Secret Sequence Comparison on Public Grid Computing Resources

K.I. Kurata, H. Nakamura, Vincent Breton

Secret Sequence Comparison on Public Grid Computing Resources

Ken-ichi Kurata and Hiroshi Nakamura
Research Center for Advanced Science and Technology,
The University of Tokyo,
4-6-1 Komaba, Meguro-ku, Tokyo 153-8904, Japan
Email: {kurata, nakamura }@hal.rcast.u-tokyo.ac.jp

Vincent Breton
Laboratoire de Physique Corpusculaire de Clermont-Ferrand,
Centre National de la Recherche Scientifique,
24 avenue des Landais, 63177 Aubiere, France
Email: breton@clermont.in2p3.fr

Abstract

Once a new gene has been sequenced, it must be verified whether or not it is similar to previously sequenced genes. In many cases, the organization that sequenced a potentially novel gene needs to keep the sequence itself in confidence. However, to compare the potentially novel sequence with known sequences, it must either be sent as a query to public databases, or these databases must be downloaded onto a local computer. In both cases, the potentially new sequence is exposed to the public. In this work, we propose a novel method to compare sequences without any exact sequence information leaks to the public. This method is based on our previous proposed method [1] to find unique sequences on grid computing environments, which is well-parallelized in reasonable performance. In order to keep the exact sequence information in confidence, this method samples intervals (subsequences) from a sequence, and these intervals are hashed. Any key cryptosystem is not used. The hashed data are open to the public to verify the novelty of the sequence. The experimental results for 19797 h.sapiens genes show that the parallel implementation of this method performs reasonably well in terms of speed and memory usage. In this paper, the implementation on the world-wide testbeds of European Data Grid (EDG) and its results are described.

1 Introduction

In the field of molecular biology, it is indispensable to discover new genes related to biologically important reactions. Once such genes have been sequenced, it is necessary to be verified whether the sequences are already reported or not. In order to verify the novelty of the sequences, in general, this verification process involves searching for matches in databases by means of a homology search program.

The comparison process using such a homology search program requires the exact sequences be exposed to public scrutiny. Thus, if it is desired to keep the putative novel sequence confidential, then an alternative approach is conceived to verify its novelty. The simple alternative approach is to gather all databases from all around the world into a local computer. Next, the verification process is executed in-house. However, some of these databases include confidential information, such as private medical information and/or precious experimental information. If you would like to touch such information, you have to establish a contract to keep the information in secret. Furthermore, some of the databases related to business are not freely available to anonymous researchers. Accessing to these databases requires quite a few costs. However, in general, all the thing we would like to do after having discovered a new sequence is to verify whether the sequence is already reported or not, and to estimate the similarity to known gene sequences.

On the other hand, the size of the genomic databases around the world is growing at an exponential pace. It is becoming intractable to gather all the databases in local. Therefore, it is unavoidable to propose a parallel calculation method implemented in a distributed computing environment. Due to the development of the infrastructure of world-wide high-speed networks, it is becoming feasible to deploy a distributed computing environment on the Internet. One of the projects that realize this type of computing environment is called the European Data Grid (EDG)

project [2]. Providing the infrastructure and tools that make large-scale, secure resource sharing possible and straightforward is the Grid's raison d'etre [3].

In this work, we implement a novel method to secretly compare sequences without any key cryptosystem on a public grid computing environment. First of all, our proposed sampling method called Interval Sampling, [4] which inhibits the reconstruction of exact sequences, is explained. Next, this method is gridified through our proposed technique [1] and sequences are compared in parallel without any exact information on the international EDG testbeds.

2 Background

In order to verify whether a sequence is already registered or not, we would like to propose a sequence comparison method which works without exposing the exact sequence of the target genes. Moreover, this method must work in parallel in a distributed computing environment.

In this section, at first, two traditional sequence comparison models are shown. The weak points of these models are discussed. Next, we propose a computation model to resolve such weak points. In this model, sequences are secretly compared in parallel. We present our previous effort to tackle to this challenge. Finally, one of the best solutions, dubbed Interval Sampling, is described.

2.1 Traditional sequence comparison models

A new sequence is sent onto the computers near target databases. Each computer gathers registered gene sequences from its target databases. The comparisons then are processed in parallel. Figure 1 shows this calculation model.

