Toward a Data Analysis Grid for Biometrics Research

Douglas Thain and Patrick Flynn

Technical Report 2008-05
Department of Computer Science and Engineering
University of Notre Dame
9 April 2008

Abstract

Advanced research in biometrics presents new opportunities and challenges in grid computing. Biometric workloads are both data and computation intensive, and have the potential to be accelerated by employing large numbers of machines. Many large workloads follow a common high level structure and could be made robust and scalable by custom high level abstractions. However, these workloads also have non-trivial security constraints to ensure the privacy of participants, and present new problems in large scale storage management. In this paper, we give an overview of the needs of a biometric research group, and discuss the research problems in grid computing that result.

A Brief Overview of Biometrics

Biometrics is the study of identifying people from physical observations (*modes*) such as fingerprints, iris images, and face images. [WZ2003] [JD2004] In recent decades, the ability to rapidly acquire and process digital recordings has opened up new possibilities for performing authentication by biometric techniques. Although this field already has a robust commercial sector that sells identification systems employing different modes, there are still many challenges to solve before biometric identification becomes widespread and highly reliable. [PP2007] For example:

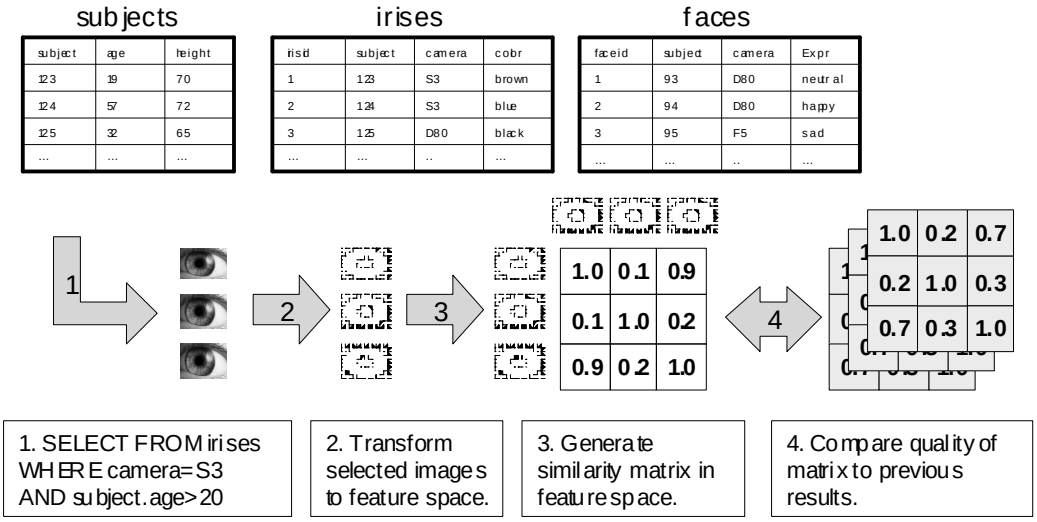
- Current commercial face recognition technologies require images to be taken in a consistent manner with fixed lighting and camera geometry and no major changes in the subject's appearance or facial expression. Are there comparison algorithms that are robust to common changes in the setting or the subject?

- Various types of recording devices can capture the geometry of a face in three dimensions, albeit at lower resolution than a traditional image. Is matching in three dimensions more or less accurate than matching using high quality two dimensional images?

- Continuous video streams collect a large amount of data about subjects, but it is of significantly lower quality than deliberate photographs of subjects. What degree of matching accuracy can be obtained from video streams?

In order to explore these problems, a researcher must have access to large corpus of recordings in the desired mode, annotated with relevant information about the subject and recording conditions. For example, a current image acquisition at the University of Notre Dame involves collecting face images from the same subjects on a weekly basis over the course of a year. Each image is taken in a different location, under different lighting conditions, and with a different facial expression. Acquisition is performed under the supervision of a trusted proctor, who carefully

A Sample Workflow in Biometrics



arranges the subject and manually records metadata according to the overall goal of the study. Such data is collected in a lab, placed in temporary storage, and eventually is migrated into a central repository. Clearly, such a dataset is time consuming and expensive to acquire.

Once obtained, a researcher will wish to perform a large number of experiments, performing data reduction and comparison on different subsets of the population. For example, consider the hypothesis that the shape of the nose is a suitable feature for face recognition. To explore this hypothesis, a researcher must create two functions. One function reduces a bitmapped image into a feature set, which might be the physical geometry of the nose. The other function compares two features sets to each other, returning a value between zero and one, indicating the similarity of the two feature sets. Then, all images of interest must be transformed into the feature space and compared all to each other, generating a matrix of results where each cell represents the similarity of any two images. From the source metadata, we know which images are actually of the same subject, so the similarity matrix may be evaluated objectively using some distance metric from the ideal results or another comparison technique.

