# Dynamic Smith-Waterman Algorithm: A High-Performance Grid-Enabled Application Integrated with Globus, GridSphere Portal Framework and CoG Workflow for Performing Biological Local Sequence Alignment

Md. Khairul Bashar Chowdhury
Islamic University of Technology (IUT)
Gazipur, Bangladesh

Md. Rubaiyet Sadi
Islamic University of Technology (IUT)
Gazipur, Bangladesh

Tanzeem Bin Noor
Islamic University of Technology (IUT)
Gazipur, Bangladesh

Md. Mahbub-ul-Alam
Islamic University of Technology (IUT)
Gazipur, Bangladesh

## ABSTRACT
Smith-Waterman algorithm is one of the most significantly used algorithm and a well known approach to gain information about unknown genes and proteins for biological research. As execution time and accuracy is of great significance as handling large-scale dataset, a more reliable high-throughput and efficient parallelism can be achieved with the adaptation of grid environment. Adapting Smith-Waterman algorithm with the grid environment brings several concerns regarding fault tolerance, variability in resource performance and workload distribution, application availability etc. The work presented here aims at the development of a Dynamic Smith-Waterman algorithm metascheduler that handles all the specifications of job submission on the grid to the end user for local alignment search. Additionally a web based portal using GridSphere portal framework integrated with Globus 4 and Java Commodity Grid Kit is developed that reduces the complexity to the end users in accessing, managing and manipulating the grid resources and applications. The main contribution towards Dynamic Smith-Waterman algorithm is the reduction of the total job execution time up to 52% with accuracy up to 99.99% and better resource utilization by 40%. In addition, this work can be used as a template for the development of similar applications in future.

## General Terms
Grid, Biological local sequence alignment algorithm

## Keywords
Grid, Globus, GridSphere, Java Commodity Grid Kit, Scheduling, Smith-Waterman algorithm

## 1. INTRODUCTION
Grid computing [1] enables sharing, selection, and aggregation of geographically distributed resources including supercomputers, storage systems, data sources, and specialized devices owned by different organizations in order to solve large-scale resource-intensive problems in science, engineering and commerce. It follows a service-oriented architecture that provides hardware services, software services and infrastructure for secure and uniform access to heterogeneous resources and enables the formation and management of virtual organization (VOs) [2]. One of the main advantages of grid computing is its affordability, since each node can be purchased separately at a low cost, yet when combined can produce the power of an expensive multiprocessor supercomputer. Now, when a VO exists, the application needs to be deployed on the grids available resources. The deployment of grid application includes transfer of the entire application (i.e., source code, dataset, scripts) to a remote site, compiling the application on the remote host and making it available for execution. But before the deployment, the application should be classified and grid-enabled [3] according to the requirements of specific application. As for enabling the application for grid environment, the amount of effort required is greatly influenced by the application that is being grid-enabled. However, based on the parallelization methods and modes employed by the application may be more suitable than other on the grid.

Based on the application communication patterns and the parallelization model, applications on grid can be divided into in the following general categories [4]:

- Sequential applications
- Parametric Sweep applications
- Master-Worker applications
- All-Worker applications
- Loosely coupled parallel applications
- Tightly coupled parallel applications
- Workflow applications

Applications belonging to categories 1 through 4 are easier to be grid-enabled in decreasing order, while applications belonging to remaining categories may require rewriting of the complete application before it can be adopted to execute on the grid resources.

After gaining access to a grid environment through a VO and enabling an application for grid environment, users immediately expect an increase in performance of application jobs which is proportional to the number of resources available on the grid. This paper shows such a methodology that will use the available resources with a significant increase in performance. This can be contributed to the high degree of variability individual applications impose on underlying resources in terms of their requirements making generalized or standardized approaches fall short of realizing set goals. Because of such variability, general purpose metascheduler [5] do not focus on understanding application-specific requirements and thus cannot realize application-specific job scheduling and job performance. In order to realize such

behavior, application-specific schedulers are needed to focus on specific needs of an individual application.

With such multiple factors not only make it difficult to execute applications across grid resources but also executing an application efficiently. It can be concluded that a typical grid user will be hard-pressed to realize desired behavior for their jobs. In order to alleviate a user form these low-level infrastructure details, help of specialized and higher-end tools that mange grid infrastructure details is needed. On these extents this paper presents a design and implementation of the most commonly used algorithms for biological local sequence alignment where similar entries in the database is searched, namely Smith-Waterman algorithm [6].

The Smith-Waterman algorithm determines optimal local alignments between nucleotide or protein sequences and is therefore used in a wide range of areas from estimating evolutionary histories to predicting behaviors of newly found genes. However, the exponential growth in the nucleotide and protein databases has made the Smith-Waterman algorithm impractical to search on these databases because of its quadratic time and space complexity.

