Challenges and Approaches for Distributed Workflow-Driven Analysis of Large-Scale Biological Data

[Vision Paper]

Ilkay Altintas,* Jianwu Wang, and Daniel Crawl
San Diego Supercomputer Center
University of California, San Diego
9500 Gilman Drive
La Jolla, CA 92093-0505 USA
{altintas,jianwu,crawl}@sdsc.edu

Weizhong Li
Center for Research in Biological Systems
University of California, San Diego
9500 Gilman Drive
La Jolla, CA 92093-0446 USA
weli@ucsd.edu

ABSTRACT

Next-generation DNA sequencing machines are generating a very large amount of sequence data with applications in many scientific challenges and placing unprecedented demands on traditional single-processor bioinformatics algorithms. Middleware and technologies for scientific workflows and data-intensive computing promise new capabilities to enable rapid analysis of next-generation sequence data. Based on this motivation and our previous experiences in bioinformatics and distributed scientific workflows, we are creating a Kepler Scientific Workflow System module, called "bioKepler", that facilitates the development of Kepler workflows for integrated execution of bioinformatics applications in distributed environments. This vision paper discusses the challenges related to next-generation sequencing data, explains the approaches taken in bioKepler to help with analysis of such data, and presents preliminary results demonstrating these approaches.

Categories and Subject Descriptors

H.4 [**Information Systems Applications**]: Miscellaneous

General Terms

Application

Keywords

Scientific workflows, data-parallel patterns, bioinformatics, next generation sequence analysis

*Corresponding author.

1. INTRODUCTION

Next-generation DNA sequencing [23] machines are generating a very large amount of sequence data with applications in genomics and metagenomics, biological and biomedical science, environmental science, ecology, and many other fields. However, these new sequencing technologies pose even newer data, computation, network, and complexity management challenges to bioinformatics in comparison to the previously identified issues [8].

1.1 Challenges

With the introduction of next-generation sequencers, *e.g.*, the 454 Sequencer [17], there has been a huge increase in the amount of DNA sequence data [25]. These second generation and the emerging third generation sequencing technologies are providing extremely huge data that overwhelms current computational tools and resources. For example, the Illumina HiSeq 2000[1] can produce two billion paired-end reads per run (200 GB). The traditional bioinformatics scenario of downloading data locally to analyze it does not scale at these huge magnitudes. Executing BLAST [4] on datasets of such sizes would consume thousands of CPU-years. Even with ultra-fast mapping programs, such as BWA [15], it would still take months to years of CPU time. In addition, as datasets get larger, moving data over the network becomes more complicated and error-prone.

This enormous data growth places unprecedented demands on traditional single-processor bioinformatics algorithms. Efficient and comprehensive analysis of the generated data requires distributed and parallel processing capabilities. Bioinformaticians often conduct parallel computation by splitting their queries or databases to turn them into smaller jobs that are executed in cluster, Grid, or Cloud environments, and then merge the results. New computational techniques and efficient execution mechanisms for this data-intensive workload are needed. Technologies like scientific workflows [27] and data-intensive computing [13] promise new capabilities to enable rapid analysis of these next-generation sequence data. These technologies, when used together in an integrative architecture, have great promise to serve many projects

[1]Illumina HiSeq 2000: http://www.illumina.com/systems/-hi-seq_2000.ilmn, 2012.

with similar needs on emerging distributed data-intensive computing resources.

To date, there have been a number of studies for data-intensive analysis of large-scale bioinformatics datasets on Cloud computing platforms. For example, the CloudBurst algorithm [22] has demonstrated the capability of MapReduce [6] to parallelize the execution of the RMAP read-mapping algorithm [24] on multiple compute nodes. However, even with the existence of easy access to data and computational Grid or Cloud resources, there are still several obstacles to overcome before domain scientists can benefit from this abundance of data and the powerful computing technologies. Bioinformaticians and other computational biologists still face having to learn how to use multiple technologies to get their science done, leaving non-expert users in a position of using less efficient techniques and achieving lower execution performance, which in turn consumes more time and resources. New higher-level abstractions like MapReduce are needed to support the easy expression of distributed bioinformatics and computational biology analysis that brings together the best practices for accessing bioinformatics data and for using multiple computing technologies. This vision paper explains the approaches taken in the new bioKepler project to help with these challenges and presents preliminary results demonstrating these approaches.