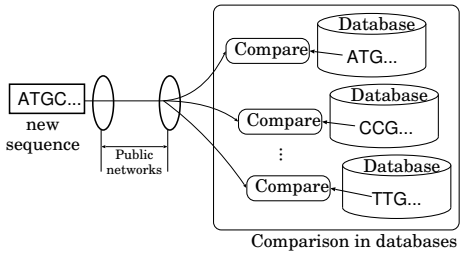

Figure 1. When public computer resources are used to compare sequences, the exact sequence information must be opened on the computers.

In this method, regrettably, the new sequence must be exposed through public networks. It is possible to send the encrypted sequence securely through public networks. However, secret or public key distribution takes a risk of exposure, if the key is not managed properly. Moreover, if you use public computer resources to compare sequences, the exact sequence information must be decoded, and becomes available on public computer resources. Therefore, it is desirable to compare sequences without any key cryptosystem.

On the other hand, if you gather into local all gene sequences on the target databases to keep in secret their exact sequence information, great amounts of network loads and local computing resources are required. Even if broadband networks between the local site and the databases are available with powerful computing resources, all the target databases must be opened. Some of these databases may have confidential information. In order to keep it in confidence, they are not freely available to anonymous researchers without any contract. Figure 2 shows this calculation model.

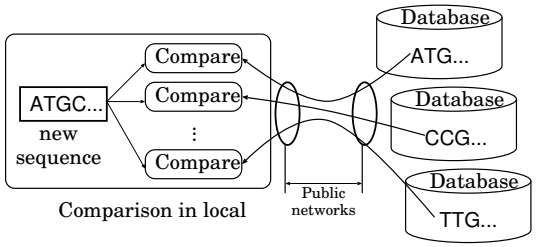

Figure 2. When all gene sequences on the target databases are gathered into a local computer , great amounts of network loads and local computing resources are required. Moreover, all the target databases must be opened.

2.2 Secret sequence comparison model

In order to settle the weak points of the traditional sequence comparison models, the following model in Figure 3 is proposed.

Suppose that there exists a one-way process by means of which a sequence is easy to transform but hard to reconstruct, and the processed data have information enough to verify the uniqueness. In other words, a sequence is encoded into a different form, by a straightforward process. On the other hand, the reconstruction of the sequence from these processed data is very difficult. In order to realize this one-way process, we use the Interval Sampling function and the hash function explained later in this article.

Moreover, suppose that the comparison job can be divided into n jobs, each of which can be processed individu-

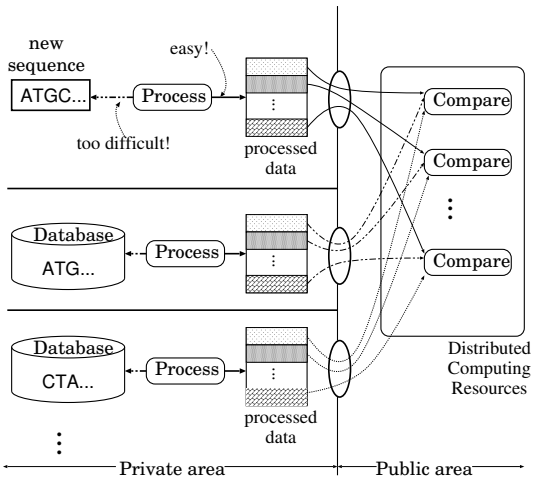

Figure 3. Secret sequence comparison using a distributed computing environment. Suppose that there exists a one-way process by means of which the sequence is easy to transform but hard to reconstruct, and the processed data have enough information to verify (or not) the novelty of the exact sequence. Moreover, suppose that the comparison job can be divided into n jobs, each of which can be processed individually. In that case, we do not need to distribute the exact sequence, but only its processed form. And then, the comparison can be executed on public computer resources by using the processed data. Moreover, since each part of the processed data can be calculated individually, the comparison processes can be launched in parallel in distributed computing environments.

ally. This job division is also realized by the hashing function. If unique sequences are discovered from these processed data, it is possible to confidentially verify the novelty of the sequence in parallel. In order to find unique sequences, we use the sort function explained in the algorithm section. This algorithm is detailed in our previous work [1].

