Comparative genomic workflow

The MIT Faculty has made this article openly available. Please share how this access benefits you. Your story matters.
Comparative Genomic Workflow

We present a comparative genomic workflow to find conserved regions in noncoding parts of genomic sequences. The workflow is designed for cross-species comparisons of all available vertebrate genome sequences, which involves high-throughput sequence alignment and high-performance computational analysis. Therefore, an implementation on a computational grid infrastructure for efficient mapping of comparison tasks is proposed. Results of comparison of mouse and human genomes are illustrated. A relational database will be created to store conserved noncoding regions among all sequenced vertebrate genomes, which enables scientists to determine the functional significances of conserved segments.

The premise of comparative genomics is that a DNA segment conserved across species over the evolution is likely to have some important biological function [1]. Coding DNA sequences conserved from the last common ancestor should be preserved in contemporary genome sequences [2]. Comparative genomics involves comparison of whole genomes across species to identify common segments of DNA or protein sequences and building evolutionary profiles. Comparative genomics applications of coding sequences have been able to detect most known genes [2], [3]. Our short-term goal in this project is to detect regions of noncoding DNA conserved across vertebrate genomes and the long-term aim is to understand potential regulatory functions of the conserved segments in noncoding DNA. This is an essential but challenging problem in future bioinformatics [4], because noncoding regions, which account for about 95% of the genomes, are widely believed not to be junk DNA as perceived before but rather to be important regions of genomes remaining to be investigated for their function in the postgenomic area [5].

We have taken a comparative genomic approach, as the comparison of genomes of interspecies has facilitated the prediction and verification of noncoding RNA genes [6], [7]. For instance, comparative genomics of the human 5q31 region and murine chromosome 11 has identified a coordinate regulator of interleukins IL-4, IL-13, and IL-5 [8]. The techniques for the discovery and characterization of conserved regions in noncoding genomic sequences could have numerous applications in future as more and more genomic sequences become available. In this project, vertebrate sequences were chosen because such conserved segments are likely to have regulatory functions of biological processes of human.

This article describes a workflow for identifying conserved patterns in noncoding regions of vertebrate genomes, with an intention of investigating possible functions of the conserved regions. The annotations of genomes are collected from the Ensembl database [9]. The sequences are then arranged to use for sequence alignment with basic local alignment search tool Z (BLASTZ) [10], which finds gap-free alignments of at least q% identity and l b/s in length. All the conserved noncoding regions identified are stored in a relational database. An user-friendly Web interface provides easy access to conserved regions and related information and visualization capabilities.

To account for the compute- and data-intensive nature of the processes [11], the workflow was implemented on a computing grid architecture that offers higher computing power at a reasonable cost from already-available resources. The implementation has to consider the heterogeneous nature of computing and data resources on the grid and the effective ways of running applications in parallel fashion [12]. We propose to use the master-slave paradigm for communication and dynamic load balancing for scheduling parallel tasks.

Besides commonalities with available noncoding databases [13], the present database of conserved noncoding regions provides 1) conserved noncoding regions in untranslated regions (UTRs: those regions which are transcribed to RNA but not translated), 2) additional information such as the genomes and locations of the conserved noncoding regions, and 3) cross-species comparisons among all available vertebrate genomes, such as human, mouse, chimp, chicken, zebrafish, tetraodon, and fugu.

This manuscript is organized as follows. The “Materials and Methods” section introduces the workflow for identifying conserved noncoding regions and provides a description of mapping tasks onto a computational grid. The “Experiments and Results” section demonstrates the efficacy of the grid-enabled workflow. Finally, the “Conclusions” section ends with an outlook of future work.
**Materials and Methods**

**Steps in Identifying Conserved Noncoding Regions**

1) **Annotations of vertebrate sequences**: The Ensembl project [9] provides a comprehensive and integrated source of annotations of large genome sequences. It employs MySQL relational databases to store sequence information and provides a comprehensive set of application program interfaces (APIs) to serve as a middle layer between the database and application programs. The vertebrate genomic sequences obtained from the Ensembl database are stored, and the necessary (BioPerl) APIs are installed locally.

2) **Extraction of noncoding sequences**: Noncoding DNA segments include introns, UTRs, pseudogenes, and intergenic regions, which code for neither proteins nor stable RNA [7]. We merge all noncoding regions into a single sequence and supplement with the information of the corresponding organism, chromosome number, noncoding region type (i.e., 5-UTR, introns, 3-UTR, or intergenic regions), and starting and ending locations.