1.2 Kepler Scientific Workflow System

A scientific workflow is the process of combining data and processes into a configurable, structured set of steps that implement semi-automated computational solutions to a scientific problem. The Kepler[2] scientific workflow system [3, 16], is developed by a cross-project collaboration to serve scientists from different disciplines. Since its initiation in 2003, a diverse set of projects encompassing multiple disciplines have used Kepler to manage, process, and analyze scientific data [1]. Inherited from Ptolemy II[3], Kepler adopts the actor-oriented modeling [10] paradigm for design and execution of scientific workflows.

Kepler provides a graphical user interface (GUI) for designing workflows composed of a linked set of components, called *Actors*, that may execute under different *Models of Computations* (MoCs) [11] implemented as *Directors*. Actors are the implementations of specific functions that need to be performed and communication between actors takes place via tokens that contain both data and messages. Directors specify what flows as tokens between the actors, how the communication between the actors is achieved, when actors execute (a.k.a. fire), and when the overall workflow can stop execution. The designed workflows can then be executed through the same user interface or in batch mode from other applications. In addition, Kepler also provides a provenance framework [2] that keeps a record of chain of custody for data and process products within a workflow design and execution. This helps track the origin of scientific end products, and validate and repeat experimental processes that were used to derive these scientific products.

1.3 Vision

As identified in Section 1.1, for enabling bioinformaticians and computational biologists to conduct efficient analyses, there still remains a need for higher-level abstractions on top of scientific workflow systems and distributed computing methods. Specifically, three challenges remain unsolved:

(a) How can large-scale sequencing data be analyzed systematically in a way that incorporates and enables reuse of best practices by the scientific community?

(b) How can such analysis be easily configured or programmed by end users with various skill levels to formulate actual bioinformatics workflows?

(c) How can such workflows be executed in computing resources available to scientists in an efficient and intuitive manner?

Based on this motivation and our previous experiences in bioinformatics and distributed scientific workflows as explained above, we are creating a Kepler Scientific Workflow System module, called "bioKepler", that facilitates the development of Kepler workflows for integrated execution of bioinformatics applications in distributed environments. To develop such an environment, we build scientific workflow components to execute a set of bioinformatics tools using distributed data-parallel execution patterns. Once customized, these components are executed on multiple distributed platforms including various Cloud and Grid computing platforms.

1.4 Related Work

Scientific workflows using distributed execution patterns have been an active area of study over the last couple of years. Several groups have developed support for distributed execution patterns, such as Map and Reduce constructs in VIEW [9], map, foldr and foldl constructs in Martlet [12], MapReduce [29] and IterateOverArray [16] actors in Kepler, and implicit iteration in Taverna [20]. The MasterSlave actor [28] in Kepler and Service Distribution in Triana [26] can distribute data to multiple remote engines and run them in parallel. The PACT programming model [5] in the Stratosphere system[4] supports Cross, Match, and CoGroup data-parallel patterns in addition to Map and Reduce. A PACT program is transformed into a dataflow graph that can be executed by the Nephele execution engine [30] to realize optimized processing on Cluster and Cloud environments.

In addition to pattern-based distributed execution, current scientific workflow systems also provide distributed execution support using special built-in architectures and atomic workflow constructs. Scientific workflow systems like Kepler, Pegasus [7], Swift [31], and ASKALON [21] also support distributed job execution in Cluster, Grid or Cloud environments.

