Enabling Large Scale Scientific Computations for Expressed Sequence Tag Sequencing over Grid and Cloud Computing Clusters

Sangmi Lee Pallickara[1], Marlon Pierce[2], Qunfeng Dong[2], and ChinHua Kong[1, 3]

[1]Community Grid Lab, Indiana University,
501 N. Morton Street, Bloomington, IN 47404
[2]The Center for Genomics and Bioinformatics, Indiana University,
Jordan Hall 044, Indiana University, 1001 East Third Street, Bloomington, Indiana 47405
[3]Computer Science Department, Indiana University,
150 S. Woodlawn Ave. Bloomington, IN 47405-7104
leesangm@cs.indiana.edu, mpierce@cs.indiana.edu,
qunfengd@gmail.com, kongch@umail.iu.edu

Abstract. Compute-intensive biological applications are heavily reliant on the availability of computing resources. Grid based HPC clusters and emerging Cloud computing clusters provide a large scale computing environment for scientific users. However, large scale biological application often involves various types of computational tasks which can benefit from different types of computing clusters. Therefore, a high level job scheduling environment which integrates the Grid style HPC clusters and the Cloud computing clusters and manages jobs accordingly based on the characteristics of the jobs is required. In this paper, we propose a Web service framework for high-level job scheduling – Swarm. Swarm is developed for scientific applications that must submit massive number of high-throughput jobs or workflows to highly distributed computing clusters. Swarm allows the users to submit jobs to both Grid HPC and Cloud computing clusters. The Swarm service itself is designed to be extensible, lightweight, and easily installable on a desktop or a small server. As a Web service, derivative services based on Swarm can be straightforwardly integrated with Web portals and science gateways. This paper provides the motivation for this research, the architecture of the Swarm framework, and a performance evaluation of the system prototype.

Keywords: Grid computing, cloud computing, bioinformatics, scientific computing, high throughput computing

*This research was supported by the TeraGrid project and utilized the BigRed(Indiana University), Abe, Cobalt(NCSA), NSTG cluster(ORNL) , Steele(Purdue), Ranger(TACC), Visualization cluster (UC/ANL).

1 Introduction

Recent activity in biologically based research has required substantial amount of computing resources, and is currently impeded by limited local computing infrastructure. National cyberinfrastructure (such as the TeraGrid [1] and Open Science Grid [2]) can potentially be harnessed if it can be made easier to use. Furthermore, the fundamental nature of this infrastructure is likely to change over the next five years as Cloud Computing [3] approaches are evaluated and adopted.

Cloud and grid computing approaches are complementary and need to be integrated: clouds are well suited for on-demand jobs that individually last a few seconds or minutes, but they are not a good fit for closely coupled parallel applications, where the TeraGrid is better suited. Since many of our applications require both types of resources, an infrastructure which can cope with the distinctive characteristics of resources are required.

To benefit from powerful computing resources, users login to remote machines or submit jobs through Grid toolkits such as the Globus Toolkit [4]. Similarly, Cloud computing clusters provide a Web service interface to manage the machine instances and is actively adapting map-reduce like programming paradigm. However, the sheer number of jobs easily exceeds one's ability to manage them manually. Finding the most efficient resource to submit, and monitoring such submitted jobs, is not feasible without intelligent support from the middleware. Furthermore, Gateway-style applications also require interfaces to distributed resources on behalf of the users.

Therefore, we have developed Swarm[5], a robust job management system that allows users to easily submit and monitor millions of serial or parallel jobs to multiple computers in Grid and Cloud computing clusters. Swarm currently supports job submission to the TeraGrid and Amazon's EC2 cloud [3]. Swarm provides a standard Web Service interface that can be easily integrated into Web portal style applications or Workflow engines. In addition, Swarm provides a highly extensible software design so that scientists or developers can easily customize it based on their application specific requirements.

