Exploring Microbial Genome Sequences to Identify Protein Families on the Grid

Yudong Sun, Anil Wipat, Matthew Pocock, Peter A. Lee, Keith Flanagan, and James T. Worthington

***Abstract*—The analysis of microbial genome sequences can identify protein families that provide potential drug targets for new antibiotics. With the rapid accumulation of newly sequenced genomes, this analysis has become a computationally- and data-intensive problem. This paper describes the development of a web service enabled, component based, architecture to support the large-scale comparative analysis of complete microbial genome sequences, and the subsequent identification of orthologues and protein families (Microbase). The system is coordinated through the use of web service based notifications and integrates distributed computing resources together with genomic databases to realize all-against-all comparisons for a large volume of genome sequences and to present the data in a computationally amenable format through a web service interface. We demonstrate the use of the system in searching for orthologues and candidate protein families which ultimately could lead to the identification of potential therapeutic targets.**

***Index Terms*—Genome analysis, grid, microbial genomes, protein families, web services.**

I. Introduction

DEVELOPMENTS in comparative genomics have been helping to provide novel techniques for therapeutic anti-microbial drug discovery. The comparative analysis of complete microbial genome sequences can identify unique proteins and homologous protein families conserved in and between genomes, which can be screened in the search for new antibiotic targets [1]-[3]. The approach promises to enhance our capability to develop antibiotics to tackle the increasing risks of infectious diseases in humans, that include the emergence of new bacterial pathogens, the spread of epidemic diseases, and the intensified resistance to existing antibiotics [2].

With the rapid increase of completed microbial genome sequences, the comparative analysis of whole microbial genomes has become a computationally- and data-intensive problem. For example, whole sequence alignment and homology search involve enormous computations over a huge volume of genomic datasets. Grid computing can federate distributed resources using open, general-purpose protocols to create a powerful computing system that meets end-user requirements of on-demand access to computing capabilities [4],[5], and promises to provide a solution to the highly increasing computational demand in biology, biomedicine and bioinformatics [6]-[8]. Web services and service-oriented architecture are important principles and technologies in the implementation of the grid and the exposure of such resources as services to end-users [4].

The Microbase project [9] has developed a grid-based system to service the timely dynamic or on-demand comparative analysis of microbial genome sequences in biological and biomedical research. We employ web service based technologies, in particular a web service based notification system, to integrate distributed components and orchestrate their interoperability. Consequently, the system is able to perform large-scale genome comparison and analysis, using a variety of bioinformatics tools, and expose our pre-computed dataset of comparison results to users across the Internet. The pre-computed dataset provides a flexible data repository of genome sequence similarities that will support a number of subsequent analyses, both by a human user and a computational client, such as a workflow. A web service-based client interface has been implemented to facilitate computational access to the data repository. Microbase enables biological and biomedical researchers to carry out customized analyses directly without having to repeat the computationally-intensive genome comparisons. In order to demonstrate the utility of the system, we have used this pre-computed dataset to discover and define protein families. The protein families we identify may aid in the discovery of new therapeutic agents and in the development of new antibiotics by highlighting proteins that are conserved in bacteria and may form suitable targets. A protein family that is conserved amongst a phylogenic group of bacteria can be viewed as a potential target for broad-spectrum antibiotics, whereas a protein unique to a specific pathogenic bacterium can be considered as the target of a narrow-spectrum drug.

Our preliminary version of the Microbase system, termed Microbase*Lite*, has been implemented and integrates computing servers, a database server, and a campus grid. The system has been used to execute all-against-all comparisons,

This work was supported by the UK BBSRC e-Science and Bioinformatics initiative and the DTI under Grant 13/BEP17027.

Y. Sun was with Newcastle University, UK. He is now with Oxford University Computing Laboratory, Oxford OX1 3QD, UK (email: yudong.sun@comlab.ox.ac.uk).

A. Wipat, M. Pocock, P. A. Lee, and K. Flanagan are with School of Computing Science, Newcastle University, Newcastle NE1 7RU, UK (e-mail: anil.wipat@ncl.ac.uk; matthew.pocock@ncl.ac.uk; p.a.lee@ncl.ac.uk; keith.flanagan@ncl.ac.uk).

J. T. Worthington was with Newcastle University, UK. He is now with Convergys Corporation (email: j.t.worthington@blueyonder.co.uk).

using different tools, for 250 microbial genomes, mainly bacteria and archaea. Genomic comparisons are automatically carried out in response to web service-based notification messages triggered as new genome sequences are deposited in remote public genome databases. Two algorithms have been implemented to search the 250 genomes for putative orthologues and COGs (Clusters of Orthologous Groups). The system has been developed with web service-based user accessibility as a prime concern. Web service-based interfaces including an application programming interface (API) and a graphical viewer have been developed to allow end users to retrieve genome sequences, pre-computed comparison results and the protein families via the web.

