Distributed Processing of Large BioMedical 3D Images

Konstantinos Liakos[1,2], Albert Burger[1,2], and Richard Baldock[2]

[1] Heriot-Watt University, School of Mathematical and Computer Sciences, Ricarton Campus, EH14 4S, Edinburgh, Scotland, UK
[2] MRC Human Genetics Unit, Western Hospital, Crewe Road, EH4 2XU, Edinburgh, UK

Abstract. The Human Genetics Unit (HGU) of the Medical Research Council (MRC) in Edinburgh has developed the Edinburgh Mouse Atlas, a spatial temporal framework to store and analyze biological data including 3D images that relate to mouse embryo development. The purpose of the system is the analysis and querying of complex spatial patterns in particular the patterns of gene activity during embryo development. The framework holds large 3D grey level images and is implemented in part as an object-oriented database. In this paper we propose a layered architecture, based on the mediator approach, for the design of a transparent, and scalable distributed system which can process objects that can exceed 1GB in size. The system's data are distributed and/or declustered, across a number of image servers and are processed by specialized mediators.

1 Introduction

The Edinburgh Mouse Atlas [1, 2] is a digital atlas of mouse development, created at the MRC Human Genetics Unit (HGU), Edinburgh, to store, analyze and access mouse embryo development. The stored objects contain 3D images that correspond to conventional histological sections as viewed under the microscope and can be digitally re-sectioned to provide new views to match any arbitrary section of the experimental embryos [2]. The volume of the processed 3D objects can exceed 1GB and typical lab based machines fail to efficiently browse such large bio-medical image reconstructions, particularly since the users may wish to access more than one reconstruction at a time. This paper addresses that problem by proposing a layered distributed system design that provides scalable and transparent access to the image data; by transparency we mean that the system hides from the end user the connection among the different servers and the distribution of the data.

The proposed architecture has been influenced by the mediator approach. Mediators were first described by Wiederfold [3] to provide a coherent solution to the integration of heterogeneous and homogenous data by "*abstracting, reducing, merging and simplifying them*". In more detail mediators are "*modules occupying* *an explicit active layer between the user applications and the data resources. They are a software module that exploits encoded knowledge about certain sets or subsets of data to create information for a higher layer of application*" [3]. They encapsulate semantic knowledge about a number of distributed sources providing to the user-applications a unique representation of the distributed database context.

The middleware initially adopted for our purposes is the Common Object Request Broker Architecture (CORBA) [4, 5]. The CORBA framework provides the means to develop open, flexible and scalable distributed applications that can be easily maintained and updated. Data integration can be partially resolved via the use of the Interface Definition Language (IDL) that separates the data model from its implementation. In the past CORBA has been proposed as an efficient middleware solution for bioinformatics projects [6, 7, 8, 9]. In addition, the European Bioinformatics Institute (EBI) [10] has adopted CORBA as a middleware for a number of its bioinformatics systems. A competing technology to CORBA, also adopted by EBI, and gained a lot of attention in general, is Web Services[11]. However, Web Services are less efficient than CORBA/IIOP over slow connected networks [12]. As a consequence they currently cannot give optimum solutions to applications that require fast processing and transmission of large amounts of data. It should be noted though, that our proposed design is independent of the middleware platform. Web services are evolving rapidly and become the preferred option in many bioinformatics applications.

A property of this application is that a query on the data requests virtual objects. These objects are analogous to a view in database terms that represent 2D section images that are computed at run time from the original 3D voxel models stored in the database.