Swarm capabilities include, (1) scheduling large number of jobs over distributed HPC clusters including Grid clusters and Cloud clusters (2) monitoring framework for large scale jobs (3) standard Web Service interface for web applications (4) extensible design for the domain specific software logics (5) use of Condor-G and Birdbath for the Grid clusters and Hadoop Map-Reduce engine for the Cloud clusters.

The aims of this paper are threefold: first, to present the architecture of the Swarm job scheduling framework; second, to describe the scheme that manages job submissions to Grid and Cloud computing clusters; and finally, to evaluate performance at the job submission and process level.

The rest of the paper is organized as follows: Section 2 describes bioinformatics research projects that motivated our research. In the section 3, we describe the architecture of Swarm. Related work is discussed in the section 4. Performance evaluation of the Swarm system is presented in section 5. Conclusions and future work are discussed in section 6.

2 Computational Challenges in the EST Sequencing

An EST (Expressed Sequence Tag) corresponds to messenger RNAs (mRNAs) transcribed from the genes residing on chromosomes. Each individual EST sequence represents a fragment of mRNA, and the EST assembly aims to re-construct full-length mRNA sequences for each expressed gene. Because ESTs correspond to the gene regions of a genome, EST sequencing has become a standard practice for gene discovery, especially for the genomes of many organisms that may be too complex for whole-genome sequencing. For example, many agriculturally important plants (e.g., wheat) have huge genomes that are full of repetitive elements. The repetitive elements pose an unsolved challenge for correct assembly. For those organisms, EST sequencing remains as the only efficient way to discover genes in large scale. Even species for which whole-genome sequences are available, EST contigs are important data for accurate gene annotation. In particular, the EST sequencing and assembly of daphnia, sea urchin, tick, lizard, wasp, and a number of other growing organisms are major target tasks. Many computational steps for a typical EST assembly task (e.g., repeat masking or running CAP3 or other assembler on clustered data set) can be easily sped up in the data parallelization fashion if sufficient computing resource can be accessed.

Challenge 1: Executing tens of thousands of jobs.

Around the world, the number of EST sequences is also growing at an ever-increasing rate. For example, more than 100 plant species have at least 10,000 EST sequences. Therefore, the assembly process (e.g. a CAP3 application) creates tens of thousands of computing jobs to be processed. Current Grid based clusters do not allow users to submit 1000s of jobs concurrently to batch queue systems.

Challenge 2: Requirement of job processing is various.

Even in the EST sequencing process for a single species, the EST assembly process composes various types of computation jobs to be processed. It includes large scale parallel processing, serial processing and embarrassingly parallel jobs. Some of the jobs require more than 100 computing nodes to process the parallel jobs, and there are small jobs which need a single processor and runs only few seconds. This variety often leads the biologist to search the various types of clusters which fit the computing requirements.

3 Architecture

Swarm provides a standard Web service interface to desktop users and Web applications to submit and manage the jobs. Each of the operations and parameters are defined in the WSDL associated with the services. As seen in Fig. 1, the requests from the users are delivered to the Large Task Load Optimizer of the Swarm-Analysis. Each of the jobs is distributed to the Swarm-Grid, Swarm-Dryad, or Swarm-Hadoop based on the characteristics of the job. Each of the components, Swarm-Grid, Swarm-Dryad and Swarm-Hadoop are also Web services. Therefore, users are

allowed to access each of the resource-specific Web services directly. Similarly, applications can host any of sub Web services for their own purpose.

Requests from the users are delivered to the Request Manager. The Request Manager creates a 128-bit universally unique identifier ticket for the series of jobs. To provide the capability to track a large number of jobs, Swarm provides a simple structure to the submitted jobs. Jobs are identified by their ticket and internal ID. Here, internal ID is the identity of the job, which is unique within the job group. This structure is especially useful for the scientific web application, which deals with multiple experiments launched by multiple users.