A full overview of the project will be presented in the future. In Section II we discuss previous work related to the project and the novelty of our approach. The Microbase*Lite* system is outlined in Section III. Section IV discusses the identification of protein families based on the system and finally our ideas for future work are covered in Section V.

II. Related work

Grid computing is increasingly being employed in biological and biomedical research and in particular to support genome comparison and analysis. For example, GNARE (Genome Analysis Research Environment) [8] is a scalable grid-based system using Globus, Condor, and GriPhyN virtual data system and running on the grid systems as GRID2003, TeraGrid, and the DOE science Grid to automate genome analysis (including data acquisition from genome databases), mainly using BLAST. TIGR's DCE (Distributed Computing Environment) [10] is an institutional grid system that connects the on-campus computers and database servers with Sun Grid Engine. Genome analysis is implemented on demand using BLAST, MUMmer, and HMMsearch, and a repository of protein and nucleotide sequence data and a protein database of all-vs.-all searches to identify protein similarity is maintained. The GPSA (Grid Protein Sequence Analysis) [11] web portal provides a user interface to run protein sequence analyses, including BLAST, FASTA, SSEARCH, and ClusterW, on the European EGEE Grid [12]. The PUMA2 system also provides a flexible system for grid based, high-throughput analysis of genome sequences, based on the use of the GADU system, leveraging experience gained from the GriPhyN physics project [13].

Whilst the projects mentioned above concentrate on grid enabled gene and protein sequence comparison, there are far fewer projects providing data from analyses that employ sequence comparison data. One such application area is the identification of orthologous genes and their grouping into protein families. The identification of orthologues is an important application of comparative genomics that seeks to establish relationships between similar proteins and genes from different genomes for subsequent evolutionary and functional studies. The COG database [14],[15] contains the clusters of orthologous proteins identified from different phylogenetic lineages and has become widely accepted for the annotation of proteins. *coli*BASE [16],[17] is a database of *Escherichia coli, Shigella,* and *Salmonella*, reflecting the full diversity of *E. coli* and its relatives, which includes the putative orthologues found in these genomes. The e-Fungi project [7] has performed homologue analysis for fungal genomes using BLASTP and has employed a Markov Chain Clustering (MCL) method to cluster protein families for phylogenetic and pathogenic analysis.

Drug discovery is also an emerging application area of grid computing. For example, ^{my}Grid [18] is a service-based grid middleware framework to manage the complex process of life science research. ^{my}Grid supports data management, new discovery notification, and provenance management in the drug discovery process, in particular through the use of e-Science workflows [19]. In addition, the EGEE Grid has a project for the virtual screening of a large amount of data to find potential drugs for infectious diseases such as malaria [20].

In comparison, the Microbase project supports the computational analysis of microbial genomes, particularly bacteria. The project provides a grid-based system that possesses a distributed component infrastructure to integrate grid computing, remote public genome databases, with a pre-computed, dynamically updated genome similarity dataset, useful to biological and biomedical researches. We build on the approaches introduced by similar projects, extending them with a focus on ease of computational access to the datasets and by introducing flexibility in the analyses available for these data. Unlike many existing systems, the individual components of the system are amenable to deployment in a distributed computing environment. Inter-component communication and interoperability is facilitated via web services, so in theory individual components may be located disparately, and in multiple instances, as long as network communication is maintained. The mechanism behind inter-component communication is based on web service-based notifications, which are used to orchestrate the behavior of the system. In particular, notification facilitates the auto-update of the pre-computed dataset to import and process new genomes from public genome databases as they are released. The system also provides a greater range of all-against-all genome comparison results at both nucleotide and protein levels, than many existing systems. We provide tools including BLASTP, BLASTN, and the suffix-tree based algorithms MUMmer and PROmer, but additional tools can easily be incorporated as the need arises.

The results generated provide a flexible dataset to support different, custom genomic analyses, in particular those that underpin the successful identification of therapeutic targets such as protein family identification. Web service-based client interfaces including a documented service API and a graphical genome viewer maximize the ease of access for end users to the pre-computed dataset. Users can directly undertake user-specified analyses without having to redo the time-consuming genome comparisons that usually exceed the computing

capability of a single institute. A web service-based infrastructure also facilitates the enactment of workflows to manage subsequent sophisticated genome analysis processes. In this work, we have used our pre-computed genome comparison dataset, to establish orthologues and protein families from 250 proteomes, (to our knowledge, a greater range than the current versions of the related tools discussed above) in order to identify potential therapeutic protein targets.

III. MicrobaseLite

Microbase*Lite* is the initial system implementation of the project. As detailed in Fig. 1, Microbase*Lite* consists of distinct components for computation, data acquisition and database management, user access, and component orchestration. The components can be deployed on distributed servers and orchestrated via web service-based notification message passing mechanisms. The main components include the microbial genome pool, the genome comparison pool, the notification service, and the client interface.