(In these studies, brute force comparison is unavoidable. For unknown matching algorithms, we must observe that matches return high values and non-matches return low values. Other research has shown that early discard [LH2004] or clustering based on features [RB2007] can reduce search times in on-line matching problems. This is only possible when the behavior of the function is already known and accepted for production use.)

There are many small variations upon this procedure. If we wish to compare the effectiveness of nose shape versus ear shape for identification, the procedure is repeated to generate another matrix. Functions for transformation to feature space and the comparison of features have an infinite number of variations that are still an open topic of research in all modes and matching architectures. Of course, researchers designing competing algorithms should process the same set of data, so that the results are directly comparable.

Although easily stated, such workloads are both computation and data intensive. In recent work [CM2008] we demonstrated a workload that compared 6000 iris feature sets all to each other. With extensive manual tuning and continuous supervision, we recently carried out a workload that compared 6000 iris images of 1.25MB each all to each other. Each comparison required about one second of CPU time, for a total of 185 CPU-days of work, which was reduced to about one day on a local batch system of 200 nodes. Although the total amount of input data (7.5GB) is not troublesome by itself, any significant fraction of the computation needs all of the data, and so it must be distributed to all nodes of the system. This required the careful management of storage and network resources to avoid upsetting other users of the system. Given access to larger resources such as the Open Science Grid or the TeraGrid, researchers could attack even larger problems, or produce more thorough results in the same amount of time. If it were possible to easily run hundreds of such workloads, then researchers could quantitatively compare a large number of feature extraction and comparison algorithms.

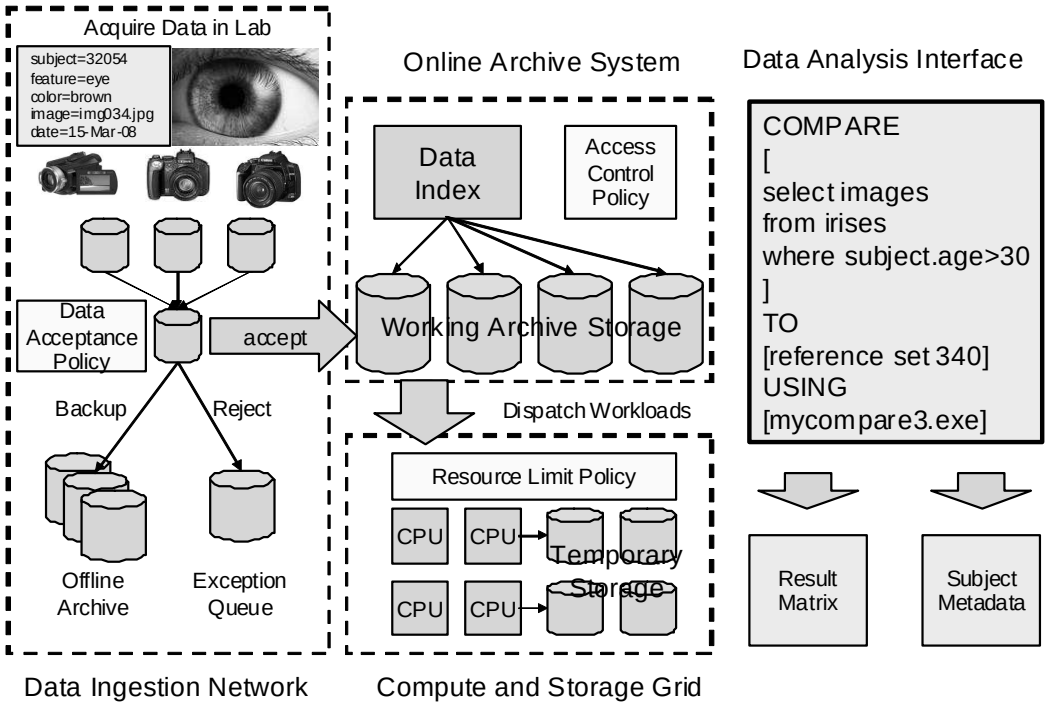
Unfortunately, biometric datasets cannot be generally distributed on the grid. There are several possible hazards. The most serious would be the exposure of a high quality recording coupled with personal information about the subject. With a good iris image and the knowledge that it belonged to one of the authors, an evildoer might be able to successfully impersonate one of us in a future authentication check. Even if such an attack is impractical today (our offices are secured by keys, not iris scans), a data loss today might pay off in the future when such technologies are more widespread. Even without the metadata, access to a large body of data might permit a brute force attack on a future authentication check. One could distribute features extracted from biometric samples instead of the samples themselves to purportedly avoid identity disclosure. However, some biometric features are invertible in that they can be used to reconstruct a synthetic biometric sample that could, under certain conditions, approximate the sample from which the feature set was derived. A less technical but still serious hazard would be an evildoer who obtains an identifiable picture or video, modifies it in an unsavory way, and distributes it widely to embarrass the subject.