Some frameworks such as CloudBurst [22] and Crossbow [14] rebuild traditional single-processor bioinformatics tools into the MapReduce parallel pattern. CloudBurst and Crossbow demonstrate the capability of patterns, *e.g.*, MapReduce, to parallelize execution of traditional bioinformatics tools on multiple compute nodes.

However, to the best of our knowledge, there are no comprehensive distributed data-parallel bioinformatics tools and supporting workflow systems so that the bioinformatics tools can be easily integrated and efficiently scheduled for different computing environments.

[2]Kepler website: http://kepler-project.org/, 2012.

[3]Ptolemy II website: http://ptolemy.berkeley.edu/ptolemy-II/, 2012.

[4]Stratosphere Project: http://www.stratosphere.eu/, 2012.

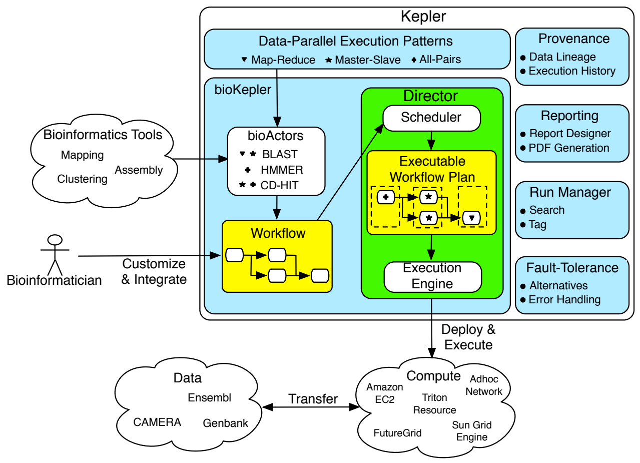


Figure 1: Conceptual framework of the bioKepler module and its interactions of other systems and tools.

2. CONCEPTUAL FRAMEWORK

The bioKepler module that is under development will enable harnessing the potential of distributed computing in bioinformatics. bioKepler will provide a comprehensive list of bioinformatics programs and tools that are incorporated in reusable components, called "bioActors", that can visually be linked together and be executed in distributed environments. A built-in mechanism to execute bioActors and bioKepler scientific workflows on diverse distributed platforms using data-parallel patterns is also being implemented as explained in Section 3.

We believe this approach will enhance the users' ability to express their domain-specific data-intensive applications in a natural way using high-order programming abstractions on top of a dataflow paradigm. The bioKepler module will also provide fault-tolerance and provenance support by building upon the other modules in Kepler.

Figure 1 illustrates a conceptual framework that includes major components of the bioKepler module along with other Kepler modules bioKepler interacts with and the computational and data resources the module is being tested on. Below, we explain the major system components shown in Figure 1.

- *Data-Parallel Execution Patterns:* Several generic actors have been implemented in Kepler to support data-parallel execution in distributed environments, *e.g.*, MapReduce [29] and MasterSlave [28]. We are extending this generic higher-order actor set to support more data-parallel execution patterns, such as All-Pairs [18], and Match and CoGroup in PACT [5]. These higher-order actors will reused to build domain-specific bioActors.

- *bioKepler:* The bioKepler module contains a specialized set of actors, namely *bioActors*, and a *Director* to deploy and efficiently execute workflows on a diverse array of computing resources.

  bioActors are Kepler actors that implement specific Bioinformatics Tools such as BLAST and HMMER. Additionally, each bioActor may implement one or more of the data-parallel execution patterns, *e.g.*, MapReduce or MasterSlave as shown in Figure 1. Ideally, every actor should support every pattern, but this is dependent on the data and computation requirements and utilization of the underlying tool. During workflow construction, the user may choose the pattern for a particular actor, *e.g.*, use MapReduce for BLAST, or let the Director make this decision during run-time. Unlike existing parallelized bioinformatic tools, such CloudBurst [22] and Crossbow [14], which have a fixed data-parallel pattern, bioActors can support multiple data-parallel patterns. The best data-parallel pattern for each bioActor can be chosen and customized at runtime based on the characteristics of each concrete execution, *e.g.*, dataset sizes, and available computational resources and software. In this case, the user does not need to know anything about data-parallel patterns since they are automatically chosen by the director.