As a result, this led to innovations on the non-algorithmic front — specifically, special-purpose hardware solutions on FPGAs [7], [8] and linear processor arrays [9]. Simultaneously, there were also developments on the algorithmic front that gave rise to heuristics such as FASTA [10] and the BLAST [11] family of algorithms that sacrificed sensitivity for speed. These solutions present a multitude of trade-offs in terms of speed, cost and sensitivity of the sequence-search solutions. However, this presents a complete optimal sequence alignment (i.e., calculating the score and generating the alignment) using the affine gap penalty scoring scheme [12] that will deliver high speed, low cost, and ideal sensitivity via an ideal parallelization of the Smith-Waterman algorithm.

For the implementation of dynamic Smith-Waterman application there are two ways of segmentation procedure. In the database segmentation method, each cluster node only needs to search a query against its portion of the sequence database. Alternatively, query segmentation can be applied to alleviate the burden of searching jobs. In the query segmentation method, instead of the database, a subset of queries is distributed to each cluster node, which has access to the whole database. Fig. 1 points at the differences and the execution modes of the two Smith-Waterman parallelization models.

Because of the physical limitations encountered during parallelizing Smith-Waterman, traditional high performance computing technologies and techniques have been adopted as the solution to grid-enable Smith-Waterman algorithm. The Smith-Waterman algorithm has a broad community whose grid-enablement clearly brings significant benefit to the scientific community as it provides most accurate results. As similar efforts of grid-enabling, Smith-Waterman have not been undertaken yet our effort is described in this paper by providing details on developing Dynamic Smith-Waterman application – a grid-enabled Smith-Waterman algorithm. Hence our approach includes -

- Enabling Smith-Waterman application to execute across grid environments through a simple user interface (a web portal), thus hiding all complexities of underling infrastructure.
- Developing a Smith-Waterman application execution model for heterogeneous grid environments that analytically captures dependencies between the application and available resources.
- Finally incorporating a Smith-Waterman application-specific metascheduler into the execution model that implements the Smith-Waterman application execution model and is capable of understanding and leveraging heterogeneity of compute resources found across a grid to minimize job runtime.

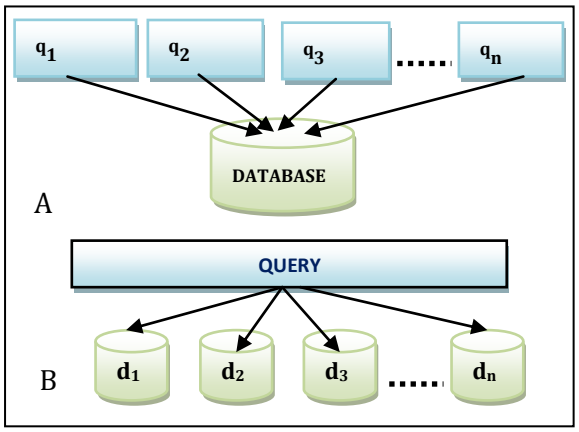



**Fig 1: Two models for parallelizing Smith-Waterman algorithm: (A) Query segmentation and (B) Database segmentation.**

The remainder of the paper is organized as follows: Section 2 provides an overview of the architecture, Smith-Waterman algorithm execution model and implementation details of Dynamic Smith-Waterman application. Results of executing Dynamic Smith-Waterman application on both query segmentation and database segmentation procedure and a comparative study over both parallelization models are provided in Section 3. Section 4 summarizes the paper and suggests some future work.

## 2. IMPLEMENTATION DETAILS

This section provides an overview of Dynamic Smith-Waterman application architecture and implementation details of the application.

### 2.1 Architecture Overview

The work discussed in this paper deals with a high-throughput application belonging to the minimal inter-node communication of applications. Although such applications are embarrassingly parallel and entail multiple executions of the same executable against different data, the challenges associated with creating a seamless Grid-enabled application are formidable. These include scheduling of the jobs on the various Grid nodes, staging of input files, executable databases and results between the initiating node and the remote nodes, setting up the executable and directory structures on the remote nodes, spawning of jobs on the remote nodes etc. Although an easier option would be to a priori transfer the relevant databases, input files and executable on the remote nodes, it would not be a truly independent Grid application.

Implementation of Dynamic Smith-Waterman application has been done with the consideration of these issues that ensures maximum flexibility and high end-user applicability. Furthermore, in developing an application for the Grid, it is

essential to have a unifying middleware that can provide a transparent interface to the underlying protocols. The Globus toolkit [13], developed as part of the Globus Project [14], is a software toolkit that addresses key technical problems in the development of Grid-enabled tools, services and applications. The Globus toolkit is a modular collection of technologies and enables the incremental development of Grid-enabled tools and applications. In addition, the toolkit implements standard Grid protocols and application program interfaces (APIs). The general approach in the development of the Globus toolkit is to define Grid protocols and APIs such that the protocols mediate the access to remote resources.

A web portal has been designed for the above work to provide the user an easy web based access to resources and to avoid complexity of installing grid softwares on end user machines. This has been implemented using GridSphere Portal Framework [15], a European project GridLab [16], developers of GPDK [17]. GridSphere provides grid-specific portlets and APIs for grid-enabled portal development that is compliant with JSR 168. Moreover, it allows for more high-level portlets using visual beans and GridSphere's user interface tag library. The portal uses a secure login and is then presented with a basic GUI based interface to the applications and other services.