A. Microbial Genome Pool

The microbial genome pool provides an up-to-date database of complete microbial genome sequences. Genomes published in the public EMBL nucleotide database [21] are imported into a local microbial genome database and subsequently compared to all other genomes in the repository. Automatic updates of the database are triggered by the web service-based *notification service* where the collector component is deployed to regularly check for new microbial genomes in the EMBL database. When a new genome is available, the collector notifies the genome loader in this pool to download the new genome into the local database. More details of the notification service are presented in subsection *C*.

To facilitate end-user access, the microbial genome pool uses BioJava [22] to parse the plain-text genome sequence records obtained from the EMBL database and enters the sequences and their annotations into the microbial genome MySQL database with a BioSQL schema [23]. At the time of writing, the microbial genome pool holds 250 microbial genome sequences.

Web service-based client interfaces have been developed to allow end users to flexibly access the genome sequences in the microbial genome database. A Java API to the database has been exposed as web services and a graphical genome viewer application developed to visualize the data. The API has been implemented using Apache Tomcat and Codehaus XFire, a service-oriented SOAP (Simple Object Access Protocol) implementation [24]. Users can retrieve DNA and protein sequences, gene features (e.g. CDS, tRNA, mRNA), and annotations (e.g. a sequence's ID, organism species, and references) using this interface. The genome viewer enables end users to browse the genome sequences in a graphical format using a web browser, and has been developed using JSP (JavaServer Pages) and Java Servlet and deployed under Tomcat. The API and genome viewer are connected to the

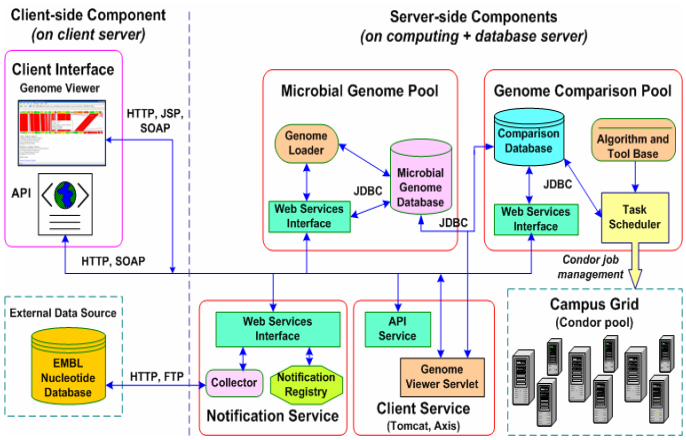

Fig. 1. Microbase*Lite* architecture

microbial genome database via JDBC (Java Database Connectivity). Details of the client graphical user interface are presented in subsection *D*.

B. Genome Comparison Pool

The genome comparison pool is a central component responsible for conducting genome comparison and analysis within the system, and for maintaining the pre-computed comparison dataset.

The genome comparison pool performs pairwise sequence comparisons using existing tools to establish sequence similarities at nucleotide and protein sequence levels, both for whole genomes and individual genes. Currently, four sequence comparison tools are applied: BLASTP, BLASTN, MUMmer, and PROmer. BLASTP [25] is a protein-protein comparison tool that searches similar proteins. BLASTN [25] is a pairwise nucleotide alignment tool to find similar nucleotide fragments. MUMmer [26] is a suffix-tree based fast tool for nucleotide alignment that gives a concise report on similar nucleotide sequence fragments. PROmer [26] is a variant of MUMmer that translates two nucleotide sequences into amino acids in all six frames, finds all matches in the amino acid sequences, and then maps the matches back to the positions in the nucleotide sequences.

Since our alignments are non-reciprocal statistical computations, we compare each member of a pair of genomes against each other (i.e. genome A is compared against B and genome B against A). In total, the all-against-all comparison of 250 microbial genomes requires 62,500 pairwise comparisons of genome sequences. The comparison of complete genome sequences and their encoded proteins is usually a computationally intensive task. For example, the BLASTP comparison of two species of *Bacillus*: *Bacillus anthracis* and *Bacillus cereus* (approximately 5500 proteins each) takes 12 minutes on a 2.8GHz CPU and produces 95MB of output data. The BLASTN comparison of two *Leptospira interrogans* genomes (approximately 4.3M base pairs each) takes over 8 hours and produces 193MB data. Using the four comparison tools, the all-against-all comparison is an overloaded task that exceeds the capability of common computing systems.