3) **Sequence alignment**: We divide or concatenate noncoding sequences to form segments of length \( L \) Mb/s, allowing for gaps and overlaps, say \( \Delta \) b/s, between adjoining segments. The overlapping lengths should be chosen sufficient enough to extract conserved regions given a maximum length \( d \) b/s. BLASTZ, an extension to BLAST [14] pairwise local alignment tool, was used as an alignment tool [10], [11]. BLASTZ is found to be suitable for aligning long noncoding sequences that have evolved neutrally [15]. Based on a three-step strategy [14], [16], BLASTZ first removes lineage-specific interspersed repeats from each sequence and searches for short near-perfect matches between the two sequences [11]. Each match is then extended gap-free, using dynamic programming, until the alignment score remains above a specified threshold.

4) **Detecting conserved noncoding regions**: A criteria to identify conserved regions from BLASTZ alignment needs to be defined: for example, gap-free alignments of at least 70% identity and at least 100 b/s in length have been widely used for identifying the conserved regions [3]–[6], [8], [13], [17]. Since the alignments are usually weaker in the noncoding regions, we additionally searched for all conserved regions complying with gap-free alignments of at least 50% and at least 20 b/s in length. This provides us with four versions of conserved noncoding regions in our database: \( q = 50\% \) or 70\%, and \( d = 20 \) or 100 b/s.

5) **Database of conserved noncoding regions**: The conserved noncoding regions are imported into a custom-designed database along with the information about their genomes and locations. The database can be accessed through a user-friendly Web interface, which allows users not only to the data but also to compare any two chromosomes of two vertebrates and find the conserved regions by setting the own alignment and length parameters.

**Grid-Enabled Pipeline**

Sequence alignment by using BLASTZ accounts for most of the running time of the workflow. We use a computational grid to speed up the alignment process, which allows one to utilize the distributed computing and memory resources at a reasonable cost [18], [19]. In a computational grid, heterogeneous computing servers and data are distributed at distinct locations connected through Intranet/Internet. A major challenge in grid computing is how to efficiently find available resources and data and distribute the tasks among the processing nodes. The development of applications in a grid-computing environment needs to handle the following characteristics of resources [12].

- **Computing servers having different computing powers**: For example, computing machines on the grid may have processors having different speed and architecture, memory, storage, and software platform.

- **Sharing of resources**: The grid should allow for several users’ tasks running at the same time; the effective usage
of CPUs in an application depends on the number of jobs running at a computing node at one time.

- *The different network infrastructures that link grid resources:* They may have different protocol and performance standards.

To parallelize an application efficiently on a computational grid, the implementation should minimize data transfer on low-speed connections and allocate the tasks to a node that can offer the required computational power to the application at that time. This assures that no processor or connection would become a bottleneck to an application on the grid.

We illustrate a grid-enabled implementation of comparative genomics pipeline in Figure 1 with Grid-BLASTZ [18]. The parallelization of the sequence alignment in BLASTZ is embedded in the workflow such that extracted noncoding sequences can be processed simultaneously at different computing nodes. The lengths of the input segments determine the granularity of sub-tasks, and the length of overlaps between the adjacent segments determines the shortest length of target-conserved segments.

Previously, a static load balancing approach was presented in [19] for mapping BLAST applications [16] onto grid architectures. However, it achieves reasonable performance only when the execution of a task is uninterrupted by other applications running at that node. Therefore, we have chosen a dynamic load balancing approach for the grid implementation so that grid-based applications are able to scale well even when there are other applications running over the resources in the grid. With dynamic load balancing, jobs are allocated dynamically to idle nodes to keep all nodes busy and to minimize the overhead of the load balancer. Dynamic load balancing has been increasingly used in several bioinformatics applications, which works similar to a server–client model [20]. For example, fastDNAmL [21] and a profile hidden Markov model software (HMMER) [22] have reported an increased efficiency by using a dynamic load-balancing strategy.

As shown in Figure 1, the processes are created by invoking the tasks [23]. The grid-task database serves as the master process that stores the descriptions of all BLASTZ processes for comparative tasks. A typical task would be to compare a specific segment in mouse genome with a specified segment in a human genome. The slaves, such as computing nodes, keep retrieving exclusively task descriptions from the master once they are free and then execute the corresponding BLASTZ tasks until all subtasks are completed.

