Applying Chimera Virtual Data Concepts to Cluster Finding in the Sloan Sky Survey

James Annis[1] Yong Zhao[2] Jens Voeckler[2] Michael Wilde[3] Steve Kent[1] Ian Foster[2,3]

[1] Experimental Astrophysics, Fermilab, Batavia, IL 60510, USA

[2] Department of Computer Science, University of Chicago, Chicago, IL 60637, USA

[3] Mathematics and Computer Science Division, Argonne National Laboratory, Argonne, IL 60439, USA

{annis,skent}@fnal.gov, {voeckler,yongzh }@cs.uchicago.edu, {foster,wilde}@mcs.anl.gov

Abstract

In many scientific disciplines – especially long running, data-intensive collaborations – it is important to track all aspects of data capture, production, transformation, and analysis. In principle, one can then audit, validate, reproduce, and/or re-run with corrections various data transformations. We have recently proposed and prototyped the Chimera virtual data system, a new database-driven approach to this problem. We present here a major application study in which we apply Chimera to a challenging data analysis problem: the identification of galaxy clusters within the Sloan Digital Sky Survey. We describe the problem, its computational procedures, and the use of Chimera to plan and orchestrate the workflow of thousands of tasks on a data grid comprising hundreds of computers. This experience suggests that a general set of tools can indeed enhance the accuracy and productivity of scientific data reduction and that further development and application of this paradigm will offer great value.

1 Introduction

The GriPhyN project [1] is one of several major efforts [2-4] working to enable large-scale data-intensive computation as a routine scientific tool. GriPhyN focuses in particular on *virtual data* technologies that allow computational procedures and results to be exploited as community resources so that, for example, scientists can not only run their own computations on raw data, but also discover computational procedures developed by others and data produced by these procedures [5]. A request to retrieve data on a particular cluster might thus either lead to the retrieval of the requested data from a local or remote database or the scheduling of a computation to produce the data.

One of GriPhyN's scientific collaboration partners is the Sloan Digital Sky Survey (SDSS) [6, 7, 25], a digital imaging survey that will, by the end of 2005, have mapped a quarter of the sky in five colors with a sensitivity two orders of magnitudes greater than previous large sky surveys. The data of the SDSS is being made available online as both a large collection (~ 10 TB) of images and a smaller set of catalogs (~ 2 TB), containing measurements on each of 250,000,000 detected objects.

The availability of this data online is particularly useful if astronomers can apply computationally intensive analyses to it. We consider here the example of identifying clusters of galaxies, which are the largest gravitationally dominated structures in the universe. Such analyses involve sophisticated algorithms and

0-7695-1524-X/02 $17.00 (c) 2002 IEEE

large amounts of computation and thus can, in principle, benefit from the use of distributed computing and storage resources, as provided by Data Grids [8, 9].

We describe here an early exploration of applying virtual data and Data Grid concepts to the cluster identification problem. In this work, we treat cluster catalogs as derived data that are constructed from base data (galaxy catalogs), with the contents of a particular cluster catalog depending on the cluster location algorithm used and on the parameter choice for the algorithm. We use GriPhyN technologies, in particular the Chimera virtual data system [5], to generate cluster catalog data. We demonstrate our ability to encode interesting algorithms and to track the materialization of both final cluster catalog data and useful intermediate products. These early successes suggest that we have been successful both in developing a useful tool for Sloan astronomers and in obtaining important data that will allow us to address critical questions that define our larger research program, namely:

Can we represent the transformations of the problem in a virtual data catalog (VDC)?

Will the overhead of managing the VDC be easier than doing this work in an ad hoc fashion?

Will the derived data be traceable in the manner expected?

Will the computations map onto effective workflow graphs for efficient Grid execution?

When code or data changes, can we identify and rebuild their dependent objects?

Will the virtual data paradigm enhance overall productivity?

More specifically, we demonstrate for the first time—albeit only in prototype form—a general, discipline-independent mechanism that allows scientists in any field to use an off-the-shelf toolkit to track their data production and, with relative ease, to harness the power of large-scale Grid resources.

The work reported here complements and extends other work by the GriPhyN collaboration [10-12]. Also related is work on data lineage in database systems [13-17]. Our work leverages these techniques, but differs in two respects: first, data is not necessarily stored in databases and the operations used to derive data items may be arbitrary computations; second, we address issues relating to the automated generation and scheduling of the computations required to instantiate data products.

2 GriPhyN Tools for the Virtual Data Grid

The current GriPhyN toolkit (VDT V1.0) includes the Globus Toolkit™ [18], Condor and Condor-G [19, 20], and the Grid Data Mirroring Package [21,22].

We apply here a new tool to be included in VDT: the Chimera virtual data system [5]. Chimera supports the capture and reuse of information on how data is generated by computations. It comprises a *virtual data catalog*, used to record virtual data information, and a *virtual data language interpreter* that translates data definition and query operations expressed in a virtual data language (VDL) into virtual data catalog operations (Figure 1).

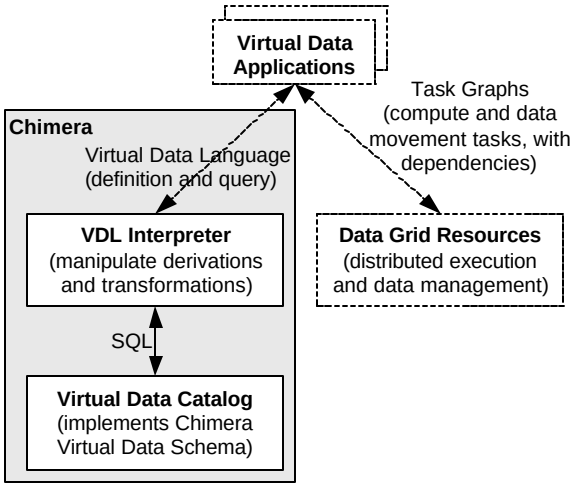

Figure 1: Schematic of the Chimera architecture

The VDC tracks how data is derived, with sufficient precision that one can create and re-create the data from this knowledge. One can then definitively determine how the data was created – something that is often not feasible today in the massive data collections maintained by large collaborations. One can also implement a new class of "virtual data management" operations that, for example, "rematerialize" data products that were deleted, generate data products that were defined but never created, regenerate data when data dependencies or transformation programs change, and/or create replicas of data products at remote locations when re-creation is more efficient than data transfer. This brings the power and discipline that we have so effectively harnessed for producing application *codes* (through mechanisms like "makefiles") to the realm of scientific *data* production.