To cope with this burden, the genome comparison pool utilizes a grid-based system to support all-against-all genome comparison. The system is based on a campus grid consisting of computing clusters distributed within different labs of our university. The clusters are federated into a powerful computing environment with Condor to handle large applications that overburden any single cluster. The access to the campus grid is authenticated by the user accounts in the Condor system. To manage the execution of the large number of pairwise comparisons, the genome comparison pool provides a *task scheduler* that calls the Condor job management mechanism to realize parallel execution of the comparison jobs on the grid. Running on a computing server, the task scheduler creates an individual job for each comparison and submits the job to the Condor job queue. Condor, in turn, allocates the job to execute on an idle node. The task scheduler can consecutively submit a large number of jobs to run in parallel depending on the available computer nodes in the grid. Meanwhile the task scheduler is also responsible for managing the pace of job submission to keep the overall workload at a reasonable level. The task scheduler uses a Condor command to check the status of each job. Once a running job has finished, a new job will be submitted to run. Therefore, the task scheduler along with the underlying Condor can progressively dispatch the large number of jobs for execution yet maintain load balancing to prevent the system from being overloaded by excessive queuing jobs. The task scheduler is implemented in Java and can use different grid middleware. For example, the task scheduler can also be executed on Sun Grid Engine (SGE) by calling the job submission and status-checking commands of SGE.

A comparison job also parses and loads its results into the comparison database when a comparison has finished. The comparison database is a MySQL database which stores all pairwise comparison results. The web service-based API and genome viewer (discussed earlier) also support access to the comparison database. Since many techniques in genome analyses are based on sequence similarities, the comparison database provides an instantly accessible, base-level pre-computed dataset that enables biological and bioinformatics researchers to directly implement in-depth genome analyses without having to redo time-consuming genome comparisons. In Section IV we illustrate this concept by using these results to define protein families for all genomes within the database.

At present, 250-against-250 genome comparisons have been completed on the grid system and the comparison database has reached 28GB. Fig. 2 shows the execution time of all-against-all comparisons on a selected number of microbial genomes. The execution time includes the parallel execution time of all pairwise comparisons using the four tools, and the time for parsing and loading the results into the comparison database. The execution time is decreased by exploiting the CPUs available on the campus grid. However, the centralized database ultimately becomes a bottleneck to the overall performance when additional CPUs are used. We intend to address this limitation in future versions by deploying a decentralized database on multiple servers.

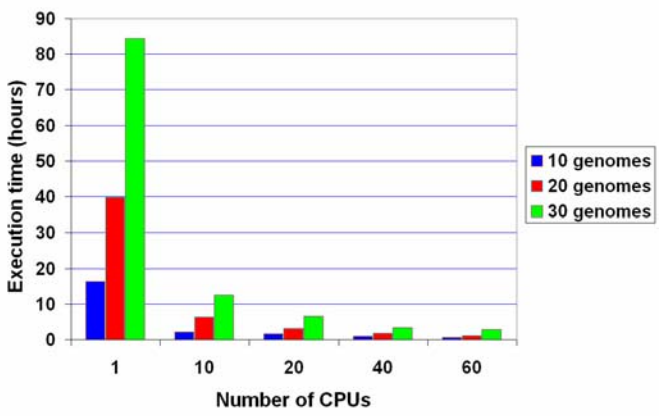

Fig. 2. The execution time of all-against-all genome comparison on the grid

Once a pairwise comparison result has been generated it is reusable in further analyses. 68 hours were required to run the comparisons of 165 genomes on 40 CPUs and to populate the results into the comparison database. Subsequently, the database was regularly updated to import the data from new genomes. When a new genome is imported, it is compared with each of the existing genomes in our database using the four tools and the database is updated with the new comparison results. The update process is automatically invoked by the notification service and the comparisons are also executed on the grid system with the support of the task scheduler.

C. *Notification Service*

The notification service is based on the ^{my}Grid notification system [27],[28]. The ^{my}Grid notification system is a web service-based event notification that supports topic-based message publish and subscribe. Subscribers can receive notification messages on a registered topic in push and pull models. A subscriber can be a real user or a software component. The push model delivers a notification by calling back to the client code deployed at a web service endpoint. Microbase*Lite*'s notification service utilizes the push model to notify subscribed components about the arrival of new genomes. The microbial genome pool relies on this notification service to trigger the updates to the local microbial genome database. A collector is deployed in the notification service to monitor a remote genome repository (the EMBL database). When a new microbial genome is published, the collector will push a specific notification to the microbial genome pool, which in turn requests the new genome sequence from the collector. The collector then downloads the genome file from the remote repository via FTP, forwards it to the microbial genome pool, which parses the genome file, and then loads the sequence into the microbial genome database. Subsequently, the microbial genome pool sends a notification to the genome comparison pool that triggers the task scheduler to start the comparison of the new genome against all existing genomes to update the pre-computed dataset. The notification service also allows human users to receive notification of new genomes. A

notification message will be sent to the subscribers when a new genome has been added to the database. The notification service is implemented using Apache Tomcat and Codehaus XFire and uses a MySQL database to store the registered publishers, subscribers, and notification messages.