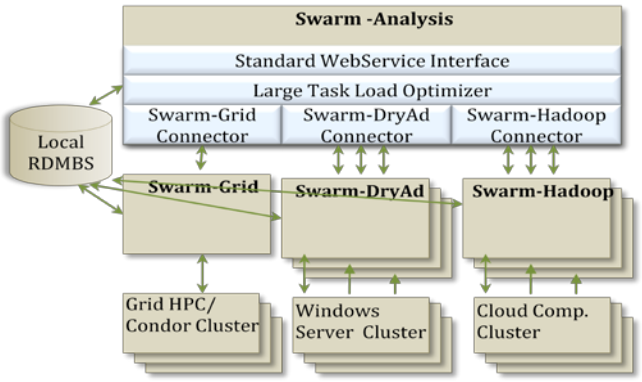

Fig. 1 Architecture of the Swarm infrastructure

For the high-throughput jobs, Swarm considers traditional Grid HPC clusters as the suitable computing resource. Parallel jobs (e.g. MPI jobs) or jobs requiring longer execution time are considered to be distributed to the Swarm-Grid. The Job submission process interacts with the Resource Ranking Manager, which prioritizes the resources over which the job is submitted to optimize the job execution process. This is especially designed for the Grid HPC clusters network such as TeraGrid. Users are allowed to specify multiple resources (computing clusters) to submit the job. To prioritize the resources listed in the user's job description, Swarm interacts with the QBETS [6] batch queue prediction service. The Data Model Manager determines the data model for the input, output and temporary files during the process.

The Fault Manager decides how to respond to the faults encountered during the job submission and execution. Swarm categorizes faults into two catagories: fatal fault and recoverable fault. A fatal fault is defined as faults that cannot be recovered without new inputs from the users or relocating the jobs on different computing clusters. Recoverable fault refers to faults that can possibly be recovered without contacting the user. It is mostly related to resource specifications such as expected execution time or memory requirements. When Swarm suspects that the fault is due to insufficient resource specifications, the jobs are resubmitted with modified arguments.

Under the Request Manager and Resource Ranking Manager, there is a group of software components; referred to as Job Board. Swarm maintains a Job Board for each user. Each Job Board contains a Job Queue, Job Distributor, and Resource

Connector. Users do not share any of these components. Matchmaking between the jobs and the resources are done in the user's Job Board.

When the Job Distributor finds a match with an available remote resource, the Job Execution Manager will submit the job through CodorG [11] Resource Connector to the Grid HPC clusters and Condor Resource Connector to the traditional Condor Cluster.

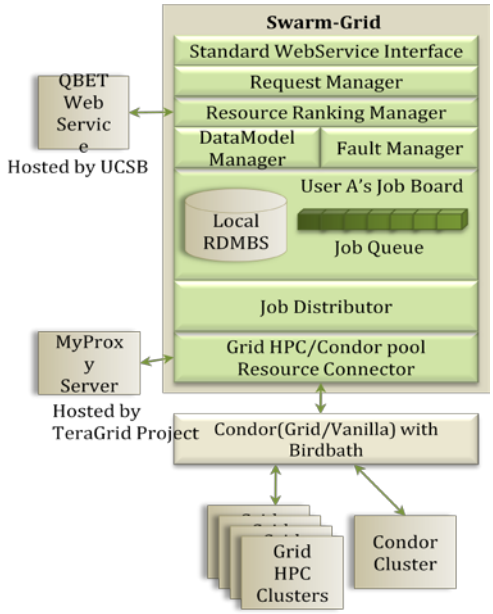

Fig. 1 Swarm-Grid architecture

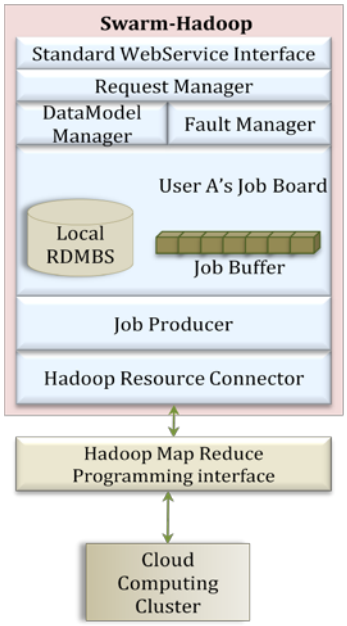

Fig. 3 Architecture of the Swarm-Hadoop

Swarm-Hadoop is a software component which enables users to submit jobs to the cloud computing clusters such as Amazon's EC2. Swarm-Cloud interacts with Swarm-Analysis via a Web service interface. Therefore, users can access Swarm-Cloud directly, if their jobs are suitable for the Cloud computing cluster.