  The workflow created by the user integrates many actors to perform a sequence of tasks. However, the most efficient way to perform these tasks depends on the available computational resources, and the user should not be burdened with the complexities of distributed computing and data transfer. The Director is responsible for the overall execution of a workflow. It uses a Scheduler to transform the workflow created by the user into an Executable Workflow Plan, and an Execution Engine to run the workflow on a set of computing resources. The bioKepler Scheduler will transform a user-created workflow into a Executable Workflow Plan that is optimized for the given resources. The Scheduler may decide that one actor should use MapReduce while another uses All-Pairs. In addition to applying the data-parallel execution patterns to bioActors, the Scheduler may split the execution of the workflow across different nodes. The Executable Workflow Plan can then be deployed and executed by the Execution Engine, which supports a diverse array of computational resources including Amazon's Elastic Compute Cloud[5], FutureGrid[6], and UCSD's Triton Resource[7].

- *Additional Kepler Modules:* The *Provenance* module captures provenance information about workflows, including the specification or structure of the workflow, *e.g.*, the actors, directors, parameters, etc., and execution information such as the data transferred between actors. Provenance information is written to a database and can be retrieved by other modules through a query API. The *Run Manager* is a graphical interface for viewing and organizing workflow runs stored in the provenance database. The user may tag past runs with keywords or semantic annotations, and search for runs based on tags or other metadata. Additionally, the Run Manager facilitates sharing workflow runs by providing the ability to import and export from the database. The *Reporting* module creates reports based on workflow executions. The report layout and formatting is specified via a graphi-

[5]Amazon EC2: http://aws.amazon.com/ec2/, 2012.
[6]FutureGrid Portal: https://portal.futuregrid.org/, 2012.
[7]Triton Resource: http://tritonresource.sdsc.edu/, 2012.

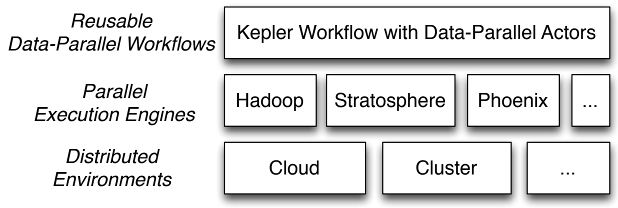

Reusable Data-Parallel Workflows
Kepler Workflow with Data-Parallel Actors
Parallel Execution Engines
Hadoop
Stratosphere
Phoenix
...
Distributed Environments
Cloud
Cluster
...

Figure 2: Three layers of data-parallel workflow design and execution.

cal user interface called the Report Designer and after workflow execution is complete, a PDF report is produced using data from the provenance database. The *Fault-Tolerance* module provides error-detection and handling mechanisms. A Contingency actor facilitates workflow recovery by allowing the execution of alternative sub-workflows when the primary sub-workflow fails [19].