D. *Client Interface*

The client interface exposes the microbial genome database, the pre-computed comparison database, and the protein families for external users to access over the Internet. The client interface includes a Java API and a genome viewer. The API provides various methods that can be called in user programs to retrieve the genome sequences and the comparison results, by connecting to the web services deployed on the server side under Apache Tomcat and Codehaus XFire. The genome viewer uses JSP and Servlet and is also deployed under Apache Tomcat. The genome viewer allows web users to browse the genome sequences and comparison results, and to search the protein families via a web browser such as Internet Explorer and Mozilla Firefox. In Fig. 3, the genome viewer shows the BLASTP alignment between *Escherichia coli* CFT073 (up) and *Escherichia coli* O157:H7 EDL933 (down). The strips in between highlight the hits of similar fragments between the two sequences. Hovering over a region using the mouse will pop up a window showing the detail of a hit or a gene.

IV. Protein Family Analysis

A protein family is a group of similar proteins that are related through evolution. Proteins directly related to each other through evolutionary processes are called homologues, and can be further classified as orthologues and paralogues. Paralogues are homologous proteins in the same genome. Orthologues are homologous proteins in different genomes that evolved from a common ancestral gene. Orthologues often retain the same function in the process of evolution. Thus, orthologue search is an effective method to predict the evolutionary relationships and infer the functions of a group of genes or proteins [15],[29],[30]. Identifying orthologues can also aid in the drug discovery process. For example, when seeking to develop a new broad spectrum antibiotic it is useful to identify potential protein drug targets that are conserved in the target species, but not present in humans or higher eukaryotes. In this respect, groups of orthologues that are unique to bacteria can be considered as potential targets for new broad-spectrum antibiotics.

As proof of principle for the Microbase system we have used the pre-computed BLASTP results from Microbase*Lite* to carry out orthologue searches providing a starting point for users wishing to identify conserved proteins as drug targets. We also extend this analysis to group orthologous genes into families, which may also provide useful information in this respect. In order to accelerate the identification of orthologues and protein families in such a large dataset, we parallelized the algorithms as described below.

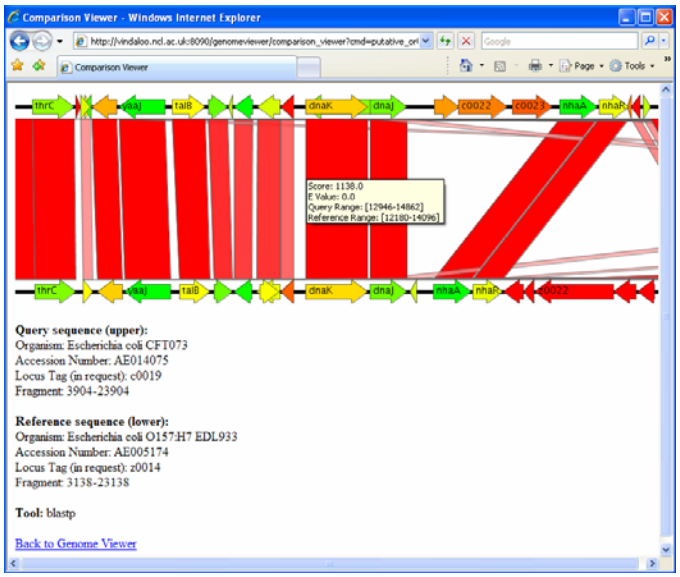

Fig. 3. The genome viewer showing BLASTP alignment between *Escherichia coli* CFT073 (upper) and *Escherichia coli* O157:H7 EDL933 (lower) fragments. The popup window shows the detail of the hit between the dnaK genes of two sequences.

A. *Putative Orthologues*

Putative orthologues are defined as the proteins that have mutual best hits in the BLASTP comparison and satisfy specific requirements on the aligned portions. We use the same criteria of putative orthologues specified by *coli*BASE [16],[17]. Since orthologues reflect the evolutionary relationships of the genes that encode those proteins, for convenience we use the terms “protein” and “gene” interchangeably when referring to orthologues in the following discussion. The search for putative orthologues begins by selecting the mutual best hits from the BLASTP results.

Definition 1: Given protein α from genome A and protein β from genome B (A and B are different genomes), α is a *best hit* to β if the hit has the highest bit score and the lowest E-value in all BLASTP hits between α and any proteins of genome B. The hit between α and β is a *mutual best hit* if α is a best hit to β and β is also a best hit to α.

The mutual best hit means that α and β are the most similar proteins in all proteins between genome A and B, as defined by BLASTP. The evolutionary and functional relationships between the similar proteins, and therefore the genes that encode the proteins, can be inferred based on the mutual best hits that are also putative orthologues as defined below.

Definition 2: If the mutual best hit between protein α and β satisfies two conditions on the aligned portion as following, α and β are *putative orthologues*:

1. α and β have at least 80% amino acid identity.
2. The aligned portion covers at least 90% of the shorter sequence.