In our proposed method, it becomes unnecessary to expose the exact sequence to the public; we only need to expose its processed data, from which it is very hard to reconstruct the exact sequence. The comparison can therefore be executed on public computer resources by using the processed data. Since each part of the processed data can be calculated individually, the comparison processes can be launched in parallel in distributed computing environments.

2.3 Our previous study

We proposed an algorithm to find unique sequences on target genome in reasonable memory size and computing performance [5]. Unique sequences are short sequences which exist only once on the target genes. We also described how to parallelize this algorithm and implement it onto a distributed computing environment. The experimental result on a world-wide grid environment was reported in our previous work [1]. Furthermore, we have proposed a method called Artificial Mutation and Splicing (AMS) site insertion to prevent the reconstruction of the exact sequences [6]. In this method, by means of inserting n sites of artificial mutation and splicing (AMS) into the exact sequences, the number of combinations of the reconstruction is multiplied by 2^n. But this method has mainly two weak points. The first is that the total file size grows in proportion to the number of AMS sites, n. The second is that the reconstructed sequences have high similarity to the exact sequence. Taking into consideration these weak points, Interval Sampling (IS) method has been proposed [4]. After IS is applied, the size of the sequence does not increase. Moreover, the similarity between the reconstructed sequences and the exact sequence is very low.

In this work, the implementation of this IS method on a grid computing environment and the experimental results are presented.

3 Algorithm

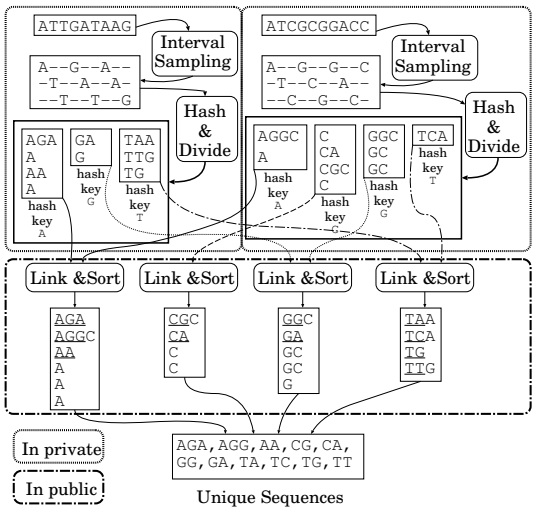

Figure 4. Work flow of secret sequence comparison.

Figure 4 shows the work flow of our proposed algo-

rithm. First of all, the exact gene sequence is processed by means of the Interval Sampling method. Second, sampled sequence data are hashed and divided on the basis of their own hash-key. Third, all the hashed data whose hash-key is the same are linked and sorted. Finally, the novelty of the gene sequence is verified by using the information of unique sequences.

Now, an example is given. In this figure, there are two input sequences, ATTGATAAG and ATCGCGGACC. Given the condition that Interval I is 3, after the former sequence is processed by IS, the subsequences AGA, TAA and TTG are produced. After the latter sequence is processed, the subsequences AGGC, TCA and CGC are produced. Next, the sequences AGA, TAA and TTG, which were produced by IS from ATTGATAAG, are processed by means of the Hash & Divide function. On condition that hash-key length is 1, there exist AGA and A, which have hash-key A at the left end, and GA, which has hash-key G at the left end, as a partial sequence of AGA. In the same way, by applying the hash function to TAA and TTG, it is confirmed that there are the partial sequences, AGA, A, and AA having hash-key A, the sequences GA and G having hash-key G, and the sequences TAA, TTG, and TG having hash-key T. Next, the Link & Sort function is applied. There are the sequences AGA, A, AA and A, which have hash-key A at the left end, on the left hand of the figure. These sequences come from ATTGATAAG. On the right hand, there are the sequences AGGC and A, which have hash-key A at the left end. Now, the link function gathers these sequences into the leftmost box. These sequences are sorted and the unique sequences on these target genes, AGA, AGG and AA, are output. In the same way, the link and sort function is executed for the hash-key sequences, C, G and T. Finally, the novelty of the target gene sequence can be verified. If one sequence exists only once on all the target gene sequences, the gene including this sequence exists only once among the target genes and is unique. Therefore, we can determine the novelty of gene sequences by finding the unique sequences on them.