Swarm-Hadoop requires a Cloud computing cluster with the Hadoop map-reduce engine. For convenient management of the Cloud computing cluster combined with the Swarm infrastructure, Swarm requires Hadoop's master nodes to reside on the machine running the Web Service container of the Swarm-Hadoop Web service. Therefore, the machine instances of EC2 or any Hadoop worker node can be integrated into the Cloud computing cluster.

Each of the jobs is processed as a single Map function in the map-reduce style application. Since many of scientific applications are not developed in the map-reduce style, Swarm-Cloud provides an interface for legacy applications. To process non-map-reduce software, Swarm-Cloud provides additional processes including:

- Transferring input files from the user's local file system to the location that Hadoop's worker nodes can access (e.g. Hadoop distributed file system, Web accessible location, or storage service such as Amazon S3)
- Transferring input and executables files from the Hadoop distributed file system, storage service (e.g. Amazon S3), or Web accessible location to the local file system of the computing nodes.
- Transferring output and standard output files from the local file system of the computing nodes to the Hadoop distributed file system or storage service.

Swarm-Hadoop also provides DataModel Manager which determines the location of the input, output and temporary files.

4 Performance Evaluation

We have implemented a prototype of the Swarm framework, in Java, based on Apache Axis2[7]. The server was hosted on a machine with 3.40GHz Intel Pentium 4 CPUs and 1GB RAM. The client software was hosted on a machine with a 2.33 GHz Intel Xeon CPU and 8GB RAM. The machines involved in the benchmark were hosted on 1 Gbps network.

We installed Swarm-Grid and Swarm-Cloud on a machine with 3.40 GHz Intel Pentium 4 CPUs with 1GB RAM. For Swarm-Grid, this testbed provided 7 of the TeraGrid roaming clusters including as described in Table 1. However, during the period of this performance evaluation, we could access only 2-3 of clusters at a time due to system downtime or service failure. Table 1 shows the capacity of the clusters.

For Swarm-Cloud, we used the EC2 cluster from the Amazon Web Service with the m1.small instance. Each of the instances is running on a machine with 2.5 GHz Dual-Core AMD Opteron Processor with 1.7 GB RAM.

We used partial set of the human EST fragments published by NCBI GenBank [8]. Data set is categorized based on the execution time on a single CPU machine with 3.40GHz Intel Pentium 4 CPUs and 1GB RAM: very small job(less than 1minute), small job (1 ~ 3 minutes), medium jobs (3 ~ 10 minutes), large job(longer than 10 minutes).

Table 1. Computing Capacity of TeraGrid HPC clusters

TeraGrid Cluster	CPU (each core, GHz)	Distributed Memory (TBytes)	Total number of nodes	Total number of processors	Disk space(TBytes)
BigRed	2.5	6.00	768	3072	266
Ranger	2.3	125.00	3936	62976	1730
Abe	2.3	9.38	1200	9600	100
Cobalt	1.6	3.00	1024	1024	100
ANL IA-32	2.4	0.24	96	128	4
NSTG	3.0	0.07	28	56	2
Steele	2.3	12.40	893	6496	170

However, the execution time may be different for different machines and clusters. For the TeraGrid, clusters provide an environment for the high throughput computing such as larger distributed memory and storage. In this paper, we considered only serial jobs. The test application we executed on each of the target cluster was CAP3 [9] which assembles the gene sequence from the file containing a set of fragments that are already clustered. We assume that the input data set resides at the clusters. For instance, for the Grid resources, we launched jobs after the data is stored in each of the clusters. For Swarm-Hadoop, jobs are launched with the input data stored in the local file system of the server which is also the master node of the Hadoop cluster.