**Experiments and Results**

**Testbed**

A testbed comprising of three Linux clusters and a database server, at two research centers at the Nanyang Technological University (NTU), Singapore, was used to test the performance of the workflow. Two clusters are located at the Bioinformatics Research Center (BIRC), one with eight nodes of Intel Itanium-1 733-MHz processors with 1 GB internal memory and the other with four nodes of advanced microprocessors (AMDS) Opteron 2.60-GHz processor with 8 GB internal memory. The third cluster is situated at the parallel and distributed computing center (PDCC), consisting of eight nodes of Intel Xeon 2.60-GHz processor with 1 GB internal memory. The clusters are internally connected with a Myrinet. An Ethernet switch was used as an intercluster connection. The grid middleware on the clusters was GLOBUS 4.0.5 [23]. The sun grid engine (SGE) [24] was installed as a distributed resource management (DRM) software, which allocates grid tasks from the control node to execution nodes inside a cluster. The grid-task database and conserved noncoding database were stored on a Sun Fire X4200 server at BIRC.

**Conserved Noncoding Sequences**

Here we present the results of comparison of noncoding sequences extracted from human chromosome 21 and mouse chromosome 16 noncoding sequences. The human chromosome 21 was split into 12 noncoding segments of length $L = 5$ Mb/s with an overlap $\Delta = 100$ b/s allowed between adjacent segments. Longer segment length may slightly reduce the number of overlapping nucleotides but increases the computational demands on the grid and the turnaround time due to data transmission. This would decrease the grid throughput and likely make the grid less sensitive to disturbances. As we are interested in extracting conserved segments in noncoding regions of at least 20 b/s, these settings are adequate to guarantee that no candidate-conserved segment is missed. The mouse chromosome 16 was split into 21 noncoding segments of the same configuration. The UTRs were also considered as noncoding sequences and extracted as candidate sequences for aligning. Previously, the Penn State, Berkeley, and the University of California, Santa Cruz (UCSC)-conserved noncoding sequence projects [13] used the Human Dec 2001 and MGSCV3 flat files as the source sequences. The complete UTRs are not all represented in these flat files. Moreover, repetitive sequences in these flat files are represented with lowercase characters and would be masked by alignment tools such as BLASTZ. These numbers of noncoding segments made a total of 252 subtasks for the grid. In BLASTZ, the threshold for maximal scoring segment pair (MSP) represents the sensitivity factor. For restricting gaps in alignments to be at least 100 b/s with the identity as low as 50%, where longer gaps are not harshly penalized, we set the MSP threshold at 3,400 [10].

The conserved noncoding segments in the database are associated with adequate supplementary information. Table 1 shows the examples of conserved noncoding regions detected by comparative genomics of human and mouse.

<table>
<thead>
<tr>
<th>Human</th>
<th>Mouse</th>
</tr>
</thead>
<tbody>
<tr>
<td>Region</td>
<td>Chromosome</td>
</tr>
<tr>
<td>14403123–14403392</td>
<td>21</td>
</tr>
<tr>
<td>16024234–16024284</td>
<td>21</td>
</tr>
<tr>
<td>13366269–13366445</td>
<td>21</td>
</tr>
<tr>
<td>15068493–15068565</td>
<td>21</td>
</tr>
<tr>
<td>16833280–16833542</td>
<td>21</td>
</tr>
</tbody>
</table>

Table 1. Examples of conserved noncoding regions detected by comparative genomics of human and mouse.
shows the content of example records in the database, resulting from the comparison of human and mouse genomes. The region column represents the actual locations of the conserved regions on the chromosome indicated by the next column. The type column indicates which functional region the corresponding conserved region belongs to.

Table 2 shows the statistics of conserved regions found by the comparative genomic workflow by using two sets of parameters: 1) at least 50% identity and 20 b/s length \((q = 50\%, l = 20 \text{ b/s})\) and 2) at least 70% identity and 100 b/s length \((q = 70\%, l = 100 \text{ b/s})\). The number of conserved noncoding regions obtained was 22,791 and 3,209, respectively. With a stricter criterion, the number of conserved regions dropped drastically. As seen, most conserved noncoding regions were from the human introns of chromosome 21 and mouse introns of chromosome 16 and characterized by a low guanine cytosine (GC) content of about 36–37%. A very small number of conserved segments were found in regions of the human 5-UTR and intergenic regions, despite its large size, compared with other mouse noncoding regions.