For these reasons, we cannot construct a conventional data grid that makes all data available to a significant user base over the wide area. And yet, we wish to have a system where researchers collaborating over a wide area can exploit a common workspace where they may process and compare results based on the same input data. For this we need a data analysis grid.

Toward a Data Analysis Grid for Biometrics

To address these challenges, we are designing and constructing a *data analysis grid* for biometrics. A data analysis grid has a public interface for browsing and analyzing data using high level actions that may potentially consume large amounts of storage and computation, along with policies that control data access and resource consumption. A data analysis grid is different than a traditional computational grid because it does not provide access to CPUs for arbitrary tasks, and it is different than a traditional data grid because it does not disseminate large amounts of data to multiple participants. We envision a system with the following components:

- **Online Archive.** The centerpiece of the system is an online archive that serves as both the data source and output target of all investigations. Multiple copies of each data item are replicated across a storage cluster and are indexed by a central database that stores the experimental metadata and the location and state of each data file. Queries made against the system can be performed to explore both the data and the metadata; both can be moved to the adjacent computing and storage grid for large scale processing. A detailed



access policy controls what data can be browsed, moved to the processing grid, or exported directly to the client.

- **Data Analysis Interface**. Users interact with the system through a high level interface that is tailored to the needs of biometric research. A fairly small number of fundamental operations can be used to support a wide variety of research: selecting datasets based on metadata properties, reducing the elements to a feature space, comparing datasets to each other, and exporting the similarity matrix. Although easily stated, each of these fundamental operations becomes complex and long running when applied to a dataset of any significant size. End users will also require a workload estimation service so that they will have some warning whether an action will take a day or a year.

- **Computing and Storage Grid.** Analysis tasks that consume a significant amount of time and resources may be offloaded to a general purpose computing and storage grid that may be shared with other applications at the same institution. As noted above, not all data is suitable for export, so sensitive stages of a workload may execute first on the archive, and later stages may be moved to the computing grid. It may be necessary to transfer a large amount of data from the archive into the computing grid, so the system must have a mechanism and policy to tracking storage use and arbitrating between users of the system.

- **Data Ingestion Network.** In order to facilitate the correct, robust, and rapid ingestion of data into the system, we require a store-and-forward data ingestion network. As data is acquired, either by a manual process or an automatic sensor, it is annotated with relevant metadata and initially stored on a device close to the sensor. A local device is necessary to support the burst I/O needs to the sensor, but is unlikely to have long term reliability.

An automatic process detects newly acquired data and forwards it over the network to an intermediate storage node, where the data is validated to determine whether it is suitable for addition to the archive. If so, it is ingested into the archive and duplicated on offline storage. If the data is not valid, it will be moved to an exception queue where a human can examine it, and then either repair and resubmit, or discard the data. End-to-end error checking can be used to ensure that data is not damaged in transit.

Open Research Issues

Many of the functional aspects of this data analysis grid have been studied by previous research, thus we expect to exploit a number of existing software packages to construct the system. For example, the Storage Resource Broker [CB1998] may be used to manage data across multiple storage devices, annotated with searchable metadata. Condor [DT2004] may be used to manage a multi user compute grid, while Chirp [DT2008] may be used to harness the internal storage of the same grid. GSI [IF1998] will allow for a common authentication infrastructure across the system, and data grid tools such as RLS [SV2001] and RFT [WA2004] can be used to track and manipulate data across the system. Pegasus [ED2004] may be used to define and execute workflows that manipulate distributed data.

However, there are a number of open research problems that would not be addressed by the simple combination of existing software. Here, we outline some of those challenges.

Top-Down Workflow Abstractions

Existing tools for constructing complex workflows are bottom-up tools. That is, the user must completely specify all of the individual underlying activities – copy a file, execute a job – and them connect them together into a graph. This model has been quite successful at complex, irregular tasks such processing satellite data [DAG-Sloan] and medical images [Swift]. However, because all of the precise activities must be specified in advance, it becomes more difficult for the workflow to adapt to the available resources, or deal with failures in execution.