Our layered design is a modification of the mediator approach and is based on the requirement for easy re-configuration for performance reasons. Simple client-server designs such as that implemented in [2] for a genome-mapping prototype are inadequate to handle the requirements derived from large voxel image files. Although smaller size voxel images can be efficiently processed, larger images result in an unacceptably slow response, due to memory paging and CPU processing time. Furthermore integration of additional object resources becomes impractical. The volume of our current as well as our anticipated future objects, in addition to the requirement of providing very fast response times introduce the necessity of distributing the cost of processing by declustering biological images in order to provide a scalable solution, minimize the cost of the overall query response time, and make an optimum usage of the available hardware resources. Our design adopts an n-tier solution by distributing the cost of processing to a number of image servers. Such a task is accomplished by declustering and distributing the image data across different image servers, processing them in parallel. The image server processing is hidden by the use of one or more mediator layers, which are responsible to provide transparent access and to monitor image servers so that user requests are directed appropriately. In addition, mediators are designed to provide other services, such as query processing-decomposition and re-assembly, query optimization and user-behavior prediction or “look-ahead”. This latter service enables pre-computation of predicted requests in order to accelerate the response time of the system. Finally an important aspect of our design is its ability to vary the number mediators and allow dynamically re-configure the system to optimize performance.

The emphasis of this paper is on the description and the efficiency of the prototype system to illustrate the advantages gained by data distribution and parallel processing. Issues such as the study of optimum declustering and placement approaches and the evaluation of particular prediction techniques will be reported elsewhere.

The remainder of this paper is organized as follows: Section 2 provides a brief discussion of the current Mouse Atlas system covering in more detail image processing issues and the notion of a virtual object. Section 3 provides a detailed description of the proposed distributed architecture, section 4 presents the query processing design and in section 5 some initial performance results to process large reconstructions are provided. Finally section 6 discusses future issues and concludes the paper.

2 Mouse Atlas

2.1 The Mouse Atlas System

The MRC Human Genetics Unit, Edinburgh, has developed the Edinburgh Mouse Atlas [1, 2] based on an object-oriented architecture in order to store and analyze biological data, including 3D images that relate to mouse embryo development [2]. The embryo framework for the database is represented as a set of voxel models (3D images), initially at each development stage defined by Theiler [13], but with the possibility of extension to finer time-steps especially at earlier stages. The voxel models correspond to conventional histological sections as viewed under the microscope and can be digitally re-sectioned to provide new views to match any arbitrary section of an experimental embryo.

The purpose of the system is the analysis and querying of complex spatial patterns, in particular the patterns of gene activity during embryo development. The underlying image processing and manipulation uses the Woolz image-processing library [1, 14, 15] that was developed for automated microscope slide scanning and is very efficient for binary set and morphological operations. The users navigate in the 3D space via the use of user interface components that correspond to particular viewing parameters, to define 2D sections (fig.1) at any orientation. At any given time only one such component can alter its value to generate a new query. For efficient browsing it is necessary to get the entire voxel image into the main memory so disk accesses are avoided. For very large reconstructions, i.e. images of $1-3GB$, this is impractical for typical laboratory based machines, particularly since users may wish to access many such reconstructions concurrently. Even with the entire image in memory most CPUs will be slow. While CPU and memory specifications will be steadily improved image processing

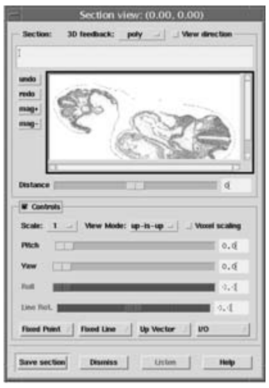

Fig. 1. GUI to process 3D Woolz images. Specialized components enable the rotation within the 3D space. The requested sections can be scaled, magnified and saved to disk

requirements are also expected to increase; e.g. future image volumes might well reach $10 - 100GB$.

There are many aspects of the Mouse Atlas, which are not directly relevant to the discussion that follows and therefore omitted from this paper. The interested reader is referred to [1, 2, 14, 15].

2.2 Virtual Objects and Woolz Image Processing

The Woolz library that has been developed by the MRC HGU performs all image-processing operations of this system. Woolz uses an interval coding data structure so that only grey-values corresponding to foreground regions (the embryo) are held in memory. For this type of image this can result in $30 - 50\%$ reduction in memory footprint in comparison with conventional image formats. While detailed information on various other aspects can be found in previous publications [1, 2, 14, 15], our emphasis here is on the efficient computation of requested 2D sections. These are *virtual* objects, objects that are not saved in the data sources, but are computed during run-time by Woolz library functions associated with the original 3D voxel data. This is analogous to a view in database terms. The rotation within the original 3D space that results in the generation of a section view is determined as follows.