With the definitions above, the search for putative orthologues can be achieved in three steps:

1. Select the best hits in all BLASTP hits of each protein of a genome against the proteins from each of other genomes;
2. Identify mutual best hits among the best hits;
3. Check the amino acid identity and alignment coverage of the mutual best hits to determine the putative orthologues that satisfy the two conditions as in Definition 2.

Microbase*Lite* has a dataset of 646,954 proteins from 250 genomes. Pairwise BLASTP comparisons have reported more than 400 million hits with a total size of 22GB. A parallelized search was implemented to identify the putative orthologues among this huge number of hits. Running on eight 2.8GHz CPUs, the search was completed in ten days (compared to in excess of two months if run on a single CPU). Additional CPUs have not been used because the search is data-intensive and restricted by the speed of the database server and therefore using more CPUs does not improve the performance. (This problem will be solved by deploying a decentralized database on distributed servers that can improve the parallel search).

The number of putative orthologues found by the search depends on the specified values of cutoff conditions. Using the conditions in Definition 2, the search found putative orthologues for 287,490 proteins. This represents 44.4% of the total proteins in our database. Among these proteins, most of them have more than one putative orthologues each. However, some genes are conserved in very limited numbers of organisms. There are 98,206 proteins that have only one putative orthologue. For example, the gene BH14430 (locus tag) of *Bartonella henselae* str. Houston-1 (an agent of cat scratch fever and bacillary angiomatosis) has only one putative orthologue, the gene BQ11380 (locus tag) of *Bartonella quintana* str. Toulouse (an agent of trench fever, bacillary angiomatosis and bacteremia).

The comparison database in the genome comparison pool has also been populated with the putative orthologues defined using this approach. Users can search for orthologues of a given gene via the genome viewer.

B. COGs

COGs (Clusters of Orthologous Groups) are a classification of homologous protein families [15],[29]. A COG is composed of orthologous proteins or orthologous groups of paralogous proteins from three or more genomes. A COGs search identifies both orthologous proteins from different genomes and paralogous proteins from the same genomes. The paralogues from a genome are collected into a group that is treated as a single candidate orthologue in the search for COGs. Putative orthologues only reflect one-to-many relationships of the proteins; nevertheless, COGs can reveal more comprehensive, many-to-many relationships amongst the proteins from the same and different genomes.

Our search for COGs is based on the same set of mutual best hits obtained in the putative orthologues search.

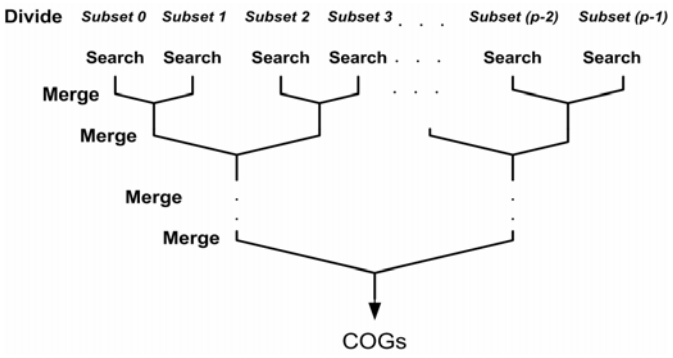

Fig. 4. Divide and conquer method for parallel COGs search. The set of proteins is split into p subsets. The search of three-orthologue groups runs on each subset per CPU, followed by $\log p$ rounds of merge.

However, the COGs search does not set any cutoff requirement on aligned portions. In addition, the COGs search needs to identify all paralogues that are the mutual best hits from the same genomes. Our COGs search includes the following steps based on the COG construction procedure from the COG database project [14],[15],[29]:

1. Identify best hits and mutual best hits from all BLASTP hits (available from the putative orthologues search).
2. Find paralogues in each genome and group them.
3. Search the groups of three orthologues in the mutual best hits. Given three proteins α, β, and γ, the proteins form a group of three orthologues if (α, β), (β, γ) and (α, γ) are mutual best hits. A group of paralogues is regarded as a single orthologue in the formation of the groups.
4. Merge the groups that have at least a common mutual best hit, if the merge will not put the proteins from the same genome (except those are paralogues) into a group.
5. COGs have finally been formed if the groups cannot be further merged.

The process involves an exhaustive search of the groups containing three orthologues (triples) and then an iterative merge of the groups that have common mutual best hits. This is an extremely compute- and data-intensive process. In order to establish a fast and parallel implementation of the COGs search, a *divide and conquer* method is used. As Fig. 4 shows, the divide and conquer method consists of three phases:

1. *Divide*: divide the whole protein set into p subsets.
2. *Search*: search the groups of three orthologues for the proteins in each subset based on the BLASTP hits, and perform an initial merge of the groups. This phase can be run in parallel on p CPUs.
3. *Merge*: merge the groups of orthologues from different subsets in $\log p$ rounds. Round i runs on $p/2^i$ CPUs ($i=1, 2, \ldots, \log p$) to merge the groups of orthologues between adjacent CPUs. The complete COGs are formed in the final round which is run on one CPU.

