

Migrating data integration agents for genomics

Ela Hunt, Karen Renaud, Richard Sinnott and Joanna Jakubowska
Department of Computing Science
University of Glasgow, Glasgow, G12 8QQ, UK
{*ela,karen,ros,asia*}@dcs.gla.ac.uk

April 22, 2005

Abstract

In order to address the needs of data visualization software we propose that e-Science should support the dynamic load balancing of data integration agents and that this mechanism could be supported by grids of collaborating computing infrastructures. Examples will be given of a service tailored along these lines to support visualization client software. The new service will better support the information requirements of visualization. The proposed solution, called **e-Aglets**, relies on a combination of techniques including web services, agents, load balancing, migration, and performance measurement.

1 Introduction

Data-hungry visualization applications for e-Science encounter severe limitations in access to large data sets. Both web servers and web services are known to suffer from temporary failures, and the outcome of such failures is that client visualization software using such services cannot be used when data servers refuse to cooperate. Moreover, as most web servers and services are optimised to serve many concurrent users who place small data requests, there is a need for a different approach to data provision for visualization. The approach we propose is focused on data provision for bulk access, and is not satisfied by current web servers. One of the possible solutions is to design two-part systems: one part being the visualization client, and the other an agent which can place itself somewhere in the local grid area, and provide the visualization component with a data service which is independent of the original server. This agent technology should be capable of self-replication and of relocation within a grid to the most suitable location near the visualization user, and would provide data replicas, and higher bandwidth for larger quantities of data.

2 Rationale

Our visualization work with SyntenVista (www.dcs.gla.ac.uk/~ela/Synteny) for comparative genomics [2, 1] requires access to large amounts of data, some of which can be acquired online, from Ensembl (www.ensembl.org), and some which require preprocessing and large joins between several external data sources. We currently combine Ensembl data with data from a local mirror of pre-processed information [3]. We find our approach to be deficient in two ways. We rely on servers which may become unavailable, or offer limited bandwidth, and we offer partly stale data, as the complex processing required in data integration is not carried out frequently enough. We believe that an alternative solution needs to be developed, and propose a new approach, e-Aglets (pronounced eaglets), combining agents and web services, in order to provide a reliable service offering up-to-date information for a large number of data-hungry client visualizations.

SyntenVista shows comparisons of chromosomes, genes and gene groups in various species, for instance the rat, the mouse and the human. It integrates data from a small number of sources, but we are planning to extend it as a front-end to a large number of integrated data sets. The bulk of the data comes from Ensembl and is acquired as requested by the user who selects data for viewing. Additional data sets, called QTLs (quantitative trait loci), are also shown. They have been gathered and integrated in a local database, and are being served from that source. Obvious problems relate to the fact that data becomes stale, and our server also suffers from occasional time-outs. We have often experienced that our application ceased to work, or data loading took a long time, due to problems with Ensembl. Trying to design a robust solution to the data needs of SyntenVista is pushing our thinking towards a new solution for data provision. Before we move towards our proposal we describe our data integration solution.

3 Current architecture

Figure 1 shows the application architecture. SyntenVista reacts to user requests for data. Requests are forwarded via jdbc to Ensembl and to a local database of integrated information from MGI (Mouse Database, www.informatics.jax.org), Ensembl and OMIM (www.ncbi.nlm.nih.gov/entrez/query.fcgi?db=OMIM). Our application sends requests for gene and QTL data. Table 1 summarises

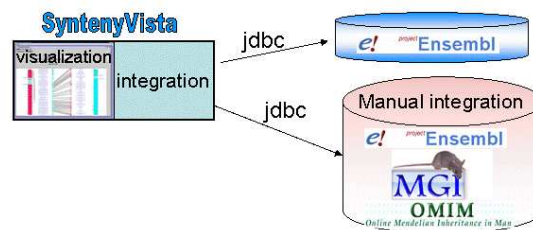


Figure 1: Current data integration architecture: combining on-demand access to Ensembl and to a local mirror of integrated data from Ensembl, MGI and OMIM

typical data sizes for gene and QTL requests. Retrieval of all genes on a chromosome takes between 0.5 and 4 s, and 0.5 s for QTLs, and at least two chromosomes are shown at any time. QTL data appear to be less abundant, but recent data dumps in MGI show a total of 2400 QTLs for the mouse, i.e. 100 on average per chromosome. We currently fetch 11 columns per data row in Ensembl. Additionally, we now have a requirement to show micro array probe mappings and results for all the chromosomes in three species. This will increase the data sizes by at least 50-100% as each gene may have a matching micro array probe, and this probe may have produced positive or negative results in a number of different experiments. There will also be new proteomics data for each gene, and additional information about the genes which have a protein with a known function or an identified 3-D structure. As the amount of data to be shown is significant, and may be user-specific, as it may include arbitrary data sources to be added on the fly, we believe that a more robust solution would improve data availability for visualization.