Dynamic Smith-Waterman application has been developed using Java on top of the Globus 4.0.7 and has adopted the Java Commodity Grid (CoG) Kits [18] and Distributed Resource Management Application API (DRMAA) [19] standard for all job invocation operations. Furthermore, Dynamic Smith-Waterman application was built on top of GridWay [20], a grid job management system, for all job submission activities. A high level overview of Dynamic Smith-Waterman application's interaction with grid components is given in Fig. 2.

Fig. 2 shows that authentication and authorization have been moved outside the Dynamic Smith-Waterman application where the user is required to have valid X.509 GSI proxy credentials [21] before any job invocation. Resources available for job submission are discovered dynamically through GIS/MDS [22] which utilizes LDAP [23] to execute queries. SWAgent is a locally developed utility that monitors Smith-Waterman application related parameters across various resources such as application installation, setting input and output data locations, prepare for execution etc. Interaction with GridWay is performed through DRMAA API and is used for all job submission and monitoring activities. Dynamic Smith-Waterman application handles resource selection and data distribution; however, resource allocation, data transfer, and job monitoring are all delegated directly to the GridWay. Introduction of GridWay in Dynamic Smith-Waterman application development brings greater modularity and portability in application state. CoG provides added functionality to implement Java-based GSI, gridFTP, myProxy, and GRAM. As adoption of DRMAA standard within GridWay alleviates direct dependencies of Dynamic Smith-Waterman algorithm to GridWay, so further modularity of developed application increase.

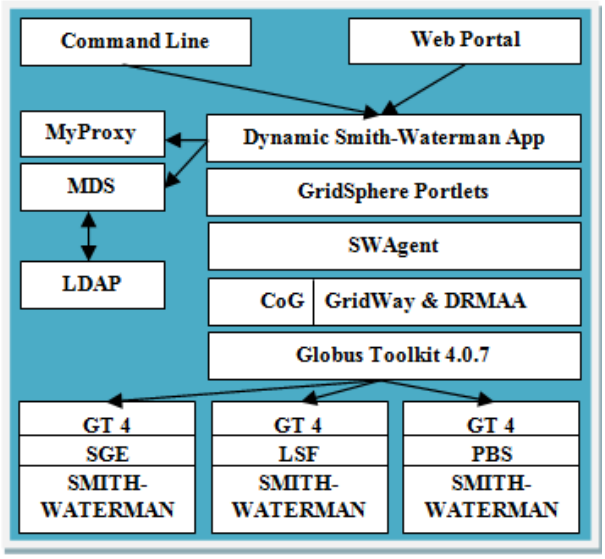



**Fig 2: High level diagram of interactions between Grid components and Dynamic Smith-Waterman application**

## 2.2 Smith-Waterman application architecture

Analysis of Smith-Waterman parallelization methods and grid resource characteristics has led Dynamic Smith-Waterman application to be internally developed under the master-worker communication model by embedding required components into a hierarchical framework. The master-worker model allows a single process to control the resource selection, data distribution, job submission and parameterization, as well as job monitoring. Thus, selected application model maximizes execution flexibility, code modularity and fault tolerance.

Going hand in hand with the master-worker model becomes the main choice regarding the parallelization model of Smith-Waterman jobs (i.e., query segmentation or database segmentation).Suitability of one method over another is dependent on both data and resource [24]. Thus based on the current resource availability, a method can often be found more appropriate than the other. So both query segmentation and database segmentation model is present in this application. The diagram of the components and dataflow is provided in Fig. 3 for both of the models. The main components of the given architecture for master node are:

- Analyze the input file
- Create schedule plan
- File (or database) parsing and fragmentation plan
- Thread creation
- Spawn remote jobs
- Gather result and cleanup procedure

Analyzing input file includes statistical analysis of user input query file to extract some parameters needed in later steps of input data processing. This information includes the number of queries, the average query length, the standard deviation etc. These information are then stored internally in a specific format for future work.

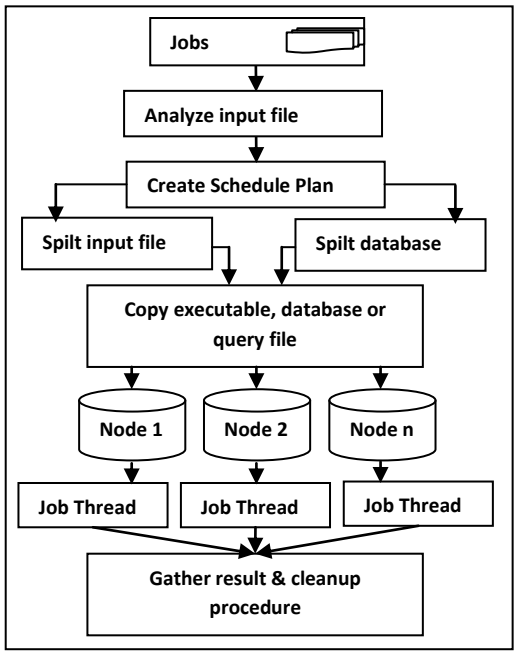



**Fig 3: Dataflow diagram for Dynamic Smith-Waterman application**

Create Schedule Plan module uses information retrieved by the Analyze input file module to perform on-line scheduling and job parameterization. This is the core module of Dynamic Smith-Waterman application and it implements a Smith-Waterman application performance model for resource selection and data distribution. The schedule plan is realized by implementing Eq. (1) for resource selection and Eq. (4) for data distribution.