Given an original coordinate $r = (x, y, z)^T$, the viewing plane (fig.2) is defined as a plane of constant z in the new coordinates $r' = (x', y', z')^T$, i.e. the new z-axis is the *line-of-sight*. This axis is fully determined by defining a single fixed-point f that is the new coordinate origin, and a 3D rotation. The actual view plane is then defined to be perpendicular to this axis and is determined by a scalar distance parameter d along the new axis. In this way the transformation between the original r and viewing coordinates, r', is determined by a 3D rotation and translation with the viewing plane defined as a plane of constant $z' = d$. A 3D rotation can be defined in terms of Eulerian angles [16] with full details given

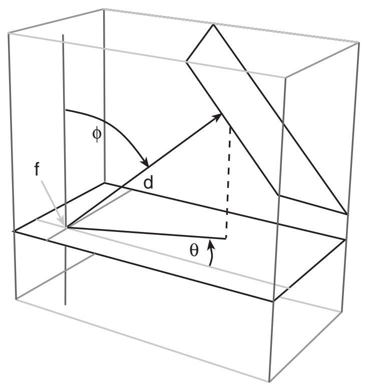

Fig. 2. The viewing plane is defined to be perpendicular to the viewing direction given by angles *phi* and *theta*. The actual plane is distance d from the fixed-point f. For the user interface *phi* is termed *pitch* and *theta* is *yaw*

in [2]. These mathematical details are not essential to the understanding of the rest of the paper, but highlight the point that the computation of a 2D section requested by a user query involves the retrieval of 3D data from the database as well as some image processing on that data.

3 The System's Architecture

The proposed design consists of four main components: client, directory, mediator and image servers (fig.3). The client consists of a GUI that provides the stage selection menus, controls for section parameters and is responsible for the generation of user queries while the mediators provide a transparent view onto the image server data sources. The mediators encapsulate the knowledge of the data sources (metadata), such as the objects which are stored in addition to their structural properties and the image servers where objects can be found, providing transparent access to the image data. Image servers provide direct access to the underlying image storage e.g. files and databases. It is only an image server that loads and directly queries a Woolz object. Due to the volume of the 3D image data, declustering is expected to be introduced. By declustering we mean the process of cutting an original 3D voxel into smaller dimensions placing them at different sites. A key difference in the concept of declustering between our approach and the one that is normally in use is that our declustering performance relates to the CPU speed up that can be gained; current declustering schemes are concerned with the disk IO speed up [17, 18]. Our system assumes that for each request the declustered 3D image data can be loaded into the main memory of each image server. Finally directory servers enable access to mediators and image servers by providing their connection details. Note that clients can only acquire information about a mediator while a mediator can acquire connection details related both to other mediators and image servers.

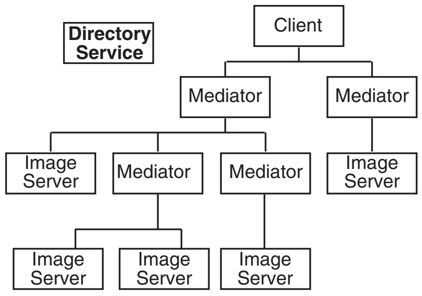

Fig. 3. Layered design based on the system's requirements

A mediator's task is to provide a unified view and transparent access to the declustered distributed Woolz objects, monitoring of the user access patterns and prediction of future requests so as to pre-compute related sections. The successful completion of these tasks depends on a metadata model that links image servers to Woolz image objects and Woolz objects to object properties, such as their structure and size. An image server on the other hand provides a logical view onto the Woolz data and a Woolz query-processing model to access them. The object oriented database in which Woolz are saved can be only accessed by appropriate image servers.