4 e-Aglets architecture

We maintain that a significant improvement to service reliability and quality could be achieved by a new architecture which combines features of web services and agents and satisfies the high

| Species | Chromosome | gene rows | size B | QTL rows | size B |
|---------|------------|-----------|-----------|----------|--------|
| rat | 10 | 1603 | 765,722 | 49 | 3,538 |
| rat | 20 | 592 | 283,472 | 5 | 465 |
| rat | 7 | 1349 | 644,563 | 25 | 1,911 |
| mouse | 1 | 1365 | 779,325 | 69 | 5,445 |
| mouse | 17 | 1072 | 638,091 | 16 | 1,347 |
| mouse | 18 | 580 | 333,620 | 14 | 1,176 |
| human | 1 | 2202 | 1,051,444 | 42 | 3,313 |
| human | 10 | 856 | 409,400 | 24 | 1,966 |
| human | 5 | 953 | 455,669 | 23 | 1,814 |

Table 1: Sample data sizes for mammalian chromosomes

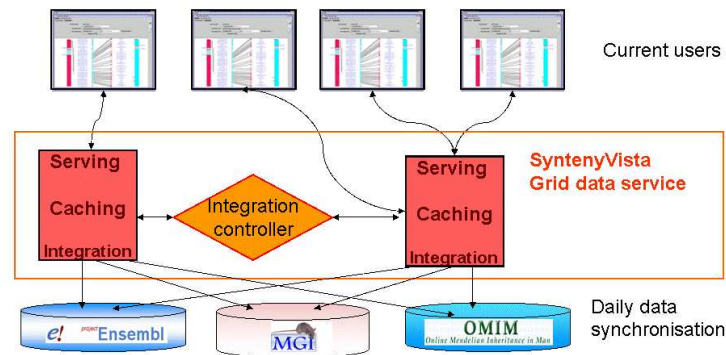


Figure 2: e-Aglets: a new architecture for data integration, consisting of migrating agents which perform data integration and serve it to client visualizations

bandwidth needs of visualization applications. Figure 2 shows a distributed service in operation. In this architecture we separate the visualization component from the task of data integration which is understood to be a migrating and replicating service. We assume that the semi-manual data integration logic has been captured algorithmically, and can be performed automatically on demand, for instance daily. This data can be cached and reused as long as it is fresh.

Service in operation consists of several clients viewing various data sets, a number of agents positioned anywhere within a grid, possibly in the same grid that the clients are operating, and a few controller instances. The architecture ensures that all clients have access to fresh data which has been acquired overnight and which is cached in the agents. As long as the system is stable (client software does not inform their controller that the service is slow, or the integrator is still robust, as far as the controller can tell) there are no changes to the architecture. The role of the controller, of which there have to be one or more instances, is to keep information about active e-Aglets, monitor their performance and collaborate with clients in e-Aglet discovery and spawning. It is also feasible that the controller keeps information about the location of full data copies and helps newly spawned e-Aglets in acquiring an up-to-date set of data.

5 e-Aglets adaptability

We assume that visualization software is supported by a technology similar to Web Start¹ which ensures that the most up-to-date version of the visualization software is used. The Web Start server

¹java.sun.com/products/javawebstart

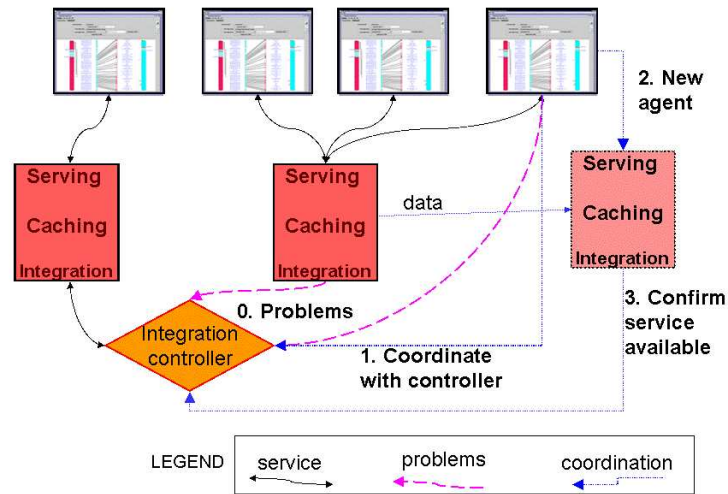


Figure 3: e-Aglets: a new service is spawned when a client notices communication delays, or a server notifies a problem