Hereafter, each process is detailed.

3.1 Interval Sampling (IS)

Suppose that there is a gene sequence, $s, s(j) \in \{a, c, g, t\}, j = 0 \dots l-1$, whose length is l. Now, Interval Sampling $f(s, I, x)$ is defined as follows.

$$
\begin{aligned}
y_{(I,x)} &= f(s, I, x) \\
&= \begin{cases} s(x)s(x+I)s(x+2I)\dots s(x+(\lfloor \frac{l}{I} \rfloor - 1)I) \\ \quad : x > m \\ s(x)s(x+I)s(x+2I)\dots s(x+\lfloor \frac{l}{I} \rfloor I) \\ \quad : x \leq m \end{cases}
\end{aligned}
$$

Here, s is an input sequence, whose length is l. $m = l$ modulo I. I is an interval value. x is a variable, whose range is $0 \dots I-1$. $y_{(I,x)}$ is the output sequence.

After the Interval Sampling method is applied, I output sequences $y_{(I,x)}, x = 0, \dots I-1$ are produced. m of the sequences have the length of $\lfloor \frac{l}{I} \rfloor$. $I - m$ of them have the length of $\lfloor \frac{l}{I} \rfloor - 1$. In other words, this method samples every I bases and makes I partial sequences.

Here we consider the reconstruction of the exact sequence. Suppose that only the sequences processed by IS are available. In this situation, the number of possible reconstructions is given by ${}_{I-m}P_{I-m} \times_m P_m$, because the sequences whose lengths are the same can be permuted arbitrarily.

Let us see the example in Figure 4. In this figure, there are two input sequences, ATTGATAAG and ATCGCGGACC. On condition that Interval I is 3, after the former sequence is processed by IS, AGA, TAA and TTG are produced. The length of the exact sequence is 9. Thus, $m = 9$ modulo $3 = 0$. When we try to reconstruct the exact sequence, there are ${}_3P_3 = 6$ possibilities, TATTGAGAA, TTATAGGAA, TATAGTAAG, TTAATGAGA, ATTGTAAGA and ATTGATAAG. After the latter sequence is processed, AGGC, TCA and CGC are produced. Now, the length of the sequence is 10. Thus, $m = 10$ modulo $3 = 1$. When we try to reconstruct the original sequence, there are ${}_2P_2 \times_1 P_1 = 2$ possibilities, ATCGCGGACC and ACTGGCGCAC.

3.2 Hash & Divide

The Hash & Divide function is detailed in our previous work [1]. Using the sequence from the left end of the partial sequences on target sequences as their hash-key, all the partial sequences are hashed. And then, the total file can be divided up to the number of the hash-key. This process is of $O(n)$ when n is the target gene file size. After a sequence is hashed, the length of partial sequences is limited by threshold θ. In other words, the length is equal to or less than θ.

The size of hashed sequences is proportional to θ. Thus, the file size becomes approximately θn. Namely, the file size becomes θ times larger than that of the input sequence.

Here, this function is explained through Figure 4. Let us note the sequences AGA, TAA and TTG, which are produced by IS from sequence ATTGATAAG. There exist AGA, GA and A as the partial sequences on AGA. On condition that the hash-key length is 1, there exist AGA and A, which have hash-key A at the left end, and GA, which has hash-key G at the left end, as the partial sequences on AGA. In the same manner, by applying the hash function to TAA and TTG, it is confirmed that there are the partial sequences, AGA, A, AA, A having hash-key A, the sequences GA, G having hash-key G, and the sequences TAA, TTG, TG having hash-key T.

Table 1. Sequences processed by IS and their subsequences.

ATTGATAAG	
Processed by IS	subsequences
AGA	AGA, GA, A
TAA	TAA, AA, A
TTG	TTG, TG, G
ATCGCGGACC	
Processed by IS	subsequences
AGGC	AGGC, GGC, GC, C
TCA	TCA, CA, A
CGC	CGC, GC, C

Table 2. Subsequences classified on the basis of hash-key.