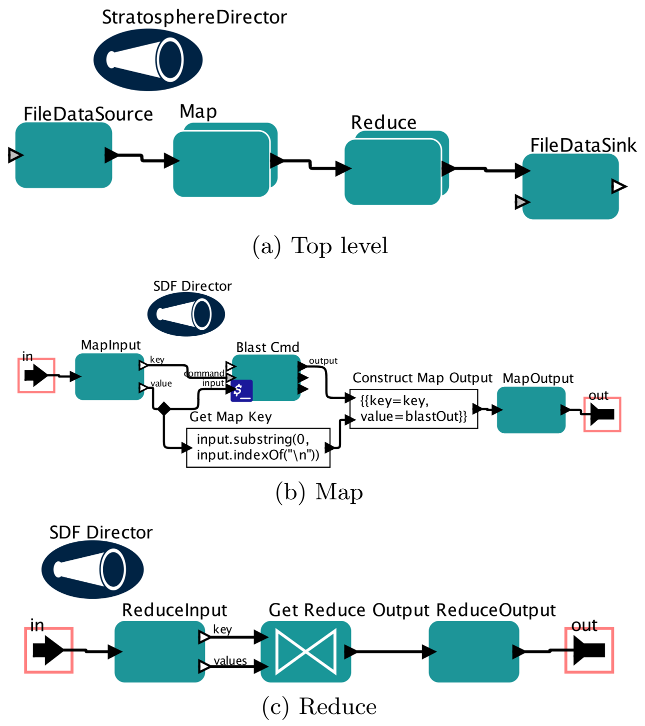
Through this framework, bioinformaticians and computational biologists will be able to use parallelized bioinformatics tools directly, compose them into larger scientific workflows, and execute them efficiently on diverse distributed environments.

3. PRELIMINARY RESULTS

In the first stage of the bioKepler project, we are currently building bioActors that can use data-parallel patterns for efficient execution. As illustrated in Figure 2, there are many frameworks supporting different data-parallel patterns, *e.g.*, Hadoop and Phoenix[8] support Map and Reduce, and Stratosphere's PACT/Nephele model [5] includes three additional patterns as explained in Section 1.4. While Figure 2 shows only a limited set of parallel execution engines and distributed environments, our vision is to extend it with more components as we build them into bioKepler. In addition, we expect the reusable components and workflows layer in Figure 2 to not only be easy-to-use for end users but also be adaptive to different execution engines and distributed environments. The challenge we are currently working on is building and executing bioActors based on these similar but different frameworks.

A new director for Kepler has been implemented to use the Stratosphere system. The *StratosphereDirector* translates workflows composed of specialized PACT actors, *e.g.*, Map and Reduce actors in Figure 3(a), into PACT programs and runs them in the Nephele Execution Engine [30]. The PACT programming model supports several types of input contracts, *e.g.*, Map, Reduce, Cross, etc., for user-supplied first-order functions. The StratosphereDirector allows users to specify these first-order functions either by creating a sub-workflow in Kepler or choosing from a set of predefined Java classes. In the former case, the Nephele execution engine runs the Kepler execution engine as the first-order function, and in turn the Kepler engine runs the sub-workflow created by the user. The Stratosphere Director also supports actors to read and write data from the file system using the *FileDataSource* and *FileDataSink* actors. These actors have

[8]Phoenix System: http://mapreduce.stanford.edu/, 2012.



(a) Top level

(b) Map

(c) Reduce

Figure 3: BLAST workflow.

configurable parameters to specify the file path and how to split and join the data.

The BLAST tool [4] detects the similarities between two biological sequences, and is one of the most widely used tools in bioinformatics. Executing BLAST can be very data-intensive since the query or reference data may have thousands to millions of sequences. Figure 3(a) shows a Kepler workflow that runs BLAST through a data-parallel approach. The Kepler actors in the data-parallel BLAST workflow, *i.e.*, *FileDataSource*, *Map*, *Reduce*, and *FileDataSink*, each have a corresponding Contract [5] in the PACT programming model. *FileDataSource* is configured to read a query sequence file, and split the query sequences into multiple subsets. Each subset will have a partial number of query sequences and the splits can be processed in parallel. Both *Map* and *Reduce* have sub-workflows as shown in Figures 3(b) and 3(c), respectively. The Map sub-workflow executes BLAST to process a subset of the query sequences against the reference database, and the Reduce sub-workflow sorts the BLAST outputs based their key. *FileDataSink* writes the sorted BLAST outputs into a single file.