A top-down abstraction can solve some of these problems for workflows that are highly regular. For example, to simplify the distribution of data to our processing grid, we have constructed a top-down tool in which the user specifies the desire to distribute data to any N nodes that meet certain criteria. The tool then transfers data efficiently using a spanning tree; if any failures occur, it simply looks for another host that meets the criteria and keeps going. If we had attempted the same by using bottom-up construction, the system would be constrained to the exact resources selected at submission time, and would have little recourse except to retry failed transfers endlessly.

If we consider the problem of transforming images into feature space, a bottom-up approach would construct a single job to transform each item, and then submit all of those jobs to a batch system. This could be disastrously inefficient if the individual transformations are short running jobs. Instead, if we take a top-down approach in which the user specifies the intent to transform the entire set using a given function, then the system can group the work into an appropriate number of jobs, based upon the locality of data, the number of available CPUs, and the running time of each function. Our initial work on such abstractions is described in [BR2008] and [CM2008].

Aggregate Storage Management

Moving large amounts of data between the archive and the processing grid presents problems of its own. As we have noted, it is most effective to stage input data sets to all of the local nodes of the processing grid. Although input datasets are commonly measured in gigabytes – no longer a large amount of data – managing a few gigabyte across several hundred machines has its own challenges. It is impractical to stage all data to all nodes simultaneously, which causes all nodes to block on I/O and yields the network unusable for other tasks. Instead, we must have a higher level abstraction that stages data to all nodes in an efficient manner. To do this, we have provided users with a tool that replicates data using a spanning tree, which completes in logarithmic time.

However, we have discovered through hard experience that, once it is easy to replicate data to many nodes, users take advantage of this facility and quickly fill up the available space. Instead, what is needed is a system that tracks storage usage across the cluster and accepts responsibility for deleting datasets once the workflows that depend upon them have completed. Such a system must be robust to a number of failure conditions, particularly the case that a node may be offline when either replication or deletion is requested.

At a higher level, such allocation facilities must be connected to workflow execution and scheduling. A large workflow may need to allocate both temporary space in the processing grid as well as output space in the online archive to store a large similarity matrix. It is important that a workflow not proceed until space is available, otherwise a significant amount of computing resources will be wasted. There are also opportunities for positive optimizations: workflows that process the same input may share a single instance of data staged into the processing grid, provided we are careful to avoid deadlocks on output space.

Resource Estimation

If we provide a simple interface for executing very large workflows, it may be too easy for users to become disconnected from the time and resources necessary to carry out the workflow. A large workflow might consume several thousand CPU-days of computing, occupy a terabyte of storage, consume tens of thousands of dollars in electricity and cooling, and take a week to run to completion. Such a degree of consumption should not be undertaken without at least informing the user, and perhaps should require the assent of the operator of the system. At the very least, a user should have some conception of whether the work requested will take an hour or a year.

To address this problem, we require an estimation service that can examine a proposed workflow and predict how long it will take and what resources will be consumed. Of course, this is completely intractable problem for a general computation on an arbitrary grid. But, for a data analysis grid running on fixed hardware with a constrained set of workflow structures and underlying applications, we hypothesize that it is possible to provide a lower bound on execution time and resource consumption within, say, one order of magnitude.

Workflow Level Security

Most mechanisms for enforcing security policies on data are enforced by the relevant file or database server, which examines the credentials of the requesting user in order to allow or deny access. However, this form of conventional access control would not satisfy the necessary security policies of this system, of which the most important is *raw biometric data may not exit the system*. A user with ordinary privilege might be permitted to execute code that manipulates and a feature space reduction in the archive or on the adjacent compute grid and even retrieve the output data, but cannot arrange for input data to leave the system.

To address this problem, we require a security mechanism that examines a high level workflow, and determines whether it conforms to the site policy. Of course, for this to be feasible, the workflow language must have a compact representation and explicit or derivable representations of the action of that workflow on the system. This is likely not practical for a generalized scripting language, but seems achievable for a language consisting of the four operations shown above.

An interesting sub-problem is how to constrain the behavior of small programs inserted into the workflow. We would like to allow users to define and apply their own operators for reducing data to feature space, and to compare one item to another. It is relatively simple to apply sandboxing techniques so that a malicious function could not simply ship the input data over the network. But, how do we prevent the user from writing a function that simply copies the sensitive input data to the output, where it can be viewed? This would require some observation of the information gain/loss from the input to the output in order to ensure that the feature space is sufficiently abstract.