VDL captures and formalizes descriptions of a how a program can be invoked and records its potential and/or actual invocations. The abstract description of how a program is to be invoked, which parameters it needs, which files it reads as input, what environment is required, and so forth, is called a *transformation.* Each invocation of a transformation with a specific set of input values and/or files is called a *derivation.* As data production proceeds, the execution of all transformations are recorded (either before or after the fact) in the Chimera database, which in effect becomes a central automated archivist of a large scientific collaboration.

VDL query functions allow a user or application to search the VDC for derivation or transformation definitions. The search can be made by search criteria such as input filename(s), output filename(s), transformation name, and/or application name.

Given a request for a "virtual" data object, the Chimera "request planner" can generate a directed acyclic graph (DAG) for use by DAGman [20] to manage the computation of the object and its dependencies. The algorithm for creating the DAG is to first find which derivation contains this file as output, then for each input of the associated transformation find the derivation that contains it as output, iterating until all the dependencies are resolved. Once the DAG is generated, DAGman dynamically schedules distributed computations, submitting them into the Grid via Condor-G, to create the requested file(s).

One can also request the execution of a specific derivation, which proceeds in an identical manner. This becomes the dominant paradigm for running programs in a virtual-data-enabled collaboration: instead of building large, unmanaged libraries of job execution scripts, all scripts are instead described as derivations, and hence their data and code dependencies and outputs are precisely tracked by the Chimera virtual data system. Our main goal in the work we describe here is to test the effectiveness of this new mode of collaboration on a scientific problem of significant but still manageable scope.

3 Finding Clusters of Galaxies in SDSS Data

The cluster detection algorithm on which we tested the virtual data paradigm is "MAXimum likelihood Brightest Cluster Galaxy" (MaxBCG) [25], which features sensitivity to a large dynamic range of cluster masses and very good redshift estimation. Abstractly, the MaxBCG algorithm moves a cylinder around in a five-dimensional space, calculating the cluster likelihood at each point. The 5-space is defined by two spatial dimensions, right ascension (RA) and Declination (Dec); two color dimensions, *g-r* and *r-I*; and one brightness dimension, *i*.

In practice, we perform calculations at the location of a galaxy. Figure 2 illustrates the approach, which is explained in detail in [25] and briefly summarized here. For each galaxy, we calculate whether it is likely to be a luminous red galaxy (BRG). If this likelihood is above a threshold, we compute the likelihood that it is a brightest cluster galaxy by weighting the BRG likelihood by the number of galaxies in the neighborhood. Finally, we ask whether this galaxy is the most likely in the area; if so there is a cluster present centered here and otherwise there is not. In Figure 2, the encircled region shows a spatial acceptance window, which gets smaller with increasing distance.

Table 1 shows the scale of the problem. The problem is reduced to SIMD parallelism by setting an upper limit on the angular size of a cluster: in effect, setting a lower limit on the distance to the cluster. We then work on a central region, with a buffer zone around that region of angular size of the upper limit. We only locate clusters in the central region, but use information from the entire buffer zone for the calculation.

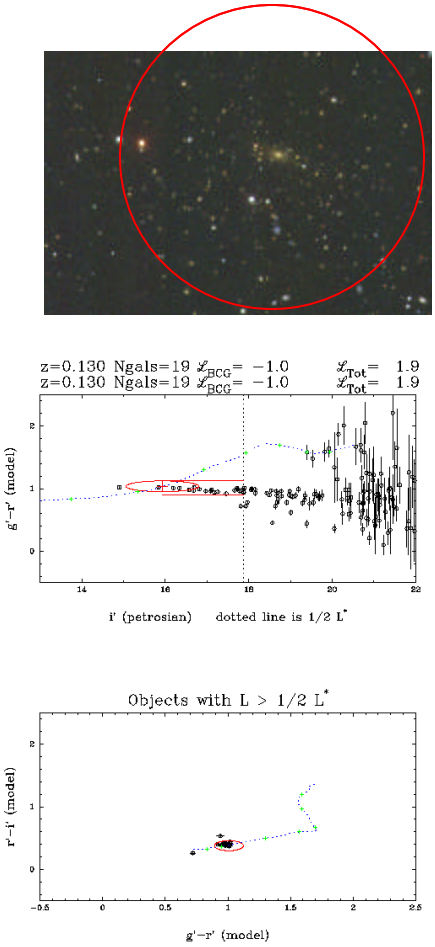

Figure 2: The MaxBCG cluster finding algorithm

Area (square-degrees)	7000
Storage (gigabytes)	1540
Compute (CPU-hours on 500 MHz PIII with 1 gigabyte RAM)	7000

Table 1: Storage and computational requirements for a cluster search covering the full SDSS survey.

4 The GriPhyN Chimera Formulation of Cluster Finding

We describe here our representation of the MaxBCG transformations and then our system design for cluster finding using the virtual data catalog and toolkit components. Our experiences with its use are presented in the next section.

4.1 Representing the Transformations

The MaxBCG algorithm consists of five file-based transformations. The input and output files of each stage form a natural dependency graph, as shown in Figure 3, where the nodes represent data files and the arrows the transformations listed below. The first stage and second stage are straightforward in that each output file corresponds to one input file. The third and fourth stages operate on a "buffer zone" around the target field, and the bcgSearch stage (transformation 3) needs to take both field files and brg files as inputs. The transformations are as follows:

1 - *fieldPrep* extracts from the full data set required measurements on the galaxies of interest and produces new files containing this data. The new files are about 40 times smaller than the full data sets.

