30KB/s	1023	22.81%
DB9	JAPAN	20KB/s	749	16.70%

Figure 1: Pre-caching transfer speeds and simulated result counts.

Figure 2 illustrates the average transfer speed overall, average transfer speed to servers outside the US region, and transfer speed to the single most heavily trafficked database outside the US:

	Average Transfer Speed	Time to Transfer 3Mb File
Overall System	172.77KB/s	17.36 sec
External Regions	54.17KB/s	55.38 sec
Most Heavily Trafficked External DB (DB8)	30KB/s	100 sec

Figure 2: Speed averages and time indications for transferring large files.

4 RESULTS AND CONCLUSIONS

The caching script ran as previously described, determined the most heavily used databases in order of usage, and cached them on the servers identified by their IDs below. Note that for the purpose of simulation, DB2 was ranked highest based on bandwidth and has the ability to cache one database, DB1, Db6 could cache 3, 2 databases, respectively.

Database to be Cached	Caching Location
DB8	DB2
DB3	DB1
DB5	DB1
DB9	DB1
DB7	DB6
DB4	DB6

Figure 3: Database cache queue results.

The post-caching transfer speeds are shown below with the same result pattern as before.

Database ID	Region	Transfer Speed from US	Number of Results	External Region Result Percentage
DB1	US	400KB/s	1229	0.00%
DB2	US	450KB/s	1105	0.00%
DB3	**CACHED (DB1)**	**400KB/s**	856	**0.00%**
DB4	**CACHED (DB6)**	**380KB/s**	473	**0.00%**
DB5	**CACHED (DB1)**	**400KB/s**	764	**0.00%**
DB6	US	380KB/s	1439	0.00%
DB7	**CACHED (DB6)**	**380KB/s**	620	**0.00%**
DB8	**CACHED (DB2)**	**450KB/s**	1023	**0.00%**
DB9	**CACHED (DB1)**	**400KB/s**	749	**0.00%**

Figure 4: Post-caching transfer speeds and simulated result counts.

Since there was enough space on the local databases to cache the external databases, the external region result percentages have all changed to 0. Examining the new system speeds based on the same metrics as before gives the following:

	Average Transfer Speed	Time to Transfer 3MbFile	Comparison to Pre-Cache Time
Overall System	404.44KB/s	7.42 sec	2.34x faster
External Regions	401.67KB/s	7.47 sec	7.41x faster
MostHeavily Trafficked External DB (DB8)	450KB/s	6.67 sec	15.00x faster

Figure 5: Post-caching speed averages and time indications for transferring large files.

Comparison of the results shows that the caching process in this simulation has greatly increased data access times. On average, transfer speed is over twice as fast and results from databases outside of the US are accessible over seven times faster. The most remarkable difference is evident with the most heavily trafficked database outside of the US. Since this database received the most, clearly it is of significant research importance at the time of running the caching script. Afterward the caching is completed, researchers in this simulation would be able to pull data from this database fifteen times faster than before caching.

5 FUTURE WORK

Additional work to the current system spans several areas. The current system has been designed to connect only to MySQL databases. However, institutions participating in a medical grid VO (Virtual Organization) may use a wide variety of database technologies to maintain their local databases. To account for this, the ongoing development incorporates a *Database Abstraction Layer* (DAL) added to the Master Template system to allow it to map to any kind of supported databases, such as MySQL, MS SQL, Access, *etc*.

The caching system may also be improved in that it caches an entire database any time interval the caching script is set to. While the grid-level administrator may manually notify the system to run the caching script, there is currently no way to tell when significant changes have been made to the source database. Hence, users might not have access to all of the newest data until the end of the time interval. With a script that keeps track of major changes, all caches could be updated automatically when changes are made, avoiding the bandwidth waste while re-caching unchanged databases.

Finally, as the system compiles data from many different sources, it is likely that a user could run into overlapping data of the same record sets. The system may be designed to automatically clear up redundant records as they are presented.

REFERENCES

Huang, C.-H., Konagaya, A., Lanza, V. and Sloot, P. 2008. Biomedical Computations on the Grid. *IEEE Transactions on Information Technology and Biomedicine, Vol 12, No 2, 2008, pp. 133-137.*

Huang, C.-H., Lanza, V., Rajasekaran, S. And Dubitzky, W. 2005. HealthGrid – Bridging Life Science and Information Technology. *Journal of Clinical Monitoring and Computing, Springer, Vol 19, No 4-5, pp. 259-262.*