Our preliminary experiments of the above workflow show a near linear execution acceleration when running on multiple compute nodes. Additionally, since BLAST can be parallelized by partitioning only the reference database or partitioning both the reference database and query sequence file, we are composing corresponding workflows support these two additional parallelization solutions. These workflows use the Map, Cross, and Reduce Contracts in PACT. We will compare the performance differences of these three workflows and identify the most applicable environment and configuration to run each workflow.

We are re-implementing our *Kepler + Hadoop* integration

[29] to have a new *Hadoop Director*, which will convert its workflow into Hadoop programs that can be executed in a Hadoop cluster. We expect the Hadoop and Stratosphere Directors can be used interchangeably to execute the same workflow on their associated engines simply by just changing the director. Since this design separates the user-created workflows from the underlying parallel execution engines, *e.g.*, Hadoop, Stratosphere, and Phoenix, users can easily switch from one execution engine to another by just switching the data-parallel director for their workflows. Further, we plan to combine the functionality of these directors into a generic one that automatically chooses and uses the "best" execution engine based on environmental settings, the user's configuration, and dataset sizes.

Figure 2 illustrates this interaction of the above workflows with the underlying supporting systems, where users do not need to know the different frameworks that support data-parallel patterns and write separate codes to work with them. They only need to compose data-parallel workflows based on existing higher-order actors, *e.g.*, bioActors, via Kepler's graphical user interface. The composed workflow can be executed through various engines supported by the directors for each data-parallel execution engine. In addition, through a set of customization parameters, these workflows and data-parallel tasks within them can be configured to run on available distributed resources.

4. CONCLUSIONS

This paper presents the challenges of data analysis for large-scale biological data and our approaches for answering some of these challenges using scientific workflows. We believe that the analysis of next-generation sequencing data can be accelerated by embracing scientific workflows and data-intensive techniques. Our preliminary work to prototype such a vision shows promising results on facilitating workflow usability, execution efficiency, and adaptability.

Based on our previous experience and related work [22, 14], we argue there is no generic methodology that can be applied to all bioinformatics tools for their parallelization. Each bioinformatics tool must be examined to determine if and how it can be parallelized. We are currently collaborating with a group of bioinformaticians to evaluate which biological tools can be implemented as higher-order data-parallel actors and to categorize the bioinformatics tools we evaluate. Based our experience in this process, we plan to create guidelines to facilitate parallelizing other scientific tools.

As other next steps, we plan to extend the *Kepler + Hadoop* framework as explained in Section 3. Since similar data-intensive challenges are faced by many other scientific disciplines besides biology, we plan to evaluate how our practices and results can benefit other scientific domains.

5. ACKNOWLEDGMENTS

The authors would like to thank the rest of Kepler and CAMERA teams for their collaboration. This work was supported by NSF SDCI Award OCI-0722079 for Kepler/CORE, NSF ABI Award DBI-1062565 for bioKepler, the Gordon and Betty Moore Foundation award to Calit2 at UCSD for CAMERA, and an SDSC Triton Research Opportunities grant.

6. REFERENCES

[1] I. Altintas, O. Barney, Z. Cheng, T. Critchlow, B. Ludaescher, S. Parker, A. Shoshani, and M. Vouk. Accelerating the scientific exploration process with scientific workflows. *Journal of Physics: Conference Series*, 46:468–478, 2006. SciDAC 2006.

[2] I. Altintas, O. Barney, and E. Jaeger-Frank. Provenance collection support in the kepler scientific workflow system. In *Proceedings of International Provenance and Annotation Workshop*, pages 118–132, 2006.

[3] I. Altintas, C. Berkley, E. Jaeger, M. Jones, B. Ludaescher, and S. Mock. Kepler: An extensible system for design and execution of scientific workflows. In *Intl. Conference on Scientific and Statistical Database Management (SSDBM)*, Santorini Island, Greece, 2004.