	Hash-key	Hash-key	Hash-key	Hash-key
	A	C	G	T
AGA	A–GA A–		G–A	
TAA	A–A A–			T–AA
TTG			G–	T–TG T–G
AGGC	A–GGC	C–	G–GC G–C	
TCA	A–	C–A		T–CA
CGC		C–GC C–	G–C	

3.3 Link & Sort

The Link & Sort function is detailed in our previous work [1]. The link function gathers all the subsequences having the same hash-key. The sort function launches the radix-sort-like algorithm we have proposed. This process is of $O(n \log n)$ when the total file size is n.

Let us consider the example in Figure 4. There are the subsequences AGA, A, AA and A, which have hash-key A at the left end, on the left hand of the figure. These subsequences come from ATTGATAAG. On the right-hand, there are the subsequences AGGC and A, which have hash-key A at the left end. These subsequences come from ATCGCGGACC. Now, the link function gathers these subsequences into the leftmost box. In this box, there are AGA, AGGC, AA, A, A and A. These subsequences are sorted. At first, the character, which is on the right hand of the hash-key A, of the subsequences is compared. The character next to the hash-key on AGA, AGGC is G. Hence, these subsequences are grouped together. Next, the second character from the hash-key is compared in this group. The second character on AGA is A. The second character on AGGC is G. Now, these subsequences are separated on the basis of their second character. In each group, there exists one subsequence. Therefore, the unique subsequences AGA, AGG are output. In the same way, the link and sort function is executed for the hash-key sequences, C, G and T.

Table 3. Sorted subsequences

Hash-key	Hash-key	Hash-key	Hash-key
A	C	G	T
AG–A	CG–C	GG–C	TA–A
AG–GC	CA–	GA–	TC–A
AA–	C	GC	TG–
A	C	GC	TT–G
A		G	
A			

3.4 Novelty of sequences

If some sequence exists only once on all the target genes, the gene including this sequence exists only once on all the target genes, and is unique. Therefore, we can determine the novelty of a gene by finding the unique sequences on it. For instance, TT is a unique sequence in Figure 4. It is part of TTG. TTG comes from ATTGATAAG by means of IS. Since ATTGATAAG has a unique sequence on itself, this sequence is unique.

On the other hand, it is possible that there exists a gene, which has no unique sequences on itself, but which is unique as a whole against all the other target genes. Hence, if no unique sequences are found on a gene by means of this method, it can not be judged whether such a gene is unique or not.

In a nutshell, if at least one of the partial sequences of a target gene is unique, the target gene is unique as a whole. To the contrary, if any of the partial sequences are not unique, it can not be judged whether the target gene is unique or not.

4 Experiments

In this section, an experimental result on the EDG testbeds is shown. The environment of this experiment is as follows. The data sets for the experiments were gathered from the KEGG database [7]. The experiments were done

on condition of Interval $I = 15$, hash-key length = 7 and threshold $\theta = 30$. Thus, the length of the hashed sequences was limited by 30. In other words, unique sequences whose length is more than 30 can not be found.

Target genes were stored on a node of the European Organization for Nuclear Research (CERN) in Switzerland. The genomic database was located at Research Center for Advanced Science and Technology (RCAST) in the University of Tokyo in Japan. The computing nodes at RCAST were used as parallel grid computing resources. This site consisted of one Pentium III 1GHz 608 MB, six Pentium III 1.2GHz 512 MB and two Pentium 4 2GHz 1GB machines. All the machines were connected onto the local network of 100 Mbps.

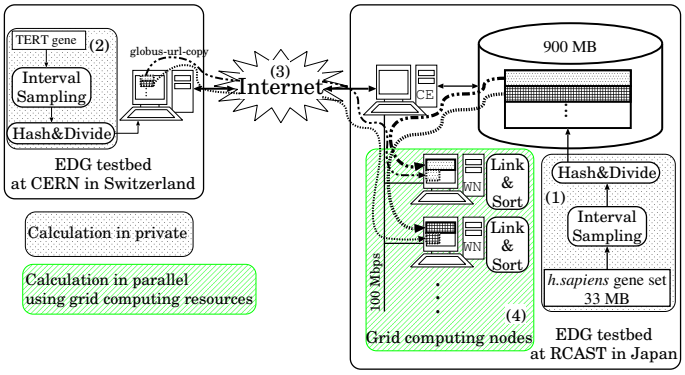

Figure 5. Secret sequence comparison on a world-wide grid computing environment. (1) *h.sapiens* gene set is processed by IS and hashed at RCAST. (2) TERT gene is processed by IS and hashed at CERN. (3) The processed files of TERT are sent through the Internet to RCAST. (4) The processed files of both TERT gene and *h.sapiens* gene set are linked and sorted in parallel on the grid computing nodes at RCAST.