In a mediator/wrapper approach (fig.4) a mediator encapsulates a global data model to serve a number of heterogeneous data sources. Such an approach has been adopted by many distributed systems such as DISCO [19], GARLIC [20] and Informia [21]. In comparison to other bioinformatics implementations where the mediator approach is used to integrate heterogeneous data sources (Raj and Ishii [8], and Kemp et al [9]) our design seems similar because it also encapsulates a schema of its integrated image servers, however in contrast to most of them, where a centralized mediator serves a number of data sources, our mediators are fully scalable (fig.3). A hierarchy of mediators can be dynamically created according to the performance needs of the system at run time. As a consequence our design can result in both centralized and distributed topologies depending on performance requirements. More precisely a mediator might not only control a number of image servers but also a number of mediators as well. Hence a mediator's schema might be based on the schema of its lower-level mediators. The concept of such a composable approach is similar to AMOS II [22, 23], a lightweight OODBMS, where mediators regard interconnecting mediators as data sources. However the purpose of AMOS II and its implementation are different from ours. AMOS II is a federated multi-database system that combines heterogeneous data sources and views. On the other hand our system's main aim is to reduce the time needed to fetch, process and display large image objects. Because the processed Woolz objects are homogeneous, the emphasis is not given to the integration of diverse data sources. The mediator approach is used to distribute the cost of processing, exploit data parallelism and enable processing of very large reconstructions. Apart from that our proposal is based on standard

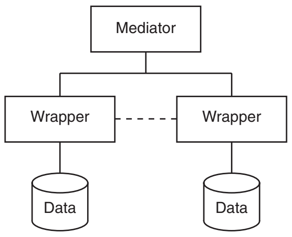

Fig. 4. The Mediator/Wrapper approach

and flexible solutions that facilitate scalability and maintenance, and is language and platform independent.

Furthermore our mediators perform additional tasks such as the monitoring of the user access patterns and the pre-computation of future requested sections so as to improve the overall performance of the system. These facilities can be seen as an extension of our query processing mechanism that is described in the following section. Details of this speculative computation model are beyond the scope of this paper and will be presented elsewhere. Under speculative pre-computation we mean that a mediator requests image servers to precompute possible future requested sections. Until the actual user request is received those predicted sections are not transmitted from the image servers to the mediator, where they are further processed, minimizing the cost of processing predicted sections that are not eventually needed.

The client component consists of a GUI that provides the tools to query and process Woolz image objects. Initially clients are unaware about the connection details of the mediators they are connected to. Such information is acquired from the directory server (fig.3). Another important issue is that clients are not aware of the underlying Woolz technology [2]. Their only intelligence relates to the display of the 2D sections, which are acquired as bitmap arrays, and their understanding of the 3D bounding box of the saved Woolz objects. An important property of the technology underlying the GUI is that it results in asking only the visible part of every reconstruction. As a consequence only a small part of the whole virtual object is requested, minimizing the data that needs to be transmitted over the net. The client GUI is intended to be as light as possible.

We are aware of the overhead that a layered approach might introduce. At this prototype level, our emphasis was on functionality. A performance study of the impact of multiple mediators is planned. The focal measures of such a study will be the tradeoff between the network and processing overhead and the potential improvement of the overall response time.

4 Query Processing

Kossman [24] has extensively described query processing optimization techniques. Assuming that Woolz internal image processing is optimal, our efforts

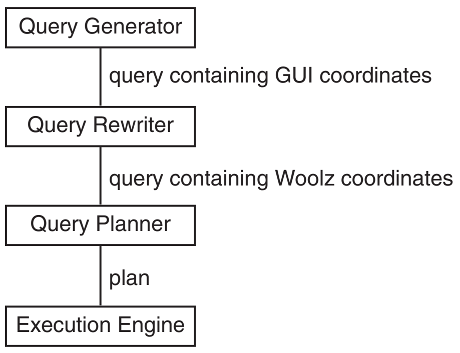
Query Generator
query containing GUI coordinates
Query Rewriter
query containing Woolz coordinates
Query Planner
plan
Execution Engine