2 - *brgSearch* calculates the unweighted BCG likelihood for each galaxy (the BRG likelihood, unweighted by galaxy count, is used to filter out unlikely candidates for the next stage)

3 - *bcgSearch* calculates the weighted BCG likelihood for each galaxy. This is the heart of the algorithm, and the most expensive step.

4 - *bcgCoalesce* determines whether a galaxy is the most likely galaxy in the neighborhood.

5 - *getCatalog* removes extraneous data and stores the result in a compact format.

For illustrative purposes, the VDL specification for the brgSearch program, the simplest of the transformations that implement the MaxBCG algorithm, is shown here and is more fully explained in [25].

```
Begin v @@vdldemo@@/bin/astro.sh
       Arg    brgSearch
       Arg    -f "
       Arg    $run
       Arg    $startField
       Arg    $endField
       Arg    "
       Arg    %runList
       Arg    %camcolList
       File   i parameters.par

       File   i field-%run-%camcol-%field.par
       File   o brg-%run-%camcol-%field.par
End
rc     parameters.par      $root/parameters.par
rc     brg-%run-%camcol-%field.par      $brgDir/brg-%run-%camcol-%field.par
rc    field-%run-%camcol-%field.par $fieldDir/field-%run-%camcol-%field.par
```

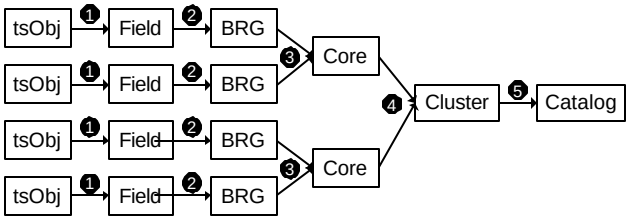

Figure 3: SDSS cluster identification workflow.

Because the MaxBCG algorithm walks through a spatial region around a target galaxy to find neighboring galaxies, the input filenames needed for this region depend on various runtime factors including the buffer size and the run offset between two runs in a stripe. As part of this application, the initial VDL mechanism was enhanced to allow the dynamic generation of file names for derivations during program execution (as opposed to cataloging them before execution). We compute the list of input filenames using an application function that maps a (RA, Dec) pair to a list of filenames. These filenames are then dynamically added to the virtual data catalog as input dependencies to the actual computation step that follows.

4.2 System Architecture

The SDSS software environment, in which MaxBcg is implemented, was integrated with Chimera and a test grid was constructed, as shown in Figure 4. In this integrated system, bulk background data production can be readily performed in parallel with interactive use. In this mode, the virtual data grid acts much like a large-scale cache. If a data product is produced through the batch process before it is needed interactively, then at the time of the interactive request no computations need be scheduled to produce it. If a data product is requested before the batch process has produced it, the required derivations will be executed on demand, and the results stored, eliminating the data item from the batch process work list.

An important aspect of Chimera is the management of input files which contain the critical input parameters to the cluster finding application. Instead of retaining full copies of all needed parameter files and tracking these files in the Chimera database as inputs to derivations, we adopted the following system which was more usable and which placed more salient information into the virtual data catalog. We created "front-end" transformations for each transformation that required an input *file* of parameters. This front-end transformation took the salient parameters (i.e., the ones that typically varied from derivation to derivation) as arguments, inserted those parameters into the *parameter file templates* that contained the numerous seldom-changing parameters, and generated complete parameter files as output. These transformations were then incorporated into the user application DAG to generate the parameter file used by the main application transformation. Thus we needed to maintain only a few template parameter files, from which any number of actual parameter files could be generated. Using this technique, the Chimera database always contained (in the "front-end" derivation records) all parameters required to rerun the main application derivations.

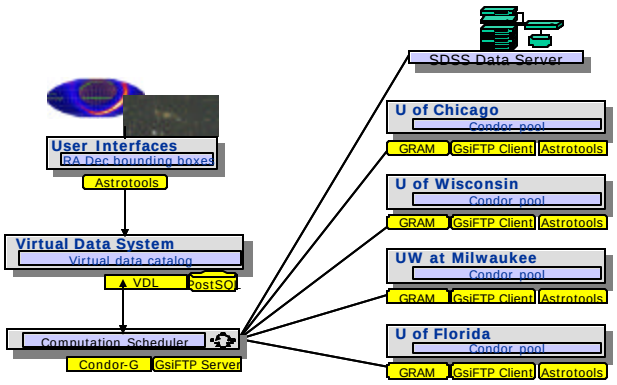

Figure 4: Architecture for integration of Chimera into SDSS environment for cluster finding.

5 Experimentation and Results

Our experimentation with the cluster finding challenge problem to date has involved the analysis of 1350 square degrees of SDSS data (some 42,000 fields) using Chimera and the GriPhyN virtual data toolkit. This represents approximately 20% of the ultimate survey coverage as summarized in Table 2, below:

Metric	# stripes	# fields
Expected coverage of survey (by 2005)	45	210,000
Total data available today	14	50,400
Total data suitable for cluster finding	13	42,000
Total cluster finding workflow generated	13	42,000
Total cluster finding workflow processed	7	14,280

Table 2: Summary of experimental scope

We constructed workflow DAGs using Chimera for all processable data in the current SDSS survey, and executed and validated approximately one third of that workload, as summarized in Table 3, below.