Figure 5 describes the calculation environment of comparing sequences in secret.

(1) First of all, the exact sequences of a *h.sapiens* gene set were processed by IS and hashed at RCAST. The file size of the *h.sapiens* gene set, which consisted of 19796 gene sequences, was 33 MB. This gene set did not include the target gene described as follows. After the gene set had been processed by the IS method and the hash function, the size became about 900 MB. This process spent about 22 minutes on a Pentium 4 machine. The processed data were stored on the database at RCAST and opened to the public. After the above process is done only once, a variety of combinations of sequence comparison can be launched.

Here, the novelty of a target gene, which is telomerase reverse transcriptase (TERT) of *h.sapiens*, was verified. (2) At first, this gene was processed by the IS method and hashed within a few seconds on a node at CERN. After processed, the processed file size became 245 KB. Next, this processed file was divided into eight processed files based on their hash-key. (3) These processed files were simultaneously sent to the nodes at RCAST through the Internet by means of *globus-url-copy* command spending less than 10 s. (4) 8 jobs were launched in parallel to the 8 nodes at RCAST. Each node linked one of the 8 processed files of TERT to its corresponding file of the *h.sapiens* gene set and sorted it. This link and sort function was done in parallel through *globus-job-run* command and spent 8 minutes in total.

Table 4. Calculation time and data size.

	TERT	*H.sapiens* gene set
Process a priori		
Raw data size	3.5 KB	33 MB
Hash & Divide	a few seconds	22 minutes
After-hashed data size	245 KB	900 MB
On-demand process: 8 jobs in parallel		
Globus-url-copy from CERN to RCAST for TERT	10 seconds	
Link & Sort	8 minutes	

The *h.sapiens* gene set did not include TERT. As a result, unique sequences were successfully found. The average length of unique sequence was less than 14. Therefore, it was verified that some partial sequences produced from the TERT gene never appeared on the *h.sapiens* gene set. In other words, the TERT gene was unique on the *h.sapiens* gene set. In addition, the average length said that the similarity between the TERT gene and the other *h.sapiens* genes was very low. The uniqueness of 99 % of the *h.sapiens* gene set was verified on this condition.

Table 5 shows the annotation of the *h.sapiens* genes having some unique sequences on condition of Interval value of 15, but no unique sequence on condition of Interval value of 1. When Interval value is 1, the sampled sequences are consecutive on the target genes. Therefore, any short unique sequence as a primer and/or probe can not be designed for these genes.

In this experiment, the length of 30 was assigned as the threshold of after-hashed sequences. When the value of Interval was 15, 30 characters were sampled every 15 bases from the partial sequences whose length was 450.

Table 5. *H.sapiens* **genes having some unique sequences on condition of Interval value = 15, but no unique sequence on condition of Interval value = 1.**

H.sapiens genes	
hsa:2574	GAGE2; G antigen 2
hsa:9426	CDY2; chromodomain protein, Y-linked, 2 [EC:2.3.1.48]
hsa:26749	GAGE8; G antigen 8
hsa:57054	DAZ3; deleted in azoospermia 3
hsa:57135	DAZ4; deleted in azoospermia 4
hsa:171489	SPANXE; SPANX family, member E
hsa:339041	LOC339041; hypothetical LOC339041
hsa:374948	LOC374948; similar to Hypothetical protein DJ845O24.1

5 Discussions

Robustness of Interval Sampling In terms of reconstruction, we have proposed the Interval Sampling method. After the exact sequence is processed by IS, ${}_{I-m}P_{I-m} \times_m P_m$ possibilities of the reconstruction occur. Here, I is the value of Interval, $m = l$ modulo I, and l is the length of the exact sequence. In general, it is possible to attach some random sequence, whose length is k, at the tail of the exact sequence in order to make $m = l + k$ modulo $I = 0$. Thus, the possibility of ${}_IP_I$ is realized. In this experiment, 15 is assigned as the value of I. ${}_{15}P_{15} = 1.3 \times 10^{12}$ possibilities of the reconstruction are realized for the TERT gene. In addition, as the second protection, the sequences are hashed after processed by IS. Hence, before taking into consideration the IS method, the before-hashed sequences must be reconstructed from only the hashed data, in order to reconstruct the exact gene sequence. That is to say, this dual protection prohibits the reconstruction.