Fig. 5. Query Processing Architecture

are focused on accessing the Woolz data sources as fast as possible producing the adequate query for them. Our query processing architecture comprises of a query generator, a query rewriter, a query planner and a query execution module (fig.5). A query generator is found on the client side and involves the generation of a Woolz query based on the visible coordinate system of the GUI. The query rewriter module transforms the query from GUI coordinates into Woolz coordinates. The generation of a query can be also introduced on the mediator's side as a consequence of the pre-computation process algorithm that monitors user activity. The query planner, that includes a query rewriter module, selects only the data servers that can respond positively to a particular query. To optimize system performance only the image servers that can respond to queries are sent modified user requests. Only after the data servers are selected, a query-rewriting process starts so as to split the initial Woolz query and request only the subregion to which each data server can respond. The query execution model that is found both in the client and mediator is used to transform either a user or a mediator query into a compatible IDL query. On the other hand query execution on the image server side relates to the use of Woolz libraries so as to physically execute a query.

5 Experiments

To test our approach, a prototype system has been developed and used for a number of experiments. The experiments were run on a SUN Netra array, which includes 10 servers running SOLARIS 8. Each server has the same capabilities and the interconnection between them is provided via the use of dedicated switches whose speed is 100Mbps. Every server has 500MB of RAM, 500 MHz of UltraSPARC-II CPU and 256 Kbytes of e-cache. The developed programs were written according to the requirements of the system's design in Java [25]. A benchmark was developed that requests all 2D sections for each reconstruction for a given viewing orientation. Only after a client acquires a requested 2D section, a generation of a new query takes place. The window size is set up to

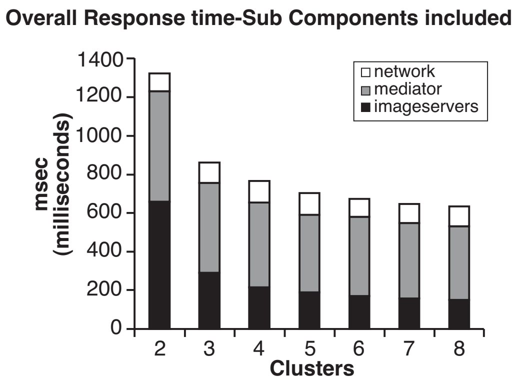

Fig. 6. Detailed analysis of the times spend throughout the experiment. Declustering and parallelism result in decreasing the system's average response time when 8 clusters are used

800x400 pixels while a centralized mediator is used throughout the experiments. Every image server responds to a mediator query by sending only the part of the section that it can generate. The requested parts are determined by the window size of the clients GUI and the section parameters.

The aim of the experiments was to establish the capability of our distributed design to handle large reconstructions and to understand and model the effects of data declustering and parallelism. For these tests the 3D reconstruction of a mouse embryo at its developmental stage Theiler stage 20 (TS20), a voxel image object of size 685.5 MB, was used. Parallelism is achieved by the use multiple processes communicating using CORBA and within a single process, e.g. the mediator or image server by the use of java programming threads [25]. Data declustering is the process of partitioning an object into a number of fragments.

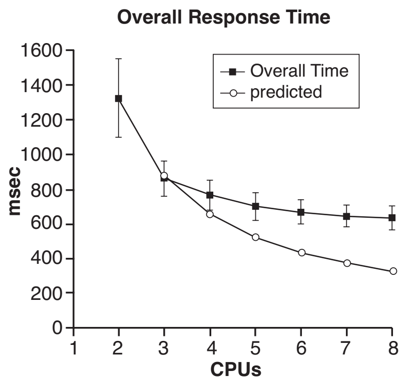

Fig. 7. Overall response time. The average response time of the system is reduced by the use of 8 clusters

The first experiment examines the overall advantage gained by the introduction of data parallelism and declustering. TS20 was cut into a number of clusters $(2-8)$ of approximately equal size and placed in different locations over the network. The benchmark described previously was used to measure the system's performance. In this case the declustering was chosen so that each image server would respond so that the overall potential benefit of the parallelism could be established.