Stripe	# of fields	# DAG nodes	Grid Site	Nodes used	Run time (secs)
9	2,400	305	UW	106	2105
12	2,400	217	UW	120	3547
32	2,400	309	UW	89	2708
34	1,320	123	UW	41	2332
37	3,000	347	UFL	20	2601
76	2,040	235	UW	93	3286
82	720	97	UC	58	3768
Totals	14,280	1633		75 (avg)	2907 (avg)

Table 3: Summary of validated cluster finding results

5.1 Test Grid

Our experimental grid consisted of four large Condor pools, as shown in Figure 5. We used the University of Chicago pool as the master, for job submission and persistent data storage, and we used all the pools for the large amount of computations involved in the cluster finding algorithm.

5.2 Chimera DAG Creation

A "virtual data generator" script was employed to enter the large number of VDL descriptions for each of the five stages of the MaxBCG computation into the Chimera database. (Over 5000 derivation descriptions were created). These scripts read control parameters both from pretuned parameter files and from user input. The resulting Chimera-computed DAGs showed a variety of complex shapes. A DAG for a modest amount of data, 12 fields, is shown in Figure 6, where stage 1, on the bottom, is brgSearch, stage 2 is bcgSearch, stage 3 is bcgCoalesce, and stage4, on the top, is getCatalog.

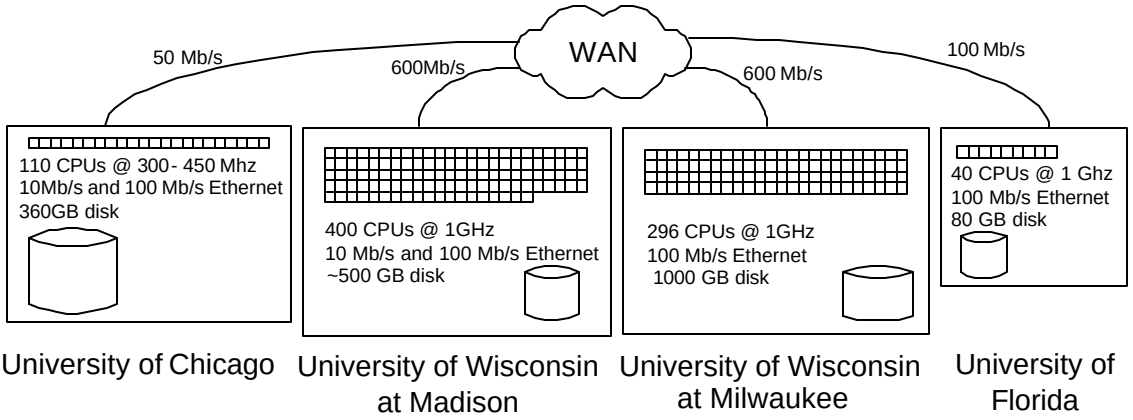

Figure 5: Configurations of Grid resources (Condor pools, storage, and networks) used in our experiments

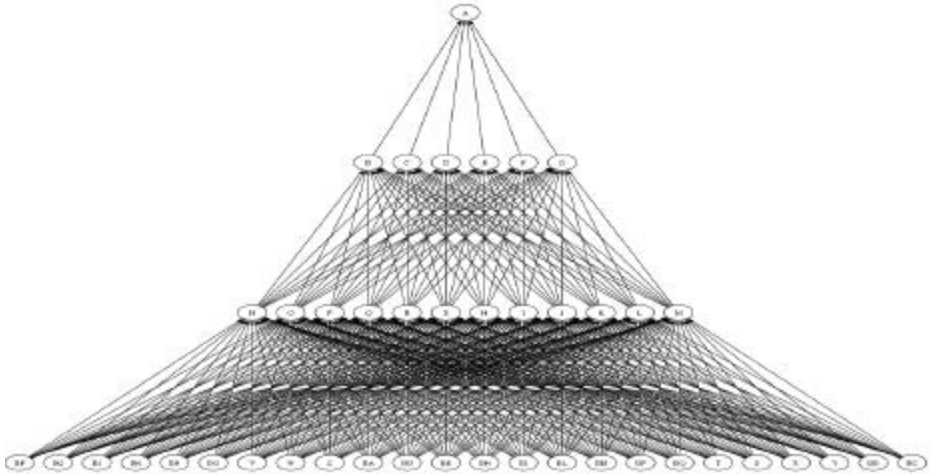

Figure 6: A basic DAG for cluster identification workflow.

The graph shown below is for one specific stripe of the survey (stripe 34). It uses 123 nodes to process 110x12 fields, and illustrates how larger workflows can be composed of many overlapping invocations of the workflow of the basic DAG pattern shown above.

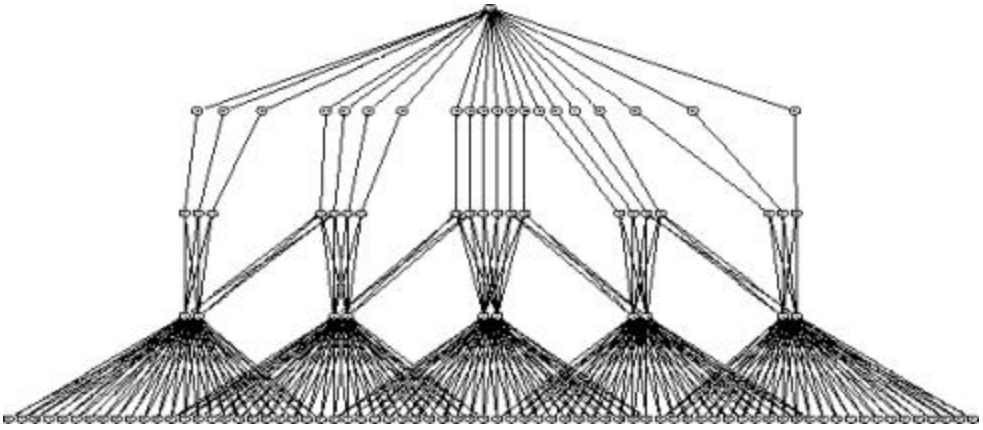

Figure 7: DAG for stripe 34 showing composition from a basic pattern.