Besides that, let us consider the reconstruction of sequences on a database. Now, suppose that all the gene sequences on the database have the same length, by attaching random sequences at their tail of the sequences. When there are C genes on the database, the possibility of ${}_{CI}P_{CI}$ is realized by means of IS. Because the origin of sampled sequences can not be distinguished from the anonymous. This value becomes an astronomical figure. Namely, the reconstruction of databases processed by IS is realistically impossible.

In order to improve the sensitivity of the verification, it might be valid to make the value of Interval I small, and to increase threshold θ. That is to say, our method becomes sensitive by making value I small and threshold θ large. However, the reconstruction from the data processed by IS becomes easier under the influence of the former. The reconstruction of the before-hashed data from the after-hashed ones becomes easier on account of the latter. There is a trade-off between security and sensitivity.

Application of unique sequences Owing to the Interval Sampling method, it becomes impossible to elaborate target specific primers for Polymerase Chain Reaction (PCR) [8]. However, using the results of this proposed method, it is possible to design probes for Ligase Chain Reaction (LCR) [9]. For example, the average number of unique sequences was about 13 and the value of Interval was 15, in this experiment. Therefore, it is possible to make a set of probes for LCR, whose product length becomes about 200-mer.

Table 5 signifies that there exists some gene sequences, for which it is impossible to design any short unique consecutive sequence as a PCR primer or DNA chip probe. In regard with probes, it is indispensable to make a short probe for detecting the expression of genes specifically, because a long probe is tolerant to a few mismatches. On the other hand, these genes have some unique sequences on condition of Interval value of 15. Therefore, the expression pattern of these genes can be specifically distinguished by means of LCR using specific short probes whose length is 15 each.

Relationship between calculation time, network speed and data size In this work, a *h.sapiens* gene set of 33 MB was used. The file size is much smaller than that of a whole genome. However, once the gene sequences are divided into some smaller pieces by the hash and divide function, these pieces can be simultaneously processed in parallel through the link and sort function. Let us consider a gene set whose size is N. The total size of the gene set processed by the hash function becomes θN. If the size of hash-key is assigned to k, the total file can be split into at most 4^k pieces. In this case, each file size becomes about $\frac{\theta N}{4^k}$. For instance, the size of the after-hashed file of the *h.sapiens* gene set became about 900 MB in this experiment. The length of 7 was selected as hash-key k. The threshold value θ was 30. Thus, it is possible to divide the after-hashed file into at most $4^7 = 16384$ pieces of less than 2 MB each. These pieces can be individually processed. Even if the file size of target gene sets becomes larger, it is feasible to process them in parallel by making the hash-key size larger.

In this experiment, 8 nodes at RCAST worked in parallel and spent 8 minutes processing the jobs. On the other hand, about 10 seconds were spent on transfering the query sequence. Hence, the computation on the 8 nodes was much heavier than the file transfer. Namely, the computing power is a bottleneck. Let us model the computation time, T_C. The sorting function is of $O(D \log D)$, where D is the size of data and can be divided into the number of nodes, n. When the hash-key length is k, the cal-

culation can be parallelized by 4^k, as mentioned above. Suppose that the size of query sequences is much smaller than that of databases and the performance of nodes is the same as one another. The computation time is described as $T_C = C\frac{D}{n}\log\frac{D}{n}$. When the parameters of this experiment, T_C=8 minutes, D=900 MB, n=8, are substituted, $C = 1.5\times10^{-2}$. The network bandwidth is effectively used if the time of file transfer, T_N, is more than T_C, namely, $T_N \geq T_C$. Let us consider how many nodes are needed to meet this condition. In this experiment, T_N=10 seconds. Hence, $\frac{10}{60} \geq 1.5\times10^{-2}\frac{900}{n}\log\frac{900}{n}$. For example, the above condition is satisfied with $n \geq 150$. In other words, if more than 150 nodes are available at RCAST, the calculation for the *h.sapiens* gene set of 33 MB can be finished within the file transfer time of a query sequence.