$$S^j \subseteq R^j \mid \begin{cases} R_i^j \in S_i^j, R_i^j \geq T \\ R_i^j \notin S_i^j, R_i^j < T \end{cases} \quad (1)$$

From Eq. (1) $S^j$ represents the set of resources to be selected for execution of job $j$ and it is further described through Eq. (2) and (3). In respective equations, $R^j$ refers to the set of available resources capable of executing job $j$ and $T$ refers to the threshold value. Performance rate of individual resource from the perspective of Smith-Waterman job $j$ is calculated as a combination of the size of resource $i$, namely $n_i$ that represents the total number of processing elements (i.e., cores) and the performance rate of individual processing element on resource $i$ for the specific job $j$. Job performance rate is obtained from historical analysis of performance of Smith-Waterman application on a specific resource.

$$R^j = \bigcup \left\{ W_i^j \right\} \quad (2)$$

$$W_i^j = n_i * P_i^j \quad (3)$$

$$d_i = \frac{n_i}{\sum_{j=0}^{j=R} n_j} * D * W_i^j \quad (4)$$

For Eq. (4), $d_i$ represents the size of a task data chunk assigned to resource $i$; $n_i$ represents the number of processing elements on resource $i$; $D$ is the total size of user input (in the case of query segmentation process) or the total size of the target database (in the case of database segmentation process) and $W_i^j$ is the normalized weight or performance rate of the resource $i$. The result of executing the Schedule Plan module is a concrete plan for resource assignment and data distribution that is followed during the remainder of the job execution.

File (or database) parsing and fragmentation plan module reads the generated Job Plan and proceeds in two steps. Initially, it splits the original user query file into chunks (in the case of query segmentation process) or splits the target database into multiple databases (in the case of database segmentation process), one for each resource, whose size is available from the Job Plan as derived from Eq. (4). Each data chunk $d_i$ is then further subdivided into fragments $f_i$. This is done according to equation (5), i.e., proportionally to the number of processing elements $n_i$ within any one resource:

$$f_i = \frac{d_i}{n_i} \quad (5)$$

Thread Creation takes place when the master thread creates worker threads; one thread is created for each resource. Individual Threads read their respective part of the Job Plan to parameterize given task. Because of such granular approach to job plan generation and execution, as stated earlier, different Smith-Waterman application and parameters can be used for different resources. This allows customization and maximizes resource utilization as well as very high level of user support and Quality of Service (QoS). Jobs are submitted directly by threads to individual resources through DRMAA and GridWay while the master thread waits on the threads to complete. Individual threads initiate output file transfer back to the initial job submission resource before completing their execution. If a resource fails or a task does not complete its execution as planned, the master thread could resubmit just the given task to another resource.

Spawn remote jobs take place on each worker node where worker nodes connect to the GASS server on the master node. Once the server-to-server connections are made, the master node initiates the transfer of the necessary executable, database and query files and upon compilation sets up the executable as on the schedule plan describes earlier. As each node completes its quota of queries, the results are copied back to the master node.

Thread creation module is a part of the master thread and its primary task is to combine all the output and result files into a single result file presented to the end user. Any cleanup and additional tasks such as bookkeeping of performance results are performed in this step as well. Similarly, collecting results

and cleanup procedure module is included in the master thread combining all the output and result files into a single result file presented to the end user is its primary task. Any cleanup and additional task, such as measuring the performance metrics and save them into appropriate place to visualize are done in this module as well.

# 3. EXPERIMENTAL RESULTS

This section presents the experimental setup and performance results of Dynamic Smith-Waterman application across iutGRID (a local grid environment inside the university campus) resources. At first a web portal has been developed for testing the functionalities of the grid and then several applications were deployed into the grid. Performance comparison of the Smith-Waterman application is performed as an iterative process with both the basic query segmentation model and database segmentation model for parallelizing Smith-Waterman application and building the Smith-Waterman application specific metascheduler model described in the previous section to derive application-specific customizations that yield improved Smith-Waterman job performance.

## 3.1 Environment setup

The experiment of performance measurement of Dynamic Smith-Waterman application has been conducted on the resources available on iutGRID. A portal named TeraGrid Portal was developed for accessing the application. A snapshot of portal is given in Fig. 4. The resources of iutGRID are located across the department labs, student dormitories; each resource is locally administered with applicable policies and procedures in place. All of the resources on the grid had a version of GLOBUS installed and required input data available for use. Technical resource details are provided in Table 1.