Compared to the existing conserved noncoding databases in the Penn State database [13], the segments identified by our pipeline for human and mouse genomes were quite different even when the same selective criterion and alignment tool (BLASTZ) were used. There are two reasons to this: our pipeline included UTRs as candidate sequences for alignment; and we used the latest versions of sequences and annotation information from the Ensembl (version 36, 2006). The Penn State project used older Human Dec 2001 and MGSCv3 flat files. Further, the annotations used are different: for instance, the length of human chromosome 21 in the Ensembl (version 36) is 46,944,323, whereas the length in Human Dec 2001 is 44,620,379.

**Performance**

We investigated the speedups of the parallel implementation of the comparative genomic pipeline on the grid. Figure 2 shows speedups of our grid-enabled pipeline with and without disturbances. The speedup of a cluster is measured by the sum of the running time of each processor in a corresponding fraction of tasks, divided by the running time when multiple processes are equally distributed in the cluster [18]. A disturbance is simulated by running an application in one of the 2.60-GHz CPU. It can be seen that the grid-enabled pipeline achieved significant speed improvements even under system disturbances.

As the computing resources on the grid are shared by all users, the execution of a job is usually disturbed by another

---

**Table 2. Statistics of noncoding segments conserved in human and mouse genomes with at least \( l \text{ b/s} \) long and \( q\% \text{ identity}.**

<table>
<thead>
<tr>
<th>Category</th>
<th>No. of Regions</th>
<th>Mean Length</th>
<th>GC%</th>
<th>No. of Regions</th>
<th>Mean Length</th>
<th>GC%</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Human</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5-UTR</td>
<td>Mouse 101</td>
<td>132</td>
<td>58</td>
<td>5-UTR 31</td>
<td>236</td>
<td>49</td>
</tr>
<tr>
<td>3-UTR</td>
<td>24</td>
<td>100</td>
<td>53</td>
<td>6</td>
<td>198</td>
<td>55</td>
</tr>
<tr>
<td>Intron</td>
<td>21,604</td>
<td>88</td>
<td>36</td>
<td>3,050</td>
<td>151</td>
<td>37</td>
</tr>
<tr>
<td>Intergenic</td>
<td>51</td>
<td>96</td>
<td>43</td>
<td>191</td>
<td>40</td>
<td></td>
</tr>
<tr>
<td><strong>Intron</strong></td>
<td>7</td>
<td>109</td>
<td>45</td>
<td>2</td>
<td>229</td>
<td>45</td>
</tr>
<tr>
<td>3-UTR</td>
<td>68</td>
<td>80</td>
<td>38</td>
<td>7</td>
<td>121</td>
<td>39</td>
</tr>
<tr>
<td>Intron</td>
<td>21,604</td>
<td>88</td>
<td>36</td>
<td>3,050</td>
<td>151</td>
<td>37</td>
</tr>
<tr>
<td>Intergenic</td>
<td>51</td>
<td>96</td>
<td>43</td>
<td>191</td>
<td>40</td>
<td></td>
</tr>
<tr>
<td><strong>Intergenic</strong></td>
<td>1</td>
<td>59</td>
<td>58</td>
<td>2</td>
<td>198</td>
<td>50</td>
</tr>
<tr>
<td>3-UTR</td>
<td>6</td>
<td>72</td>
<td>53</td>
<td>1</td>
<td>113</td>
<td>68</td>
</tr>
<tr>
<td>Intron</td>
<td>1</td>
<td>97</td>
<td>52</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Intergenic</td>
<td>3</td>
<td>154</td>
<td>46</td>
<td>2</td>
<td>198</td>
<td>50</td>
</tr>
<tr>
<td><strong>Total</strong></td>
<td>22,741</td>
<td>88</td>
<td>37</td>
<td>3,209</td>
<td>153</td>
<td>37</td>
</tr>
</tbody>
</table>
application running at a node. A good implementation should be robust to such perturbations. We measure the extent of the effect by a disturbance on the performance of an application with the performance degradation ratio under disturbance (PDRD) [22]. Smaller PDRD values indicate a better robustness of the application to disturbances. Figure 3 shows the PDRD of our grid-enabled pipeline in comparison to the grid-enabled BLAST pipeline employing the static load balancing for job allocation. As seen, the use of dynamic load balancing has made the pipeline more robust to the disturbances on the grid.

Conclusions

Comparative genomics compares DNA sequences of different species at genome scale to discover preserved patterns across the species. The conserved patterns have evolved naturally and therefore should relate to an important biological function. Comparative genomics is useful in detecting novel genes or sequence segments that are likely to have regulatory functions. Here we present our preliminary work on building a comprehensive database of conserved noncoding segments across vertebrates. Such a database may contain yet undiscovered regulatory segments that researchers can look for regulatory function.