As expected the system's performance greatly improved through the use of the parallel processing design. In fig.6 an estimation of the time spent at each component of the system is given, while in fig.7 a more statistical analysis of the system's performance is provided. The displayed curve represents optimal performance assuming linear speedup with respect to the two-computer case i.e.: $\frac{2AVG(2)}{clusterNo}$ where $clusterNo$ is the number of clusters and $AVG(2)$ is the average

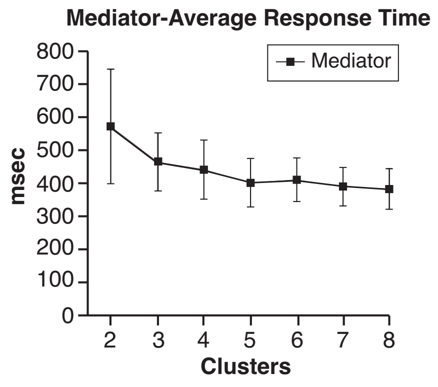

Fig. 8. Estimated Time spent on the mediator

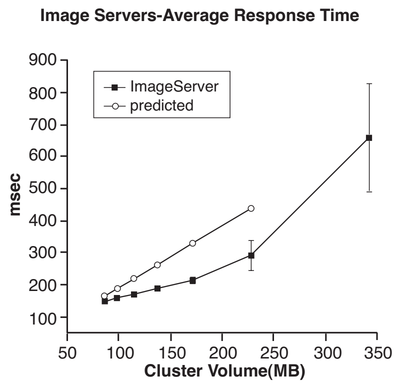

Fig. 9. Declustering effects. The process in which object were cut along their $Y-axis$ and placed in different image servers results in improving the image server's response times when the number of clusters increases

response time when 2 clusters are used.The difference between the curve and the measured points presumably reflects the performance of our declustering approach (fig.9), and the cost of processing many clusters simultaneously in the mediator (fig.8). The most important module of our experiment is the mediator since it monitors, directs and finally processes user requests. The use of one CPU in our mediator server results in similar performance results for all the experiments that were carried out (fig.8). The multi-threaded model adopted for the mediator can further improve its performance results provided that a multi CPU machine will be used. Under the current implementation java threads add an additional cost that is however negligible. The derived variations under the use of a small number of clusters are a consequence on variations on the response times of the individual image servers.

Finally the effect of declustering is studied by examining the performance of the image servers when the volume of their data sources decreases. The declustering approach that was followed resulted in objects whose volume varied from 342.75 MB (2 Clusters) to 85.69 MB (8 Clusters) As expected (fig.9) the smaller the image volume the faster a section is generated and sent back to the mediator for further process. The low standard errors and the average response time results are considered very efficient and reveal the response times that can be achieved for larger volume image files that are declustered to a larger number of image servers. By examining the predicted line of fig.9 it is noticeable that it performs worse than the actual results obtained from the experiment. Such behavior is explained by the poor results obtained by the usage of 2 clusters. To be more precise, in the case of the 2 clusters, the processing of them may exceed the available main memory, resulting in such a poor performance.

While both the effects of declustering and parallelism have been described in detail, a last experiment is carried out to establish the capability of our design to handle very large reconstructions. More precisely TS20 is scaled by a factor of 2 across its $X-Y-Z$ axis resulting in an object sizing 5.5 GB. As previously, the objects are declustered across their Y coordinate system, producing 8 sub objects of roughly the same volume, that are distributed across 8 image servers. In addition Theiler Stage 16 (TS16) is also scaled accordingly resulting in an object sizing 1.47 GB. The window size remained the same ($400x800$ pixels) while our benchmark is slightly changed in order to request the same amount of sections we had requested in the previous experiments.

The results of the experiment can be seen in fig.10. As expected, the response time of the system gets slower when the volume of the requested objects get larger. The gap between the image server response time and the system response time represent the mediator time in addition to the network and idle time. By better examining the graph it can be noticed that the mediator's time is slightly bigger when the original TS20 is under use. Because of the volume of the scaled TS16 and TS20 it is impossible to make maximum usage of the image servers. As a consequence when the scaled TS16 is used, 7 image servers respond in parallel while only 4 of them respond in parallel when the scaled TS20 is requested. As a result there is an overhead related to network requests and thread processing that