[4] S. F. Altschul, W. Gish, W. Miller, E. W. Myers, and D. J. Lipman. Basic Local Alignment Search Tool. *Journal of Molecular Biology*, 215(3):403 – 410, 1990.

[5] D. Battré, S. Ewen, F. Hueske, O. Kao, V. Markl, and D. Warneke. Nephele/PACTs: A programming model and execution framework for web-scale analytical processing. In *Proceedings of the 1st ACM symposium on Cloud computing*, SoCC '10, pages 119–130, New York, NY, USA, 2010. ACM.

[6] J. Dean and S. Ghemawat. Mapreduce: Simplified data processing on large clusters. *Communications of the ACM*, 51(1):107–113, 2008.

[7] E. Deelman, G. Mehta, G. Singh, M.-H. Su, and K. Vahi. Pegasus: Mapping large-scale workflows to distributed resources. In I. J. Taylor, E. Deelman, D. B. Gannon, and M. Shields, editors, *Workflows for e-Science*, pages 376–394. Springer London, 2007.

[8] T. Disz, M. Kubal, R. Olson, R. Overbeek, and R. Stevens. Challenges in large scale distributed computing: bioinformatics. In *Proceedings of Challenges of Large Applications in Distributed Environments, 2005. CLADE 2005.*, pages 57 – 65. IEEE, 2005.

[9] X. Fei, S. Lu, and C. Lin. A mapreduce-enabled scientific workflow composition framework. In *ICWS '09: Proceedings of the 2009 IEEE International Conference on Web Services*, pages 663–670, Washington, DC, USA, 2009. IEEE Computer Society.

[10] A. Goderis, A. Brooks, I. Altintas, C. Goble, and E. Lee. Composing different models of computation in Kepler and Ptolemy II. *Lecture Notes in Computer Science*, III:182–190, 2007. Proc. 2nd International Workshop on Workflow systems in e-Science in conjunction with ICCS 2007.

[11] A. Goderis, C. Brooks, I. Altintas, E. Lee, and C. Goble. Heterogeneous composition of models of computation. *Future Generation Computer Systems*, 25(5):552–560, 2009.

[12] D. J. Goodman. Introduction and evaluation of martlet: a scientific workflow language for abstracted parallelisation. In *WWW '07: Proceedings of the 16th international conference on World Wide Web*, pages 983–992, New York, NY, USA, 2007. ACM.

[13] I. Gorton, P. Greenfield, A. S. Szalay, and

R. Williams. Data-intensive computing in the 21st century. *IEEE Computer*, 41(4):30–32, 2008.

[14] B. Langmead, M. C. Schatz, J. Lin, M. Pop, and S. L. Salzberg. Searching for snps with cloud computing. *Genome Biology*, 10(134), November 2009.

[15] H. Li and Z. Durbin. Fast and accurate short read alignment with burrows-wheeler transform. *Bioinformatics*, 25(14):1754–1760, 2009.

[16] B. Ludaescher, I. Altintas, C. Berkley, D. Higgins, E. Jaeger-Frank, J. Jones, M.and Lee, and Y. Zhao. Scientific workflow management and the Kepler system. *Concurrency and Computation: Practice & Experience, Special Issue on Scientific Workflows*, 18(10):1039–1065, 2006.

[17] M. Margulies, M. Egholm, W. Altman, S. Attiya, J. Bader, L. Bemben, J. Berka, M. Braverman, Y. Chen, Z. Chen, S. Dewell, L. Du, J. Fierro, X. Gomes, B. Godwin, W. He, S. Helgesen, C. Ho, G. Irzyk, S. Jando, M. Alenquer, T. Jarvie, K. Jirage, J. Kim, J. Knight, J. Lanza, J. Leamon, S. Lefkowitz, M. Lei, J. Li, K. Lohman, H. Lu, V. Makhijani, K. McDade, M. McKenna, E. Myers, E. Nickerson, J. Nobile, R. Plant, B. Puc, M. Ronan, G. Roth, G. Sarkis, J. Simons, J. Simpson, M. Srinivasan, K. Tartaro, A. Tomasz, K. Vogt, G. Volkmer, S. Wang, Y. Wang, M. Weiner, P. Yu, R. Begley, and J. Rothberg. Genome sequencing in microfabricated high-density picolitre reactors. *Nature*, 437(7057):376–380, September 2005.