In the search of three-orthologue groups (α, β, γ), only the starting point α is selected from the corresponding subset. Its orthologues β and γ can come from other protein subsets.

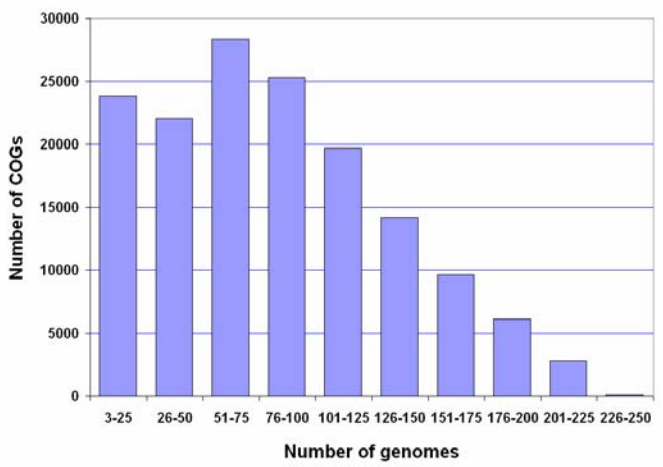

Fig. 5. The composition of the COGs in terms of the number of distinct genomes in each COG. For example, there are around 24,000 COGs each containing the orthologues contributed from 3 to 25 distinct genomes.

Therefore, the parallel search can find the same groups of orthologues as a sequential search.

The COGs search over all proteins of the 250 microbial genomes took 30 days on eight 2.8GHz CPUs excluding the time for filtering the mutual best hits which are already available. We estimate that the sequential search would take in excess of 200 days if running on a single CPU. Due to the more intensive search on the table of mutual best hits and the table of intermediate orthologous groups, the centralized database restricts the performance improvement of COGs search when using more than 8 CPUs. This problem will also be solved when a decentralized database is deployed in the future.

Our COGs search identified 152,011 clusters of orthologous groups containing 546,699 orthologues, of which 531,441 are single proteins and 15,258 are groups of paralogues. In total, 571,701 proteins were assigned to one or more COGs, representing 88.37% of all proteins from the 250 genomes. Also, 18,455 groups of paralogues have been found which consist of 47,608 proteins. Fig. 5 shows the composition of the COGs in terms of the number of distinct genomes contributing to each COG, giving an indication of the degree of conservation of COGs across a range of genomes. The results demonstrate that a large number of COGs span between 50 and 75 genomes and hence appear to be well conserved. Around 3000 COGs contain members from 200 genomes, but far fewer span greater than 225 genomes.

As a COG is formed by merging the orthologous groups, the COGs search collects more orthologues together reflecting the many-to-many relationships between proteins and between the genes that encode them. The results of our COGs identification are also incorporated into the comparison database within Microbase*Lite* and users can search the COGs for a given gene via the genome viewer or access the results via the web service exposed API. When a new genome is imported, it will be compared with existing genomes to find the mutual best hits between them. If a protein from the new genome has found two mutual best hits in a COG, it can be assigned to that COG.

V. Conclusions

Grid technologies enable a more rapid analysis of genome sequences facilitating a more intensive exploration of genomic data than can be achieved with traditional technology. In turn, this allows knowledge to be more quickly derived from our investment in sequencing programs and helps to address the problem of the analysis of rapidly accumulating genomic data. The Microbase project is developing a grid-based environment to support computationally- and data-intensive genome comparison and analysis, particularly for the analysis of microbial genomes. Microbase*Lite* is a system implementation that integrates distributed computing and data resources to perform the comparison and analysis of genome sequences. The system features the extensive use of web service technologies for component orchestration, notification, database update, and user access. A large volume of pre-computed comparison dataset has been generated on the system and exposed as a base-level database to end users for in-depth biological and biomedical research. We expose the results in both a computational amenable and user-friendly form through the use of web services and graphical user interfaces.

One of the important applications implemented within Microbase*Lite* is the identification of protein families in a large number of proteomes. Such searches may aid the identification of potential targets for drug discovery and increase our understanding of protein evolution, and we hope the system will prove useful in this respect.

In the future, we aim to develop a workflow framework to support the definition and enactment of custom applications based on which the system will be able to service user-defined, remotely conceived genome analyses. The system will provide the services to support user application submission and execution on the grid system. A decentralized database will be deployed. We will extend the protein family analysis to include the TribesMCL algorithm and implement other applications such as metabolic reconstruction and promoter searching. Finally, the system is not limited to the analysis of microbial genomes, and we intend to extend our approach to the analysis of eukaryotic genomes.