We have noticed there are significant amount of overhead for the jobs submitted to the TeraGrid HPC clusters: it includes:

- Overhead from the job submission middleware: in our case, Condor, Globus Toolkit job submission service. This overhead is typically 30 seconds to few minutes.
- Overhead within the clusters: between many of computing nodes and batch queue systems, there was overhead especially for the synchronization of the status and file systems. This overhead is various from site to site.
- Wait time in the batch queue: Depending on the local policy and the status of the job queue, it varies from few seconds to hours.
- Overhead during the file stage in and out: after the job is completed, file transfer adds another set of overheads in the middleware and within the cluster.

Therefore, we have experienced job execution times in the order of a few minutes for the very small jobs which take less than 1 minute. This overhead could be considered as trivial amount for high-throughput computing jobs. However, for a large number of very small jobs, the accumulated overhead could be a serious bottleneck in executing jobs over Grid style resources.

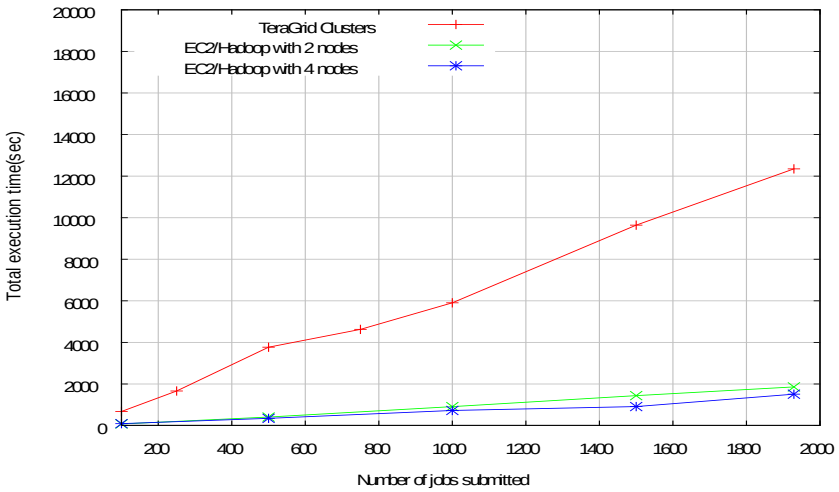

Fig. 4 Total Execution Time for the Various Numbers of Jobs (~1minute) with Swarm-Grid and Swarm-Hadoop

As seen in the Fig. 4, Swarm-Hadoop processes smaller size of jobs much more efficiently than the Swarm-Grid. Cloud Computing clusters such as Amazon EC2 or Eucalyptus[10] provide immediate access to the resources with smaller overhead for each of jobs without waiting time in the job queue, and overhead due to the additional software stacks using polling mechanism to synchronize status.