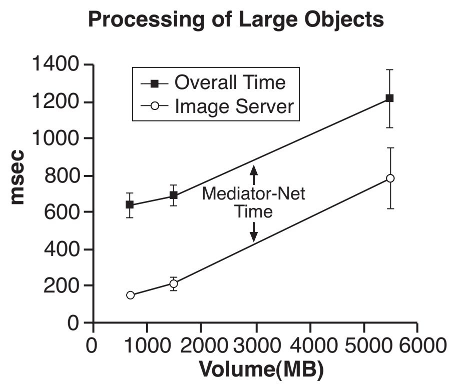

Fig. 10. The larger the volume and the dimensions of the objects are, the slowest their response time. The gap between the 2 diagrams represents the mediator time in addition to network and idle time

is not provided in these experiments. This overhead corresponds to the additional requests and threads that are generated when all image servers are involved in processing a user request, that result in increasing the cost of processing at the mediator side. All experiments are expected to reveal almost the same response time at the mediator side, given the fact that the same window size is used and the volume of the requested sections is roughly the same. Moreover such a feature also proposes further experimentations to define data declustering optimization techniques.

6 Future Work and Conclusions

This paper discussed the necessity of a distributed approach to efficiently process large biomedical 3D image data. In the Edinburgh Mouse Atlas, user queries do not directly correspond to the 3D objects saved in the database but rather to 2D sections, which need to be calculated at run-time from the 3D image data. This creates an interesting context in which new analysis related to different declustering schemes is required.

Some preliminary experiments reveal that user navigation patterns strongly affect the optimum cluster generation procedure and hence further details concerning optimum declustering and placement approaches is required. The required analysis will examine different placement approaches to identify the one that minimizes the cost of distribution and optimizes the performance of the system.

In addition our mediators will be enhanced with prediction capabilities to pre-compute future requested queries based on the monitoring of the user access patterns. Initial studies here suggest that time series prediction approaches such as the Autoregressive Integrated Moving Average (ARIMA) [26, 27] and the Exponential Weighed Moving Average (EWMA) [28] adapt fast and efficiently to changing user behaviors. However the cost of the algorithms and the improvement of the system need to be evaluated. Apart from that a unified prediction policy should be suggested to decide how, when and where precomputation will be introduced. Under this concept the identification and the properties of different user access patterns are of great importance, so as to apply prediction only when needed.

In summary, this paper has presented a layered design to handle large bioinformatics 3D images, which represent different stages of mouse embryo development. Such a design is based on the mediator approach and exploits the advantages gained by parallel processing and data distribution.The work thus far has shown the validity of the approach and already provides an acceptable response time for user interaction with a very large-scale image (5.5 GB). Data distribution and parallel processing are essential not only to provide faster response times but also to enable processing of large objects unable to fit into the main memory of a single machine. Such features have been presented not only by processing a very large scale biomedical image but also by examining the results of declustering and data distribution for a smaller scale object (685.5 MB) in which the overall response time has been decreased more than 100% when all 8 available distributed CPUs have been used.

Acknowledgments

The MRC HGU in Edinburgh has sponsored the presented work.

References

1. R. A. Baldock, F. J. Verbeek and J. L. Vonesch, 3-D Reconstructions for graphical databases of gene expression, Seminars in Cell and Developmental Biology, 1997, pp. 499-507.
2. R. A Baldock, C. Dubreuil, B. Hill and D. Davidson, The Edinburgh Mouse Atlas: Basic Structure and Informatics, Bioinformatics Databases and Systems, Ed. S Levotsky (Kluwer Academic Press, 1999), pp. 102-115.
3. G. Wiederhold, Mediators in the Architecture of Future Information Systems, IEEE Computer, vol 25, no 3, 1992 pp: 38-49.
4. Object Management Group, OMG, www.omg.org .
5. R. Orfali, D.Harkey, Client/Server Programming with Java and CORBA, Second Edition, Wiley Computer Publishing, ISBN: 0-471-24578-X,1998.
6. A. Spiridou, A View System for CORBA-Wrapped Data Source, European Bioinformatics Institut, Proceedings of the IEEE Advances in Digital Libraries 2000 (ADL2000).
7. J. Hu, C. Mungall, D. Nicholson and A. L. Archibald, Design and implementation of a CORBA-based genome mapping system prototype, Bioinformatics, ,2nd ed, vol.14, 1998,pp. 112-120.
8. P. Raj, N Ishii, Interoperability of Biological Databases by CORBA, Nagoya Institute of Technology, Proceedings of the 1999 International Conference on Information Intelligence and Systems, March 31 April 03, Rockville, Maryland, 1999.