References

[1] M. T. Black and J. Hodgson, "Novel target sites in bacteria for overcoming antibiotic resistance," *Advanced Drug Delivery Reviews,* vol. 57, no. 10, 2005, pp. 1528-1538.

[2] H. Loferer, "Mining bacterial genomes for antimicrobial targets," *Molecular Medicine Today,* vol. 6, 2000,

[3] J. Rosamond and A. Allsop, "Harnessing the power of the genome in the search for new antibiotics," *Science,* vol. 287, no. 5460, 2000, pp. 1973-1976.

[4] I. Foster, "Describing the elephant: the different faces of IT as service," *ACM Queue* vol. 3, no. 6, 2005,

[5] I. Foster, C. Kesselman, J. M. Nick, and S. Tuecke, "Grid services for distributed system integration," *Computer,* vol. 35, no. 6, 2002, pp. 37-46.

[6] N. Jacq, C. Blanchet, C. Combet, E. Cornillot, L. Duret, K. Kurata, et al., "Grid as a bioinformatic tool," *Parallel Computing,* vol. 30, no. 9-10, 2004, pp. 999-1167.

[7] M. Cornell, I. Alam, D. Soanes, H. Wong, M. Rattray, S. Hubbard, et al., "e-Fungi: an e-science infrastructure for comparative functional genomics in fungal species," in Proc. *4th UK e-Science All Hands Meeting (AHM 2005)*, Nottingham, UK, 2005.

[8] D. Sulakhe, A. Rodriguez, M. D'Souza, M. Wilde, V. Nefedova, I. Foster, et al., "GNARE: an environment for grid-based high-throughput genome analysis," in Proc. *5th IEEE Int. Symp. Cluster Computing and Grid (CCGrid 05)*, Cardiff, UK, 2005, pp. 455-462.

[9] Microbase project, http://www.microbase.org.uk

[10] TIGR Grid Computing, http://www.tigr.org/grid/

[11] GPSA: Grid protein sequence analysis, http://gpsa.ibcp.fr/

[12] EGEE, http://public.eu-egee.org/

[13] N. Maltsev, E. Glass, D. Sulakhe, A. Rodriguez, M. H. Syed, T. Bompada, et al., "PUMA2-grid-based high-throughput analysis of genomes and metabolic pathways," *Nucleic Acids Research,* vol. 34, no. Supplement 1, 2006, pp. D369-D372.

[14] COGs, http://www.ncbi.nlm.nih.gov/COG/

[15] R. L. Tatusov, M. Y. Galperin, D. A. Natale, and E. V. Koonin, "The COG database: a tool for genome-scale analysis of protein functions and evolution," *Nucleic Acids Research,* vol. 28, no. 1, 2000, pp. 33-36.

[16] coliBASE, http://colibase.bham.ac.uk/

[17] R. R. Chaudhuri, A. M. Khan, and M. J. Pallen, "*coli*BASE: an online database for Escherichia coli, Shigella and Salmonella comparative genomics," *Nucleic Acids Research,* vol. 32, no. Database, 2004, pp. D296-D299.

[18] R. Stevens, A. Robinson, and C. Goble, "myGrid: personalised bioinformatics on the information grid," *Bioinformatics,* vol. 19 Suppl 1, 2003, pp. i302-i304.

[19] R. Stevens, R. McEntire, C. A. Goble, M. Greenwood, J. Zhao, A. Wipat, et al., "myGrid and the drug discovery process," *Drug Discovery Today: BIOSILICO,* vol. 2, no. 4, 2004, pp. 140-148.

[20] EGEE battles malaria with grid wisdom, http://public.eu-egee.org/news/fullstory.php?news_id=53

[21] EMBL, http://www.ebi.ac.uk/embl/index.html

[22] BioJava, http://www.biojava.org/

[23] BioSQL, http://www.biosql.org/

[24] Codehaus XFire, http://xfire.codehaus.org/

[25] NCBI BLAST, http://www.ncbi.nlm.nih.gov/BLAST/

[26] MUMmer 3, http://mummer.sourceforge.net/

[27] A. Krishna, V. Tan, R. Lawley, S. Miles, and L. Moreau, "myGrid notification service," in Proc. *UK e-Science All Hands Meeting*, Nottingham, 2003, pp. 475-482.

[28] myGrid project, http://www.mygrid.org.uk/

[29] R. L. Tatusov, E. V. Koonin, and D. J. Lipman, "A genomic perspective on protein families," *Science,* vol. 278, 1997, pp. 631-637.

[30] A. K. Bansal and T. E. Meyer, "Evolutionary analysis by whole-genome comparisons," *Journal of Bacteriology,* vol. 184, no. 8, 2002, pp. 2260-2272.