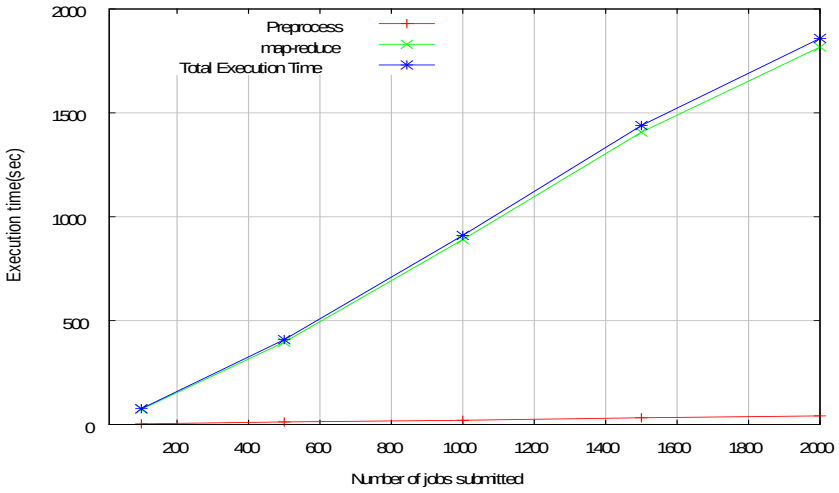

Fig. 5 Job Execution time in Swarm-Hadoop using Hadoop distributed file system

Although compared to the Grid HPC clusters, Cloud computing clusters provides a more suitable job execution environment for small size serial jobs, we have observed that there are non trivial amounts of data movement to cope with map-reduce style programming paradigm. Since most of current scientific application is not developed as map-reduce style, we developed a software component which distributes serial jobs to the map function and provides a local environment for each of the computing nodes. Fig. 5 depicts the execution time for the preprocess and map-reduce for various number of jobs. Preprocess time includes the time for copying the input files from the local file system of master node to the Hadoop distributed file system. Map-reduce time includes the time for copying input, output, executable files between Hadoop distributed file system and the local file system of the computing node, and process jobs on the computing node.

For medium sized jobs (3~10minutes), Swarm-Grid processed 100 jobs in 217 seconds. Similarly, Swarm-Grid processed 17 large size jobs (>10 minutes) in 3718 seconds. For the medium or large size jobs, Swarm-Grid provides reasonable execution time. In addition, we have observed that the numbers of medium and large size jobs are significantly smaller than the number of small and very small size jobs in the EST sequencing (actually, they appear to be distributed according to a simple power law). Therefore, the overhead in the medium and large size jobs is not expected to accumulate and impact the overall performance.

5 Related Works

There have been several approaches in the HPC community, especially in the Grid community to manage jobs remotely. Condor [11] is a well-known high-throughput resource management system that has been widely adopted in the scientific computing community. Swarm-Grid utilizes CondorG as the basic job submitter. GridWay [12] is another metascheduling framework for grid resources. Besides the scheduling capability, GridWay provides other advanced features such as fault tolerance, checkpoints, and process migration− that are not available to users who access Grid resources directly. Similarly, PanDa[13] provides a large-scale job scheduling and analysis framework. Condor-G, GridWay, and PanDa harness the Globus toolkit to cope with security and policy issues in Grid settings.

myCluster [14] and Falkon[15] are built on top of the glide-in approach. myCluster provides the capability of provisioning a large number of distributed resources across the TeraGrid into personal clusters created on-demand. Falkon provides support for provisioning a large number of distributed resources and allows user groups to access resources via a Web service interface. It also factors in data management techniques to improve the performance. Glide-in style approach provides transparent access to remote resources.

Cloud computing is relatively new to the scientific community. Many activities to integrate Clouds and Grids are underway, but these are in early stages. However, there has been active research to adapt Cloud computing technology to scientific problems. There are several Cloud computing infrastructures available including Amazon EC2 [3], Eucalyptus [6], and Nimbus [16]. These infrastructures provide scientific application such as Matlab, and mpi.

6 Conclusions and Future Work

In this paper, we have introduced a high-level job scheduling framework, Swarm. The Swarm framework integrates Grid style HPC computing resources and Cloud computing clusters to solve a specific scientific computing problem. The Swarm service is designed to be extensible and lightweight so that users working on desktops or small servers can easily install and host it. Since it is Web service based, clients to derivative services based on Swarm can be integrated in Web portals and science gateways. In this paper, we have discussed the motivation for this research, the architecture of the Swarm framework, and a performance evaluation of the system prototype.