[18] C. Moretti, H. Bui, K. Hollingsworth, B. Rich, P. Flynn, and D. Thain. All-pairs: An abstraction for data-intensive computing on campus grids. *IEEE Transactions on Parallel and Distributed Systems*, 21:33–46, 2010.

[19] P. Mouallem, D. Crawl, I. Altintas, M. A. Vouk, and U. Yildiz. A fault-tolerance architecture for kepler-based distributed scientific workflows. In *Proceedings of Scientific and Statistical Database Management, 22nd International Conference (SSDBM 2010)*, volume 6187 of *Lecture Notes in Computer Science*, pages 452–460, Berlin, Heidelberg, 2010. Springer.

[20] T. Oinn, M. Addis, J. Ferris, D. Marvin, M. Senger, M. Greenwood, T. Carver, K. Glover, M. R. Pocock, A. Wipat, and P. Li. Taverna: a tool for the composition and enactment of bioinformatics workflows. in. *Bioinformatics, Oxford University Press, London, UK*, 20(17):3045–3054, 2004.

[21] J. Qin and T. Fahringer. Advanced data flow support for scientific grid workflow applications. In *SC '07: Proceedings of the 2007 ACM/IEEE conference on Supercomputing*, pages 1–12, New York, NY, USA, 2007. ACM.

[22] M. Schatz. Cloudburst: Highly sensitive read mapping with mapreduce. *Bioinformatics*, 25(11):1363–1369, April 2009.

[23] J. Shendure and H. Ji. Next generation-dna sequencing. *Nature Biotechnology*, 26(10):1135–1145, 2008.

[24] A. D. Smith, Z. Xuan, and M. Q. Zhang. Using quality scores and longer reads improves accuracy of solexa read mapping. *BMC Bioinformatics*, 9(128), February 2008.

[25] L. D. Stein. The case for cloud computing in genome informatics. *Genome Biology*, 11(5):207, 2010.

[26] I. Taylor, M. Shields, I. Wang, and O. Rana. Triana applications within grid computing and peer to peer environments. *Journal of Grid Computing*, 1, 2003.

[27] I. J. Taylor, E. Deelman, D. B. Gannon, and M. Shields, editors. *Workflows for e-Science*. Springer, 2007.

[28] J. Wang, I. Altintas, P. R. Hosseini, D. Barseghian, D. Crawl, C. Berkley, and M. B. Jones. Accelerating parameter sweep workflows by utilizing ad-hoc network computing resources: An ecological example. In *Services, IEEE Congress on*, pages 267–274. IEEE Computer Society, 2009.

[29] J. Wang, D. Crawl, and I. Altintas. Kepler + Hadoop: A general architecture facilitating data-intensive applications in scientific workflow systems. In *WORKS '09: Proceedings of the 4th Workshop on Workflows in Support of Large-Scale Science*, pages 1–8, Portland, Oregon, 2009. ACM New York, NY, USA.

[30] D. Warneke and O. Kao. Exploiting dynamic resource allocation for efficient parallel data processing in the cloud. *Parallel and Distributed Systems, IEEE Transactions on*, 22(6):985 –997, June 2011.

[31] Y. Zhao, M. Hategan, B. Clifford, I. Foster, G. V. Laszewski, I. Raicu, T. Stef-praun, and M. Wilde. Swift: Fast, reliable, loosely coupled parallel computation. In *Services, 2007 IEEE Congress on*, pages 199 – 206. IEEE Press, 2007.