Figure 8 and Figure 9 show the execution trace of the basic DAG of Figure 6, both by IP of the host processor of each job step (left) and by job ID (right). The job ID plot clearly shows how each of the four stages of the DAG progressed within the computing Grid. The execution pattern of a larger DAG for 1,200 fields is illustrated in Figure 10 and Figure 11.

5.3 Performance Analysis

We found that the processing time for MaxBCG DAGs depends on how survey fields are grouped in the DAG nodes. (Recall that a DAG node corresponds to a job run on a single machine). For a 600 field area, grouped into 84 fields per brgSearch node, 48 fields per bcgSearch node, and 60 fields per bcgCoalesce node, time to completion was 2402 seconds using 62 hosts and 2480 seconds using 49 hosts. These times include all overheads except initial data transfer. For comparison, the Fermilab machines, essentially optimal for this computation and performing without the Chimera system, take 3360 seconds over 10 hosts for the same area. This illustrates one of the obvious strengths of the Grid: though special purpose machines and stripped down code (i.e., without the problem partitioning encouraged by the virtual data approach) may be much faster on a per-node basis, the larger number of available Grid compute resources enables faster overall problem solution.

5.4 Results of the Computation

The primary result of the computation is the cluster catalog. Figure 12 shows a histogram of the number of galaxies per cluster in the final cluster catalog for survey stripe 10. The power-law relationship is of great interest, because it allows one to constrain various features of the power spectrum of matter in the universe [23, 24]. Equally important for our purpose is the Chimera database containing the transformations that create cluster catalogs given galaxy catalogs, and the derivations that show exactly with which parameters and for what region of the sky each catalog was created.

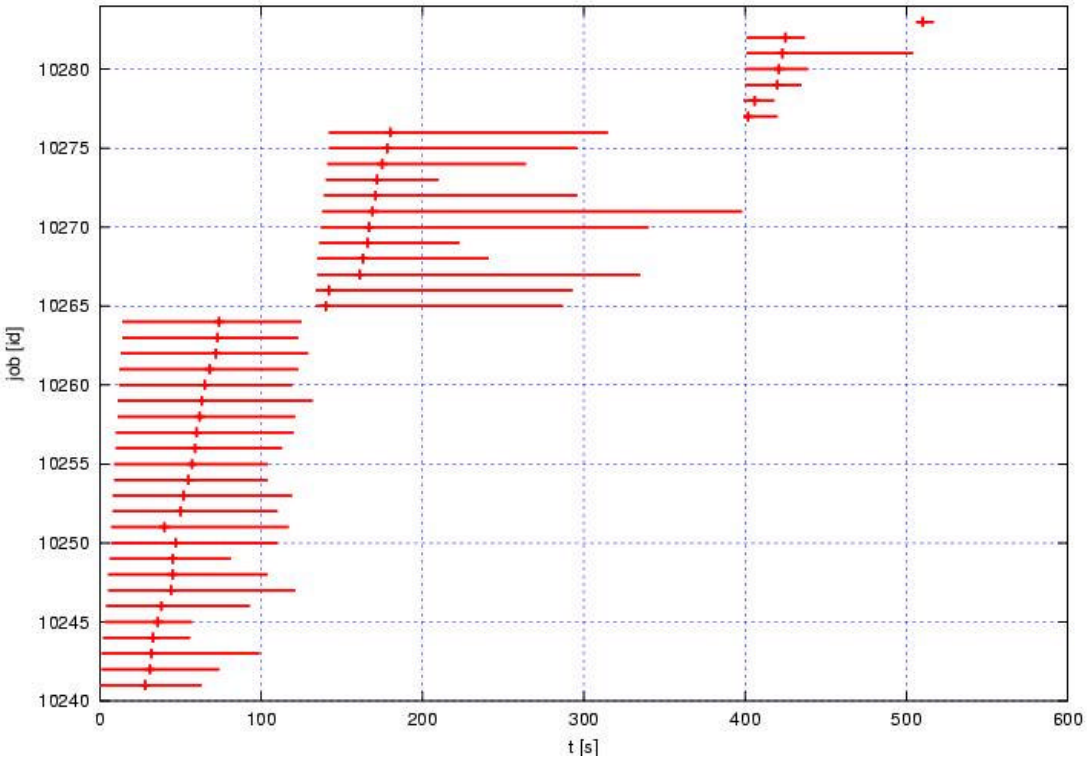

Figure 8 Queuing and execution trace for each stage (by job ID)