**Table 1. Architectural details of resources on IUTGrid**

| Resource | Type A | Type B | Type C |
|---|---|---|---|
| ***Processor*** | Intel Core 2 Duo | Intel Core i3 | Intel Dual Core |
| ***Processor speed (MHz)*** | 2.80 | 2.93 | 3.2 |
| ***Main Memory(GB)*** | 2 | 2 | 2 |
| ***Cache (MB)*** | 4 | 4 | 2 |
| ***No. of nodes*** | 7 | 5 | 3 |
| ***No. of processor*** | 14 | 10 | 6 |

We used the popular nr (non-redundant) database to search against. The nr database is a non-redundant protein database with entries from GenPept, Swissprot, PIR, PDF, PDB, and RefSeq. The version used was 113 MB in size and available for download from the National Center for Biotechnology Information (NCBI) [25]. The input file used consisted of 128 search queries those were randomly selected from the Viral Bioinformatics Resource Center (VBRC) [26] database. The VBRC database contains the complete genomic sequences for all viral pathogens and related strains that are available for about half a dozen of virus families.

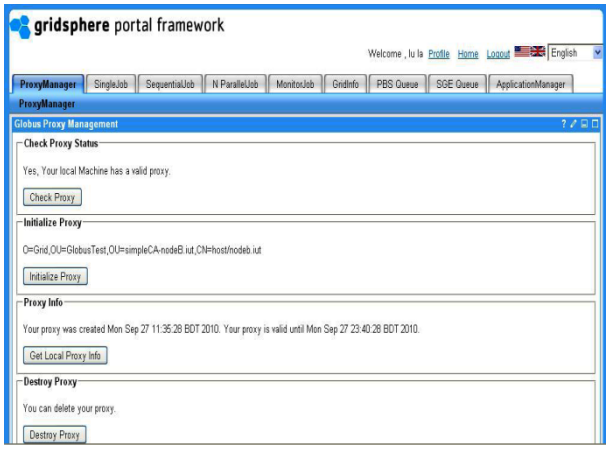



**Fig 4: Snapshot of TeraGrid Portal**

## 3.2 Performance definitions

In order to evaluate the results obtained from the execution of the application, it is necessary to define certain criteria. Specifically, while developing applications on the Grid, it is essential to define conditions for Grid-feasibility or infeasibility, that is, the criteria for measuring efficiency of a Grid-enabled application under the given conditions and resource set. Two essential measures that are needed to evaluate the performance of a Grid-application are Grid-speedup and Grid-efficiency. These are defined as follows.

- Definition 1: The Grid-speedup, $S_G$ is defined as the ratio of the total time taken by an application on a local node, $T_1$ of $p_1$ processors to that taken on $N_G$ nodes on the Grid, $T_g$ as Eq. (6)

$$S_G = \frac{T_1}{T_g} \tag{6}$$

- Definition 2: The Grid-efficiency, $E_G$ is defined as the ratio of Grid-speedup, $S_G$ to the total number of processors $P$ on $N_G$ nodes as Eq. (7)

$$E_G = \frac{S_G}{P} \tag{7}$$

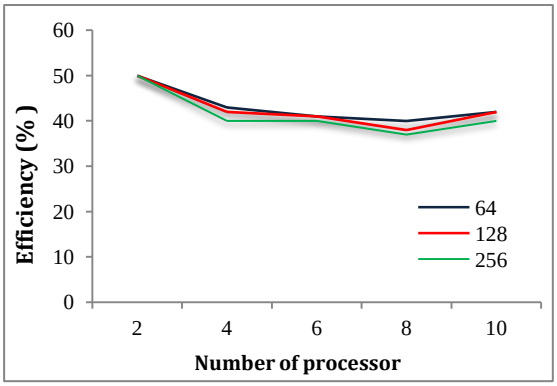
Grid-feasibility can then be defined as follows.

- Definition 3: A Grid-enabled application is said to be Grid-feasible if the Grid-speedup $S_G > 1$.

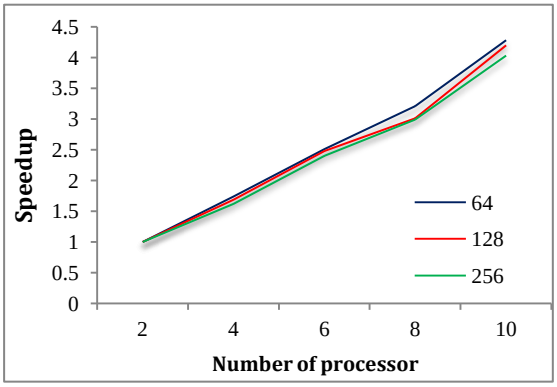
## 3.3 Performance Analysis

Performance of Dynamic Smith-Waterman application is measured with the definitions stated above for both basic query segmentation and database segmentation model. Dynamic Smith-Waterman application is designed to support single or multiple sequence searches against the target database. In the case of single sequence search the database segmentation model is initiated. For multiple sequences searching, both segmentation variant of Smith-Waterman job parallelization is necessary. Finally the performance of the query segmentation model against database segmentation model has been compared with the same sequence file.