Our proposed method is well parallelized and can be easily extended to use the databases and computing resources distributed in the world. Let us consider an environment in which databases and computing resources are distributed. The network bandwidth among the nodes is given as V_N. Suppose that there are n nodes and each node processes the data of $\frac{D}{n}$. On this condition, each node launches $\frac{n-1}{n}\frac{D}{n}$ to others. If there exist lots of available nodes, namely $n \gg 1$, the data flow between nodes is $\frac{n-1}{n}\frac{D}{n} \approx \frac{D}{n}$, so that the file transfer time T_N, is $\frac{D}{n\times V_N}$. The network must be rapid enough to avoid idling the nodes. Namely, it is required that $T_C \geq T_N$. Hence, $V_N \geq \frac{1}{C\log\frac{D}{n}}$. For example, when the parameters in this experiment, D=900 MB, $C = 1.5\times10^{-2}$ are substituted for the equation, if $n = 512$ and $V_N \geq 2MB/s$, then the computation works well without any stall due to network delay.

An objective of the Data Grid project is to distribute an enormous amount of data into a great deal of databases in the world. In the near future, when it is realized to transfer the data of G-byte order without any difficulties, our proposed method works well on world-wide grid computing environments.

6 Conclusion

In this research, we proposed a novel sequence comparison method on grid computing environments. Without any key cryptosystem, this method avoids the reconstruction of the exact sequences by means of the Interval Sampling method and the hash function. Only the processed data are opened to the public for security reason. These opened data have information enough to calculate similarity of sequences to verify the novelty of them. As a by-product of the Interval Sampling method, some *h.sapiens* genes were discovered, which could not be specifically detected by hybridization but by LCR. The sequence comparison has been successfully executed in parallel by means of public grid computing resources.

7 Acknowledgments

This research is supported by CREST (Core Research for Evolutional Science and Technology) Program by Japan Science and Technology Agency ("MegaScale Computing by ultra low power technology and modeling").

References

[1] K. Kurata, V. Breton, and H. Nakamura, "A method to find unique sequences on distributed genomic databases," in *Proc. the 3rd IEEE/ACM International Symposium on Cluster Computing and the Grid*, Tokyo, Japan, May 2003, pp. 62–69.

[2] V. Breton, J. Montagnat, and R. Medina, "Datagrid, prototype of a biomedical grid," in *Proc. the conference 'Synergy between bioinformatics, medical informatics and neuroinformatics'*, Brussels, 2001.

[3] I. Foster, "The grid: A new infrastructure for 21st century science," *Physics Today*, vol. 54, no. 2, 2002.

[4] K. Kurata, H. Nakamura, and V. Breton, "Secret sequence comparison in distributed computing environments by interval sampling," in *Proc. the 2004 IEEE Symposium on Computational Intelligence in Bioinformatics and Computational Biology*, San Diego, USA, 2004, pp. 198–205.

[5] K. Kurata, H.Nakamura, and V. Breton, "Finding unique PCR products on distributed databases," *Transactions on Advanced Computing Systems, Information Processing Society of Japan*, vol. 44, no. SIG6, pp. 34–44, 2003.

[6] K. Kurata, H. Nakamura, and V. Breton, "A method to verify originality of sequences secretly on distributed computing environment," in *Proc. The 7th International Conference on High Performance Computing and Grid in Asia Pacific Region*, Saitama, Japan, 2004, pp. 310–319.

[7] Anonymous ftp of the genomenet. [Online]. Available: ftp://ftp.genome.ad.jp/pub/kegg/

[8] K. Kurata, G. Dine, C. Saguez, H. Nakamura, and V. Breton, "Evaluation of unique sequences on the european data grid," in *Proc. First Asia-Pacific Bioinformatics Conference*, Adelaide, Australia, 2003, pp. 43–52.

[9] F. Barany, "The ligase chain reaction in a pcr world," *PCR Methods Application*, vol. 1, no. 1, pp. 5–16, 1991.