is normally contacted when the client application starts up, and, if possible, it should tell the client the address of an active controller. If a controller is not available, the client should contact a web registry to see if any copies of the e-Aglet controller for SyntenyVista can be found. There should always be redundant controllers running, to help any new clients to join the system. A controller should know locations of the data. Its task is to collaborate with the clients, control existing e-Aglets, monitor their data content, and help with e-Aglet spawning and migration. Following scenarios need to be handled, and some of the work involved is shown in Figure 3.

- When a client is started up and issues a data request, it registers with the controller via Web Start, and the controller decides if a new service is needed to satisfy the information needs. If such a service is not available it will have to be created. If a service is available and performing well, the client is redirected to use it.
- A client may realise that a service is slow. In that case it should initiate service replication, in collaboration with the controller, or on its own, if no controller can be found. The client may initiate the creation of a controller. A new service with the new controller and integration subsystem should be positioned within a grid that supports the client.
- The controller may find out that a service is blocked or performs poorly. It may then initiate service replication, control its progress, and then redirect some of the clients to a new service.
- A service finds out that it has to shut down. It informs the client and the controller, and a new service is found or a replica created.

The migration/spawning is carried out as follows. First an available server is identified in one of the collaborating grids, and asked if it can host an agent. Data are copied from one of the instances that are running and have the appropriate data, and is immediately made available to the client. Then the agent software is copied and restarted to suit local constraints, and select an appropriate time slot for cache updates. An agent that is not used for some time will notify the controller, and may be allowed to shut down. This will prevent the system from having too many agents, as new instances will only be generated when there is no immediate access to a controller or no available service. Appropriate mechanisms will have to be developed to schedule checks on the servers, and data held at each instance. Algorithms will be needed to decide which servers to shut down, how

best to populate them with data, and how to balance caching against data re-acquisition from the internet sources. Monitoring software will gather information on the relative time spent on integrating, data copying between instances, and service performance, and will be used to tune the solution. Later a self-tuning algorithm will be developed.

6 Discussion and conclusions

We observe that visualization software requires better data provision than is possible with current technologies. Visualizations require bulk access to data, and web servers and services do not provide the correct granularity and speed for bulk requests. Additionally, any timeouts to the services are extremely undesirable and could be prevented by new data delivery architectures which focus on bulk data consumers. Such bulk data provision could also be useful in data mining and integration environments and would make current grid architectures more suited to user-end applications where limited processing happens at the client, but the user requires a highly responsive service. The data integration dimension is considered here as essential, and is also catered for, as it cannot be done on the fly, due to large volumes of data that are being processed, and lack of full query support at data sources.

Typically, visualization solutions consist of two parts: data acquisition part, and data rendering part, with a possible third element of data processing. Data acquisition and integration is distinct from data rendering, and if freshness of data can be guaranteed, it is faster to use cached pre-processed data. In our scenario the proposed visualization will be able to use data from a cache, if such data are present, directly access the data sources for some data that does not require preprocessing, or choose to wait for a process which will establish a local cache of up-to-date information. This will provide resilience and flexibility.

The e-Aglets proposal is a mixture of agent and web-service technologies, and can employ many solutions from the database world. It can use existing cache consistency algorithms, grid security technologies, XML, compression and web services for fast delivery of data, and adapt existing protocols to suit the needs of applications which require large data sets.



Photo Credit: Dave Menke
U.S. Fish and Wildlife Service

Figure 4: e-Aglets

References

- [1] E. Hunt and N. Hanlon. SyntenyVista. In *NordiCHI 04*, pages 455–456. ACM, 2004.
- [2] E. Hunt, N. Hanlon, D. Leader, H. Bryce, and A. F. Dominiczak. The Visual Language of Synteny. *OMICS - A Journal of Integrative Biology*, 8(4):289–305, 2004.
- [3] R. O. Sinnott, M. Atkinson, M. Bayer, D. Berry, A. Dominiczak, M. Ferrier, D. Gilbert, N. Hanlon, D. Houghton, E. Hunt, and D. White. Grid Services Supporting the Usage of Secure Federated, Distributed Biomedical Data. In *Proceedings of UK e-Science All Hands Meeting, September 2004*, pages 135–138, 2004.