For the first set of experiment three different size sequences of 64, 128 and 256 characters were used. The experiment is repeated for finding the grid speedup and efficiency by increasing the number of processors. Fig. 5(a) and 5(b) show the corresponding speedups and efficiencies for the three different searching sequence sizes respectively. From the graph it is clear that the application is grid-feasible as the speedup is greater than 1, hence we proceed to the other experiments.




(a)




(b)

**Fig 5: Results on single query search of (a) Grid speedup (B) Grid efficiency**

The second set of experiment shows how many the query segmentation variant operates under the model of dividing the total number of input queries across individual resources based on those resources' relative size. As such, resource weight factor from Eq. (4) is uniformly set to 1 and derived equation can thus be used to derive proportional amount of data that should be assigned to each resource. Given resource availability from Table 1, data distribution is obtained using Eq. (4), shown in Fig. 6.

After obtaining job parameters for the basic query distribution parallelization model, the Smith-Waterman approach of sequence alignment job has been executed across the available resources and the recorded runtime data for the basic query segmentation model are shown in Fig. 7. As master-worker parallelization model was used on the

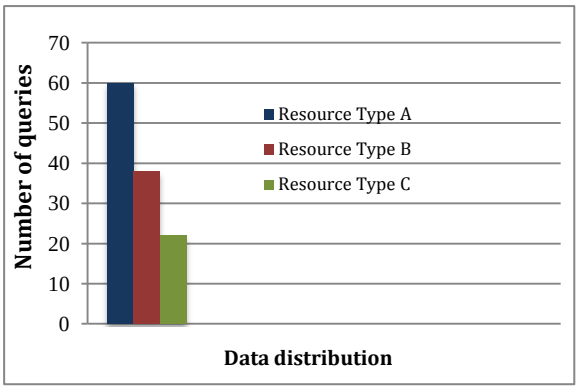



**Fig 6: Initial input data distribution based on the type of resources**

application, the overall job was considered to be complete after the longest job on the resource had finished. As a result of this scenario, load imbalance comes into front and a big discrimination is noticed in performing individual jobs on the resources. So, obviously the next step is to minimize the load imbalance across the individual resources by analyzing the issues behind the imbalance.

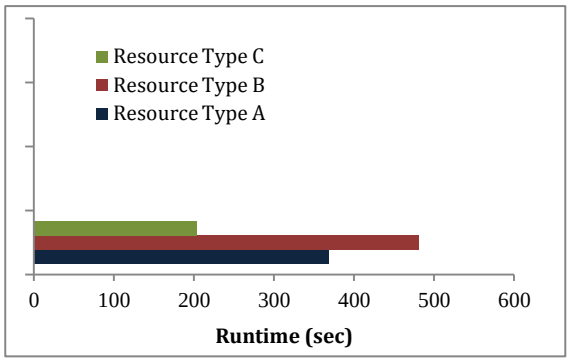



**Fig 7: Runtime results across the three resources for query segmentation model**

Analyzing the application in a context of performance, load imbalance can be explained by two properties. The first property is of Smith-Waterman algorithm that describes the runtime of the algorithm is significantly affected by the length of the given query input for search against the database. By simply taking the input file provided and dividing it into a number of chunks at predetermined data points, the type of data that gets assigned to individual nodes within a resource can vary greatly and so result in the load imbalance problem as lengths of individual queries vary greatly as well as spread unevenly across the provided input file. As these data chunks are divided among the available nodes on a particular resource, an inappropriate type of queries in length results in a load imbalance.

Dynamic Smith-Waterman addresses this issue by reorganizing the input data so that a proportional number of short, medium and long queries are assigned to each individual node. This problem can be generalized into a bin-packing problem where the number and size of bins is predetermined (i.e., number of chunks and number of queries assigned to each individual resource). For this purpose an effective and efficient heuristic implementation of the first fit decreasing algorithm (complexity is $\Theta(n \log n)$ where $n$ is the number of queries) is used, which assigns input data

elements across individual bins in a decreasing order for as long as there is input [27]. As the application was run onto a heterogeneous and distributed environment the File Parsing module was introduced for the dynamic Smith-Waterman application that provided an optimal solution to the load imbalance problem. The File Parsing module of Dynamic Smith-Waterman application implements the first fit decreasing algorithm as a two-step process: at first, the input file is divided into chunks of proportional type of data as prepared by the Job Plan, and secondly, each chunk is divided into a number of proportional fragments, as prepared in the Job Plan again. By applying the first-fit decreasing algorithm to reorganize assignment of individual queries to corresponding nodes, a better data distribution can be obtained which ensures a much more even distribution of comparative queries across individual compute nodes. As a result load balance among the nodes is achieved with better application runtime across the grid resources.