When large genomes like vertebrates are involved, comparative genomics applications become compute-intensive and therefore the need for high-computing resources and platforms arises. To increase the efficiency and speed of comparative genomics applications, we presented an implementation of Grid-BLASTZ and grid-enabled comparative genomic workflow. Our aim is to identify conserved noncoding regions in vertebrate genomes, using already-available computational resources spanning across different sites, which may be distant located. As seen, the proposed grid implementation considerably speeded up the comparative workflow. Our implementation was robust to the disturbances by other applications in the grid and highly efficient in an environment composed of diverse resources with heterogeneous characteristics.

We created a relational database to store conserved segments, including additional information such as location reference and prior knowledge about conserved noncoding regions. Our workflow is incremental in the sense that the database can be updated when new vertebrate genomic sequences become available. Presently, the database contains conserved segments obtained from mouse and human genomes, but in future, other vertebrates such as chimp, chicken, zebrafish, tetraodon, and fugu will be added. In addition to the conserved segments generated, we will allow researchers to compare chromosomes across genomes at different similarity and length parameters.

Our future work will involve designing a methodology for characterizing the conserved noncoding regions, which could be performed by genome-wide statistical and computational analysis. It involves finding how statistically significant these potential regulatory elements present within and across genomes, assessing the frequency of all elements in the databases across genomes, finding the similarity among elements, and creating new potential elements by merging the elements that are similar. These findings can be verified by using previously studied elements and regions.
to assign the biological functions to conserved noncoding sequences. If closely related elements have not been previously found, the help of wet-lab experiments is needed.

Further, comparative genomics allows us to build phylogenetic profiles and evolutionary rates of sequences. The segments with functional importance evolve at comparatively lower rate. Also, the conserved segments could be analyzed against the features of protein-coding sequences to investigate their functionality. For example, it has recently been found that codon-usage bias is different in different genes [25]. Our finding may be compared across the existing databases of regulatory elements for possible role of their regulatory functions. However, testing for possible regulatory functions of conserved noncoding sequences in wet-lab experiments will eventually be necessary before making strong conclusions about their functions.

Acknowledgments
The work is partly supported by grants to J.C. Rajapakse, by the Biomedical Research Council (grant no. 04/1/22/19/376) of Agency of Science and Technology Research, administered through the National Grid Office, Singapore, and by Sun Microsystems, Inc., USA.

Jagath C. Rajapakse received his M.Sc. and Ph.D. degrees in electrical and computer engineering from the University of Buffalo (USA). He was a visiting fellow at the National Institute of Mental Health (USA) and a visiting scientist at the Max Planck Institute of Brain and Cognitive Sciences (Germany). Currently, he is a professor of computer engineering and the director of Bioinformatics Research Center, Nanyang Technological University (NTU), Singapore. He is a visiting professor to the Department of Biological Engineering, Massachusetts Institute of Technology (MIT). His research interests include bioinformatics, informatics, modeling brain connectivity through functional brain imaging, and building pathways from gene and protein expressions obtained by microarrays and high-content microscopy. He has authored or coauthored more than 225 peer-reviewed journal and conference papers and book chapters and was listed among the most cited scientists. He serves as an associate editor of *IEEE Transactions on Medical Imaging*, *IEEE Transactions on Computational Biology and Informatics*, and *IEEE Transactions on Neural Networks* and in editorial boards of several other journals.

Pooja received her B.Sc. degree in computer science from Birla Institute of Technology, India, and her M.Sc. (bioinformatics) degree from NTU, Singapore. She worked as a research associate in BIRC at NTU. She is now working as a lead software developer at ITN, London.

Chunxi Chen received his Ph.D. degree in computer science from NTU, Singapore. His research interests include parallel and distributed computing and sequence alignment. He was with the BIRC, School of Computer Engineering, NTU.

Sy-Loi Ho received his B.Sc. degree in computer science from the Vietnam National University, Hanoi, in 2000 and Ph.D. degree in computer engineering from NTU, Singapore, in 2005. Since January 2005, he has been working in the BIRC, NTU, where he is currently a research fellow. He has authored more than ten research papers in the area of bioinformatics, including computational techniques, to detect various signals and motifs in biological sequences and comparative genomics.

Address for Correspondence: Jagath C. Rajapakse, Bioinformatics Research Center, Nanyang Technological University, Singapore 639798. E-mail: asjagath@ntu.edu.sg.

References