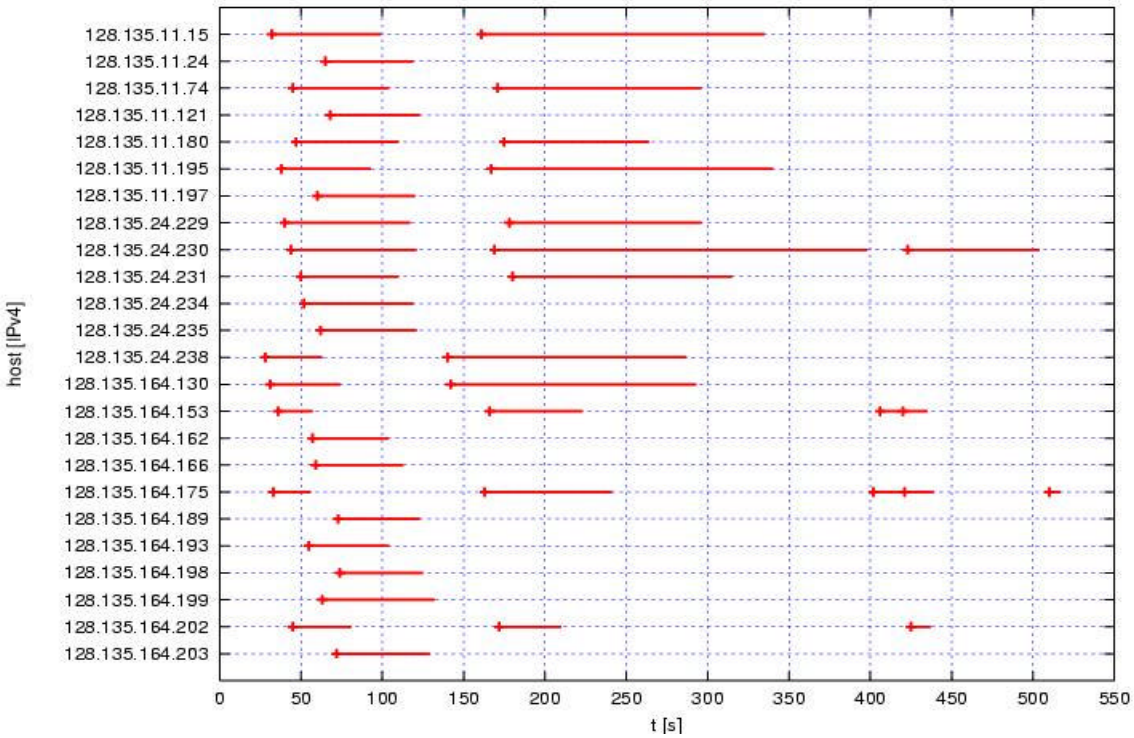

Figure 9: Queuing and execution trace, by compute host

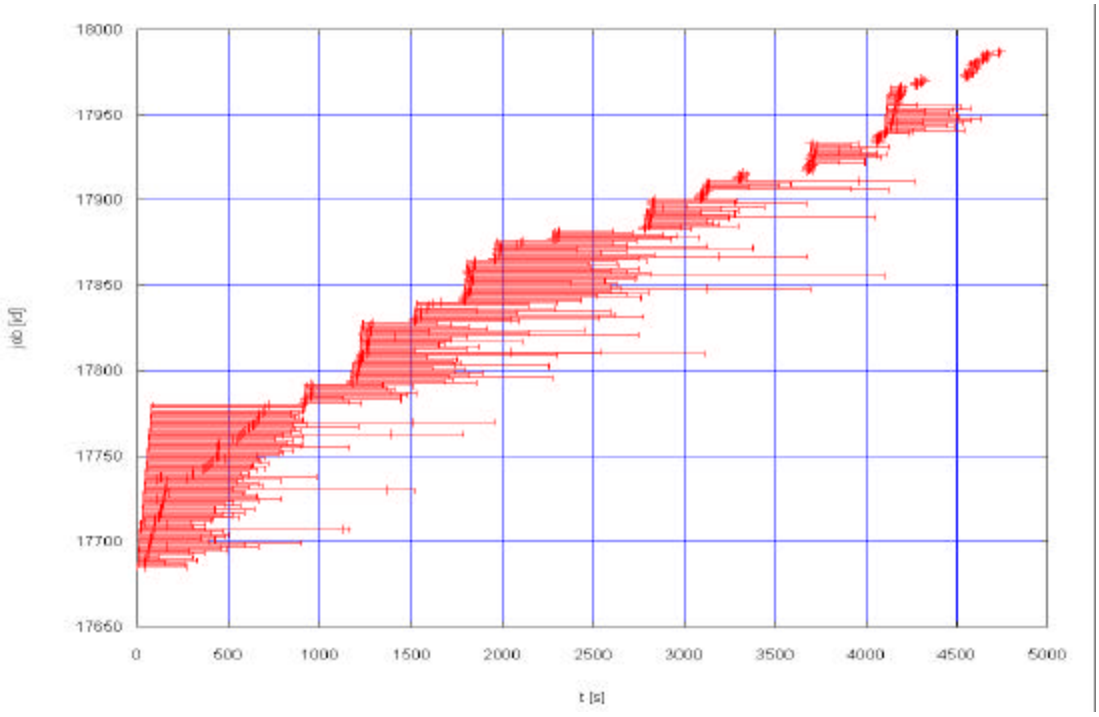

Figure 10 Computation of DAG for 1,200 Fields, by Job ID

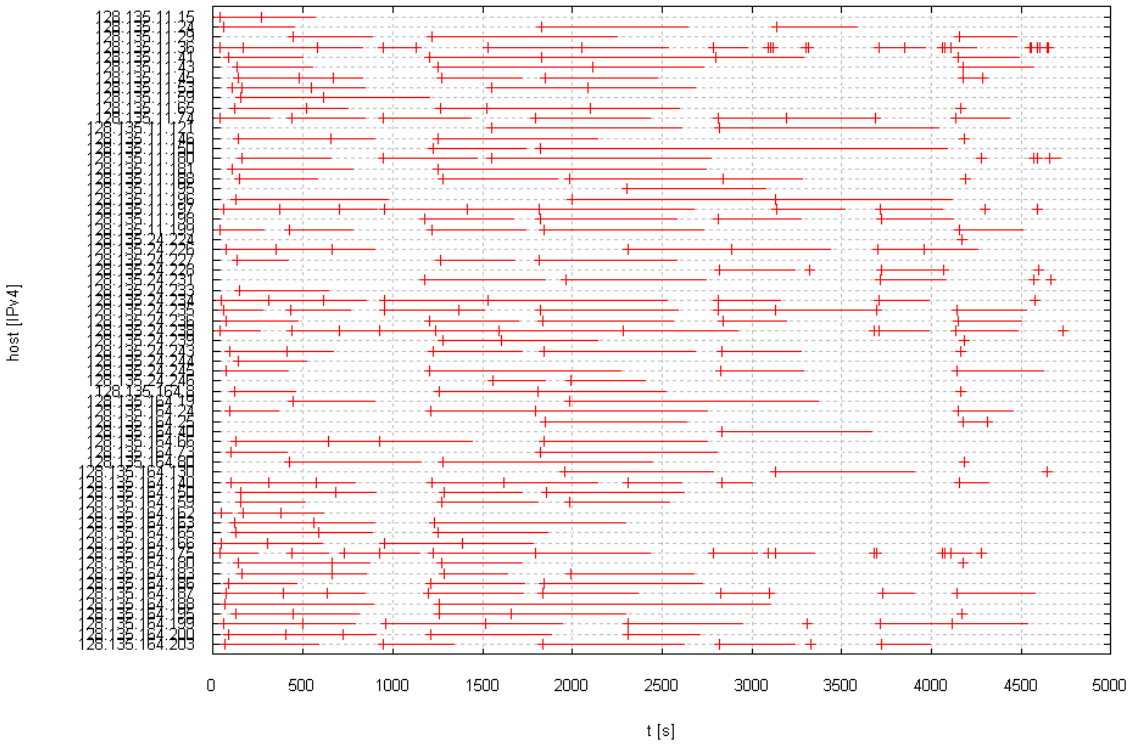

Figure 11: Computation of DAG for 1,200 Fields, by compute host

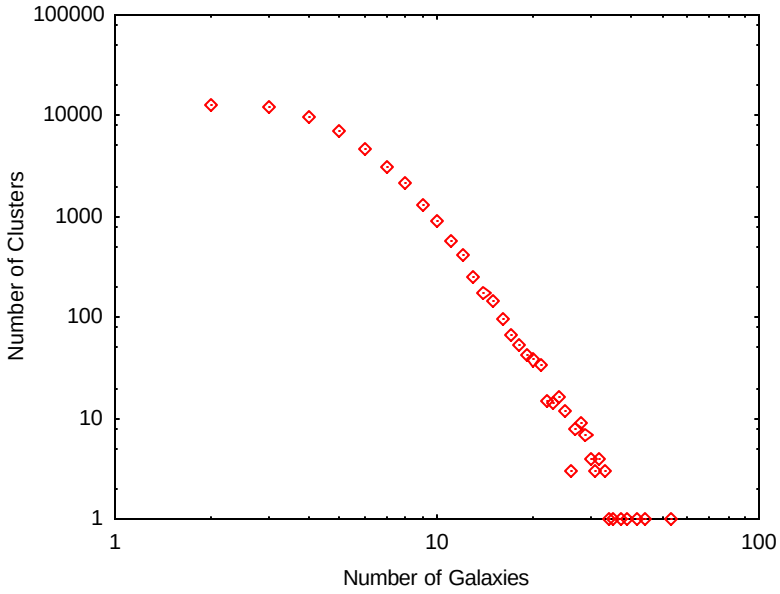

Figure 12: Cluster Distribution

6 Conclusion

We have described a large-scale experiment into the applicability of virtual data concepts to real science problems. This effort has been successful, both convincing us of the merits of the approach and uncovering much ground for future work.

The experiment involved both a "social" test of whether virtual data could be accepted and embraced by scientists, and a technical experiment into the nuances of data representation, request planning, and Grid performance. Our target community of astrophysicists had previously addressed the same challenge problem on a local cluster without virtual data tracking or grid technology. They felt that:

- The complex astrophysics application codes could be readily captured and replayed through the Chimera VDL
- Having the VDL "compile" directly to Grid execution jobs was a powerful and highly productive way to leverage significant compute resources
- The VDL was relatively easy to use, essentially proving a highly structured manner in which to create job execution scripts that gave the enormous benefit of input, output, and executable tracking—something previously done in SDSS (as is typical in science today) by hand, with logbooks.

The process of "wrapping" complex parameter files as a virtual data derivation has proven to be a productive and effective structuring paradigm for storing and tracking these complex control parameters. Moreover we have prototyped a solution to the difficult problem of dynamic determination of filenames; we feel that the approach taken here will be useful in similar cases but that a more dynamic "trace-based" approach will be necessary for cases where the file determination algorithm is not so readily extracted or executed a-priori, in isolation.