9. J. Graham,L. Kemp, N. Angelopoulos, P. M.D.Gray, A schema-based approach to building a bioinformatics database federation, Proceedings of the IEEE International Symposium on Bio-Informatics and Biomedical Engineering (BIBE 2000),Arlilngton,Virginia, USA, November 08-10, 2000.
10. European Bioinformatics Institute, http://www.ebi.ac.uk/ .
11. World wide World Consortium, www.w3.org .
12. D. C. Schmidt, S. Vinoski, Objet Interconnections: CORBA and XML-Part3: SOAP and Web Servises, C/C++ Users Journal, www.cuj.com
13. K. Theiler, The House Mouse: Atlas of Embryonic Development, Springer-Verlag, 1989.
14. R. A. Baldock, J. Bard, M. Kaufman, D. Davidson, A real mouse for your computer. BioEssays, vol. 14 no 7, 1992, pp. 501-503.
15. M. Ringwald, R. A. Baldock, j. Bard, M. Kaufman, J. T. Eppig, J. E. Richardson, J. H. Nadeau, D. Davidson, A database for mouse development, Science 265, 1994, pp. 2033-2034.
16. H. Goldstein, Classical Mechanic, Addison-Wesley, Reading, MA, 2nd ed., 1950.
17. Y. Zhou, S. Shekhar, M. Coyle,Disk Allocation Methods for Parallelizing Grid Files, ICDE 1994, 1994, pp 243-252
18. C. Chen, R. Sinha, R. Bhatia, Efficient Disk Allocation Schemes for Parallel Retrieval of Multidimensional Grid Data, Thirteenth International Conference on Scientific and Statistical Database Management, Fairfax, Virginia, July 18 - 20, 2001
19. A. Tomasic, L. Raschid, P. Valduriez, Scaling access to heterogeneous data sources with disco, IEEE Transactions on Knowledge and Data Engineering, vol 10, no 5 ,September/October 1998, pp. 808-823.
20. L. Haas, D. Kossmann, E. L. Wimmers, J. Yang, Optimizing Queries across Diverse Data Sources, Proceedings of the Twenty-third International Conference on Very Large Databases, Athens Greece, August 1999.
21. M. L. Barja, T. A. Bratvold, J. Myllymki, G. Sonnenberger, Informia, A Mediator for Integrated Access to Heterogeneous Information Sources, In Gardarin, French, Pissinou, Makki, Bouganim (editors), Proceedings of the Conference on Information and Knowledge Management CIKM'98, Washington DC, USA, November 3-7, 1998, pages 234-241, ACM Press, 1998.
22. V. Josifovski, T. Katchaounov, T. Risch, Optimizing Queries in Distributed and Composable Mediators, Proceedings of the Fourth IECIS International Conference on Cooperative Information Systems, Edinburgh, Scotland, September 02-04 1999.
23. T.Risch, V.Josifovski, Distributed Data Integration by Object-Oriented Mediator Servers, Concurrency and Computation: Practice and Experience J. 13(11), John Wiley & Sons, September, 2001.
24. D. Kossman, The State of the Art in Distributed Query Processing, ACM Computing Surveys, September 2000.
25. Java, www.java.sun.com
26. C. Chatfield, Time Series Forecasting, Charman & Hall/CRC, ISBN 1-58488-063-5, 2001.
27. G. E. P. Box, G. M. Jenkins, Time Series Analysis: forecasting and control, ISBN: 0-8162-1104-3, 1976.
28. C. D. Lewis, Industrial Business Forecasting Methods, ISBN: 0408005599