The second property affecting performance of Smith-Waterman jobs is the resource standard for the Smith-Waterman application. This is implemented as the resource weight factor in Eq. (4) and is realized by performing an application-specific standard on a given resource. The standard can be explicitly executed when a resource joins a largely static resource pool, alternatively a short running standard can be executed prior to the submission of the real job, or the information about historical runs of Smith-Waterman across resources can be kept in a local repository and then a resource-specific task can be performed before the job submission. The Dynamic Smith-Waterman relies on the availability of such application-specific information from Application Performance Database (AppDB) from the Application Information Services (AIS) [28] where runtime characteristics of previous application executions are stored and made available for extraction and analysis.

Final runtime results of Smith-Waterman-specific optimizations implemented as part of Dynamic Smith-Waterman are shown in Fig. 8. Obtained performance results are shown along with the runtime performance characteristics of the basic query segmentation model and the data redistribution model resulting from the application of the first-fit decreasing algorithm.

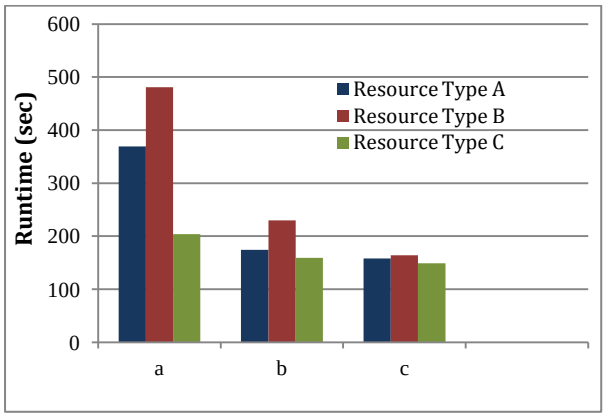



**Fig 8: Runtime characteristics of (a) basic query segmentation parallelization (b) first-fit decreasing data re-distribution, and (c) Dynamic Smith-Waterman (weighted first-fit decreasing).**

As such, results obtained in Fig. 8 shows a 52% reduction in runtime by using the Smith-Waterman-specific data redistribution model. Incorporating the Smith-Waterman-specific resource weight factor into the data distribution model and implementing the Dynamic Smith-Waterman application results in additional 16% reduction of the overall job runtime.

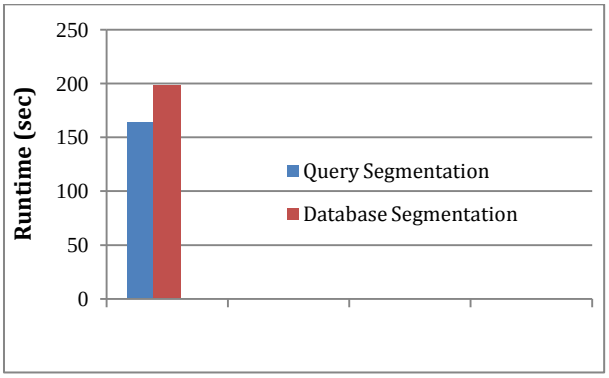



**Fig 9: Comparison between basic Query segmentation and Database segmentation**

On the third set of experiment the same input for query was used for searching. But in this case the database was segmented instead of the query input. The entire query search has been performed on all the local nodes of grid resources. After performing search on the specified resources the results are returned to the master-node and then it compares the result strings for the optimal search result. Fig. 9 shows the comparison of job runtime between the basic query segmentation and database segmentation. The results obtained in Fig. 9 shows using query segmentation reduces 17% runtime compared to database segmentation.

## 4. CONCLUSION AND FUTURE WORK

This paper presents the detail regarding development of a true grid enabled application with detailed architecture, implementation and performance results of Dynamic Smith-Waterman that can be accessed by a web portal. Dynamic Smith-Waterman is a powerful application used to perform Smith-Waterman searches on widely available grid resources. Described application achieves set goals by exploiting Smith-Waterman -specific characteristics to meet job requirements of resource capabilities, resulting in performance improvements exceeding 52%. At the same time, given approach enables efficient execution of Smith-Waterman jobs across general grid resources in a simple fashion resulting in significantly easier access to otherwise individual and heterogeneous grid resources that a user may have to deal with. From obtained experience and as part of future work, we plan on extending derived functionality into a general framework that would enable easier development of application-specific grid wrappers or applications where benefits observed in case of Dynamic Smith-Waterman can be easily realized.

## 5. ACKNOWLEDGMENTS

This work was made possible in part by helping with the labs and resources from the Department of Computer Science and Information Technology (CIT) at the Islamic University of Technology (IUT). The authors would like to thank the various system administrators who helped them in making the grid and running the application on the Grid.

## 6. REFERENCES

[1] Ian Foster and Carl Kesselman, "The Grid: Blueprint for a New Computing Infrastructure," Morgan Kaufman Publishers, San Francisco, California, 1998. *(references)*

[2] Ian Foster, Carl Kesselman, Steven Tuecke, "The anatomy of the Grid: Enabling scalable virtual organizations," International J. Supercomputing Applications, 15(3), 2001.