The paradigm of "creating" virtual data en masse through generating scripts proved effective, in essence mapping out a large experimental data space that can then be populated by both interactive and batch-oriented processes. Future work will explore the use of automated triggers to produce data as experiments generate new data.

The possibility of regenerating data when code or dependent data objects change, which is enabled by the Chimera data dependence tracking model, is seen as a huge potential benefit—as significant as the application of "makefiles" to the process of compiling executable applications.

In future work we will expand the size of our Grid and complete analysis of the currently available SDSS data. We plan to also address further astrophysics problems that explore other parts of the SDSS computation space, including the search for near-earth asteroids, a problem that demands analysis of the images themselves and hence is dominated by data transfer, and computation of the angular power spectrum of clusters, which demands large matrix inversions and hence calls for MPI-enabled Grids. We have outlined other extensions of the virtual data paradigm elsewhere [5].

In summary, this effort was an important first step in what we expect to be a lengthy engagement with many scientific disciplines. We expect that each new application, and certainly each new scientific domain, will bring new challenges to the application of virtual data. As we endeavor to expand on these results, we are eager to determine how far we can go toward improving the productivity of other data-intensive areas of scientific practice.

Acknowledgments

We gratefully acknowledge many helpful and stimulating discussions on these topics with our colleagues Peter Buneman, Ewa Deelman, Carl Kesselman, Miron Livny, Gaurang Mehta, Alain Roy, Alex Szalay, Karan Vahi, and many other members of GriPhyN and the other Grid projects. The latest version of the DAG planning code was developed at the University of Southern California Information Sciences Institute by Deelman, Mehta, and Vahi.

We especially thank Catalin Dumitrescu and Michael Milligan of the University of Chicago and Alain Roy of the University of Wisconsin Condor Project for their invaluable assistance in the experimental stages of this work.