Multi-Modal Transactions

Most grid related workflow systems have concentrated on the relationship between executables, batch systems, and file systems. Typically, robustness is provided by re-attempting executions, relying on the assumption that most actions are idempotent. However, adding databases and storage allocation systems to the mix introduces new kinds of problems. Neither space allocation nor database updates are necessarily idempotent. In the even of failure or cancellation of a workflow, there must be a robust method for identifying and rolling back incomplete portions of transactions, otherwise the system will quickly fill up with unreferenced garbage.

This problem also extends to the input side of the system. Biometric data acquisition is a bursty and error-prone process. Some data is collected manually by an operator that must work a camera, type subject data, and then save the results when satisfied. Operators can make mistakes by entering incorrect data or saving images of little value. Other data is collected by automatic systems that can also fail in unexpected ways. In both cases, data must be protected against accidental corruption, and data collection cannot stop if the network or the archive is unavailable.

A system to perform asynchronous data movement from node to node is required.
The solution is clearly some form of asynchronous data movement in the spirit of RFT [WA2004], Stork [TK2004], and Kangaroo [DT2001], but must go farther to include inline processing for validation and error-checking, as well as robust integration with the archival system, and a user interface for examining and dispatching exceptional data.

Conclusion

In this paper, we have described the high level goals of biometric research, and motivated the need for a data analysis grid with properties that depart somewhat from the usual goals of data and computing grids. To build a robust system, we require new techniques that exploit high level workflow abstractions, aggregate storage management, workflow level security, resource estimation and multi-modal transactions. Although each of these techniques is motivated by the biometric example, we believe that they will have further application in many other kinds of data intensive systems.

Acknowledgment
This work is supported by National Science Foundation grant CCF-0621434.

References

[BR2008] B. Rich and D. Thain, "DataLab: Transactional Data-Parallel Computing on an Active Storage Cloud", Technical Report 2008-02, Department of Computer Science and Engineering, University of Notre Dame. March 2008.

[CB1998] C. Baru, R. Moore, A. Rajasekar, and M. Wan, "The SDSC Storage Resource Broker", CASCON 1998

[CM2008] C. Moretti, J. Bulosan, D. Thain, and P. Flynn, "All-Pairs: An Abstraction for Data-Intensive Cloud Computing", International Parallel and Distributed Processing Symposium, April 2008.

[DT2003] D. Thain, T. Tannenbaum, and M. Livny, "Condor and the Grid", in Fran Berman, Anthony J.G. Hey, Geoffrey Fox, editors, *Grid Computing: Making The Global Infrastructure a Reality*, John Wiley 2003.

[DT2008] D. Thain, C. Moretti, and J. Hemmes, "Chirp: A Practical Global Filesystem for Cluster and Grid Computing", Journal of Grid Computing, to appear in 2008.

[ED2004] E. Deelman, J. Blythe, Y. Gil, C. Kesselman, G. Mehta, S. Patil, M. Su, J. Vahi, M. Livny, "Pegasus: Mapping Scientific Workflows onto the Grid", Across Grids Conference, 2004.

[IF1998] I. Foster, C. Kesselman, G Tsudik, S. Tuecke, "A security architecture for computational grids," ACM Conference on Computer and Communications Security, 1998.

[JD2004] J. Daugman, "How Iris Recognition Works", IEEE Transactions on Circuits and Systems for Video Technology 14(1), pp. 21 -30, 2004.

[LH2004] L. Huston, R. Sukthankar, R. Wickremsinghe, M Satyanarayanan, G Ganger, E. Reidel, A. Ailamaki, "Diamond: A Storage Architecture for Early Discard in Interactive Search", USENIX File and Storage Technologies, 2004.

[PP2007] P. J. Phillips, W. T. Scruggs, A. J. O'Toole, P. J. Flynn, K. W. Bowyer, C. L. Schott, M. Sharpe, "FRVT 2006 and ICE 2006 Large-Scale Results", National Institute of Standards and Technology, NISTIR 7408, 2007.

[RB2007] R. Bayardo, Y. Ma, and R. Srikant, "Scaling Up All Pairs Similarity Search", World Wide Web Conference, 2007.

[SV2001] S. Vazhkudai, S. Tuecke, and I Foster, "Replica Selection in the Globus Data Grid", IEEE Cluster Computing and the Grid, 2001.

[TK2004] T. Kosar and M. Livny, "Stork: Making Data Placement a First Class Citizen in the Grid", International Conference on Distributed Computing Systems, March 2004.

[WA2004] W. Allcock, I. Foster, R. Madduri, "Reliable Data Transport: A Critical Service for the Grid", Workshop on Building Service Based Grids, 2004.

[WZ2003] W. Zhao, R. Chellappa, P. J. Phillips, A. Rosenfeld, "Face recognition: A literature survey", ACM Computing Surveys 35(4): 399 - 458, December 2003.