As we discussed in the section 4, Swarm-Hadoop can process small-sized jobs (shorter than 1 minute) much more efficiently than Swarm-Grid. Cloud Computing clusters such as Amazon EC2 or Eucalyptus provides immediate access to the resources with smaller overhead for each of jobs. Meanwhile, Grid style HPC clusters add a few minutes (or an even longer overhead) to each of jobs. Therefore, if the user's application requires processing large number of jobs, it can benefit from access to the Cloud Computing clusters. However, Grid style HPC clusters still provide stable environment for large-scale parallel jobs.

As part of our future work, we plan to enhance our service to access to Windows based clusters and use Microsoft's such as Dryad [17] for job management. We also plan to investigate the fault tolerance scheme and enhanced job-monitoring mechanisms for Cloud Computing environments. Finally, we plan to investigate an intelligent job distribution mechanism among various types of computing resources. We expect that intelligent job distribution will offer significant performance improvements for large-scale biological applications.

References

1. Catlett, C. et al. "TeraGrid: Analysis of Organization, System Architecture, and Middleware Enabling New Types of Applications," HPC and Grids in Action, Ed. Lucio Grandinetti, IOS Press 'Advances in Parallel Computing' series, Amsterdam, 2007.
2. Ian T. Foster et al. : The Grid2003 Production Grid: Principles and Practice. HPDC 2004: 236-245.
3. Youseff, Lamia; Butrico, Maria; Da Silva, Dilma: Toward a Unified Ontology of Cloud Computing, Page(s): 1-10 In Proceedings of Grid Computing Environments, 2008. Digital Object Identifier 10.1109/GCE.2008.47384432.
4. Ian T. Foster: Globus Toolkit Version 4: Software for Service-Oriented Systems. NPC 2005: 2-13.
5. Sangmi Lee Pallickara and Marlon Pierce SWARM: Scheduling Large-scale Jobs over the Loosely-Coupled HPC Clusters Proceedings of the IEEE International Conference on e-Science. Indianapolis. 2008. December 7-12 2008.
6. Daniel Nurmi, John Brevik, Richard Wolski, "QBETS: queue bounds estimation from time series," *SIGMETRICS*, 2007: 379-380
7. Srinath Perera, et al., "Axis2, Middleware for Next Generation Web Services" in the ICWS 2006: 833-840, 2006
8. National Center for Biotechnology Information, http://www.ncbi.nlm.nih.gov/
9. Huang, X. and Madan, A. (1999) CAP3: A DNA sequence assembly program. *Genome Res.*, **9**, 868-877.
10. Daniel Nurmi, et al., "The Eucalyptus Open-source Cloud-computing System, in Proceedings of Cloud Computing and Its Applications" [online], Chicago, Illinois (October 2008)
11. Todd Tannenbaum, Derek Wright, Karen Miller, and Miron Livny, *Condor - A Distributed Job Scheduler. Ed. Thomas Sterling, Beowulf Cluster Computing with Linux*, The MIT Press, 2002, ISBN: 0-262-69274-0
12. Eduardo Huedo, Rubén S. Montero, Ignacio Martín Llorente, "A framework for adaptive execution in grids," *Soft., Pract. Exper.* 34(7): 631-651, 2004
13. Globus Toolkit, Overview and Status of Current GT Performance Studies. [online][accessed 2008 August 9]. Available: http://www.globus.org/toolkit/docs/4.0/perf_overview.html
14. Edward Walker, J. P. Gardner, V. Litvin, and E. P. Turner, "Personal Adaptive Clusters as Containers for Scientific Jobs," *Cluster Computing*, vol. 10(3), September, 2007
15. Ioan Raicu, Yong Zhao, Catalin Dumitrescu, Ian Foster, Mike Wilde,"Falkon: a Fast and Light-weight tasK executiON framework," in the IEEE/ACM SuperComputing, 2007
16. Keahey, K., T. Freeman, "*Contextualization: Providing One-Click Virtual Clusters,"* eScience 2008, Indianapolis, IN. December 2008
17. Michael Isard, Mihai Budiu, Yuan Yu, Andrew Birrell, Dennis Fetterly: Dryad: distributed data-parallel programs from sequential building blocks. EuroSys 2007: 59-72