This research was supported in part by the National Science Foundation under contract ITR-0086044 (GriPhyN) and by the Mathematical, Information, and Computational Sciences Division subprogram of the Office of Advanced Scientific Computing Research, U.S. Department of Energy, under Contract W-31-109-Eng-38 (SciDAC Data Grid Middleware).

References

[1] P. Avery and I. Foster, "The GriPhyN Project: Towards Petascale Virtual Data Grids," Technical Report GriPhyN-2001-15, 2001.

[2] "The DataGrid Architecture," EU DataGrid Project DataGrid-12-D12.4-333671-3-0, 2001.

[3] I. Foster, E. Alpert, A. Chervenak, B. Drach, C. Kesselman, V. Nefedova, D. Middleton, A. Shoshani, A. Sim, and D. Williams, "The Earth System Grid II: Turning Climate Datasets Into Community Resources," presented at Annual Meeting of the American Meteorological Society, 2002.

[4] "Particle Physics Data Grid Project (PPDG), www.ppdg.net."

[5] I. Foster, J. Voeckler, M. Wilde, and Y. Zhao, "Chimera: A Virtual Data System for Representing, Querying, and Automating Data Derivation," presented at 14th Conference on Scientific and Statistical Database Management, 2002.

[6] A. Szalay and J. Gray, "The World-Wide Telescope," *Science*, vol. 293, pp. 2037-2040, 2001.

[7] A. S. Szalay, P. Z. Kunszt, A. Thakar, J. Gray, D. Slutz, and R. J. Brunner, "Designing and Mining Multi-Terabyte Astronomy Archives: The Sloan Digital Sky Survey," *SIGMOD Record*, vol. 29, pp. 451-462, 2000.

[8] A. Chervenak, I. Foster, C. Kesselman, C. Salisbury, and S. Tuecke, "The Data Grid: Towards an Architecture for the Distributed Management and Analysis of Large Scientific Data Sets," *J. Network and Computer Applications*, pp. 187-200, 2001.

[9] I. Foster and C. Kesselman, "The Grid: Blueprint for a New Computing Infrastructure," Morgan Kaufmann, 1999.

[10] E. Deelman, C. Kesselman, and G. Mehta, "Transformation Catalog Design for GriPhyN," Technical Report GriPhyN-2001-17, 2001.

[11] E. Deelman, K. Blackburn, P. Ehrens, C. Kesselman, S. Koranda, A. Lazzarini, G. Mehta, L. Meshkat, L. Pearlman, K. Blackburn, and R. Williams., "GriPhyN and LIGO: Building a Virtual Data Grid for Gravitational Wave Scientists," presented at 11th Intl Symposium on High Performance Distributed Computing, 2002.

[12] E. Deelman, I. Foster, C. Kesselman, and M. Livny, "Representing Virtual Data: A Catalog Architecture for Location and Materialization Transparency," Technical Report GriPhyN-2001-14, 2001.

[13] P. Buneman, S. Khanna, and W.-C. Tan, "Why and Where: A Characterization of Data Provenance," presented at International Conference on Database Theory, 2001.

[14] Y. Cui, J. Widom, and J. L. Wiener, "Tracing the Lineage of View Data in a Warehousing Environment," *ACM Transactions on Database Systems*, vol. 25, pp. 179–227, 2000.

[15] Y. E. Ioannidis, M. Livny, S. Gupta, and N. Ponnekanti, "ZOO : A Desktop Experiment Management Environment," presented at 22th International Conference on Very Large Data Bases, 1996.

[16] A. Woodruff and M. Stonebraker, "Supporting Fine-Grained Data Lineage in a Database Visualization Environment," Computer Science Division, University of California Berkeley, Report UCB/CSD-97-932, 1997.

[17] Y. E. Ioannidis and M. Livny, "Conceptual Schemas: Multi-faceted Tools for Desktop Scientific Experiment Management," *International Journal of Cooperative Information Systems*, vol. 1, pp. 451-474, 1992.

[18] I. Foster and C. Kesselman, "Globus: A Toolkit-Based Grid Architecture," in *The Grid: Blueprint for a New Computing Infrastructure*, I. Foster and C. Kesselman, Eds.: Morgan Kaufmann, 1999, pp. 259-278.

[19] M. Litzkow, M. Livny, and M. Mutka, "Condor - A Hunter of Idle Workstations," in *Proc. 8th Intl Conf. on Distributed Computing Systems*, 1988, pp. 104-111.

[20] J. Frey, T. Tannenbaum, I. Foster, M. Livny, and S. Tuecke, "Condor-G: A Computation Management Agent for Multi-Institutional Grids," presented at 10th International Symposium on High Performance Distributed Computing, 2001.

[21] W. Hoschek, J. Jaen-Martinez, A. Samar, H. Stockinger, and K. Stockinger, "Data Management in an International Data Grid Project," presented at International Workshop on Grid Computing, 2000.

[22] H. Stockinger, A. Samar, W. Allcock, I. Foster, K. Holtman, and B. Tierney, "File and Object Replication in Data Grids," presented at 10th IEEE Intl. Symp. on High Performance Distributed Computing, 2001.

[23] J. Einasto, M. Einasto, E. Tago, A. A. Starobinsky, F. Atrio-Barandela, V. Müller, A. Knebe, P. Frisch, R. Cen, H. Andernach, and D. Tucker, "Steps Toward the Power Spectrum of Matter. I. The Mean Spectrum of Galaxies'," *Astrophysical Journal*, vol. 519, pp. 441-455, 1999.

[24] M. Gramann and I. Suhhonenko, "The Power Spectrum of Clusters of Galaxies and the Press-Schechter Approximation'," *Astrophysical Journal*, vol. 519, pp. 433, 1999.

[25] Annis, J., Kent, S., Castender, F., Eisenstein, D., Gunn, J., Kim, R., Lupton, R., Nichol, R., Postman, M., and Voges, W., "The MaxBCG Technique for Finding Galaxy Clusters in SDSS Data", In *AAS 195th Meeting* (2000).