[3] Gabrielle Allen, Tom Goodale, Michael Russel, Edward Seidel,and John Shalf, "Classifying and enabling grid applications," Concurrency: Pract. Exper. 2000; 00:1-7.

[4] E. Afgan, P. Bangalore, and J. Gray, "A Domain-Specific Language for Describing Grid Applications," in Designing Software-Intensive Systems: Methods and Principles, P. F. Tiako, Ed., 2007.

[5] A. Kertesz and P. Kacsuk, " A Taxonomy of Grid Resource Brokers," in Distributed and Parallel Systems from Cluster to Grid Computing, 1 ed, P. Kacsuk, T. Fahringer, and Z. Nemeth, Eds.: Springer, 2007, pp. 201-210.

[6] T. F. Smith and M.S. Waterman, "Identification of common molrcular subsequences," J Molecular Biology, vol. 147 pp. 195-197, 1981.

[7] D. Lavenier, "Dedicated hardware for biological sequence comparison," Journal of Universal Computer Science, vol. 2, no. 2, pp. 77–86, 1996.

[8] Y. Yamaguchi, T. Maruyama, and A. Konagaya, "High Speed Homology Search with FPGAs," Pacific Symposium on Biocomputing 7:271-282 (2002).

[9] R. Hughey, "Parallel hardware for sequence comparison and alignment," Comput Appl Biosci. 1996 Dec;12(6):473-9.

[10] D. J. Lipman and W. R. Pearson, "Rapid and sensitive protein similarity searches.," Science, vol. 227, pp. 1435–1441, March 1985.

[11] S. F. Altschul, W. Gish, W. Miller, E. W. Myers, and D. J. Lipman, "Basic local alignment search tool.," J MolBiol, vol. 215, pp. 403– 410, October 1990.

[12] A. M. Aji, W. Feng, F. Blagojevic, and D. S. Nikolopoulos, "Cell-SWat: Modeling and Scheduling Wavefront Computations on the Cell Broadband Engine," in Proc. of the ACM International Conference on Computing Frontiers, May 2008.

[13] I. Foster and C. Kesselman, "The Globus toolkit," in The Grid: Blueprint for a New Computing Infrastructure, I. Foster and C. Kesselman, Eds., San Francisco, California: Morgan Kaufmann, 1999, pp. 259--278.

[14] The Globus Project. http://www.globus.org.

[15] Jason Novotny, Michael Russel, and Oliver Werens. GridSphere: An Advanced Portal Framework. In EUROMICRO '04: Proceedings of the 30thEUROMICRO Conference, pages 412-419, Washington, DC, USA, 2004. IEEE Computer Society.

[16] The GridLab Project. http://www.gridlab.org

[17] Jason Novotny.The Grid Portal Development Kit. Concurrency and Computations: Practice and Experience, 14(13-15):1129-1144,2002.

[18] Gregor von Laszewski, Ian Foster, JarekGawor, and Peter Lane.A Java Commodity Grid Kit. Concurrency and Computations: Practice and Experience, 13(89):643-662, 2001.

[19] H. Rajic, R. Brobst, W. Chan, F. Ferstl, J. Gardiner, A. Haas, B. Nitzberg, and J. Tollefsrud, "Distributed Resource Management Application API (DRMAA) Specification 1.0 GFD-R-P.022," Global Grid Forum (GGF) 2004.

[20] E. Huedo, R. S. Montero, and I. M. Llorente, "A Framework for Adaptive Execution on Grids," Journal of Software - Practice and Experience, 34(2004, pp. 631-651.

[21] I. Foster, C. Kesselman, G. Tsudik, and S. Tuecke, "A Security Architecture for Computational Grids," in 5th ACM Conference on Computer and Communication Security Conference, San Francisco, CA, 1998, pp. 83-92.

[22] K. Czajkowski, S. Fitzgerald, I. Foster, and C. Kesselman, "Grid Information Services for Distributed Resource Sharing," in 10 th IEEE Symp. On High Performance Distributed Computing (HPDC), Los Alamitos, CA, 2001, pp. 181-195.

[23] G. von Laszewski and I. Foster, Usage of LDAP in Globus.1999.

[24] C. Wang and E. J. Lefkowitz, "SS-Wrapper: a package of wrapper applications for similarity searches on Linux clusters," BMC Bioinformatics, 5(171), 2004.

[25] The NCBI database. http://www.ncbi.nlm.nih.gov/

[26] The VBRC home. http://www.biovirus.org/

[27] J. Y.-T. Leung, Ed. Handbook of Scheduling: Algorithms, Models, and Performance Analysis, 1st ed., vol. 1: CRC Press, 2004.

[28] E. Afgan and P. Bangalore, "Assisting Efficient Job Planning and Scheduling in the Grid," in Handbook of Research on Grid Technologies and Utility Computing: Concepts for Managing Large-Scale Applications, E. Udoh and F. Z. Wang, Eds.: IGI Global, 2009.