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, IL, 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

1 Introduction

The GriPhyN project [4] is one of several major efforts [1, 2, 12] 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 [15]. 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) [23, 24], 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 (some 10 TB) of images and a smaller set of catalogs (some 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. 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 large amounts of computation and thus can, in principle, benefit from the use of distributed computing and storage resources, as provided by Data Grids [6, 14].

We describe here an early exploration of virtual data and Data Grid concepts in an application 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 [15], 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 the virtual data catalog?
- 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 DAGs for efficient grid execution?
- When code or data changes can we identify dependent re-derivations?
- 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 [8-10]. Also related is work on data lineage in database systems [5, 7, 19, 20, 25]. 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 toolkit (VDT V1.0) includes the Globus Toolkit [13], Condor and Condor-G [16, 21], and the Grid Data Mirroring Package [18, 22].

We apply here a new tool to be included in VDT: the Chimera virtual data system [15]. 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, plus 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).

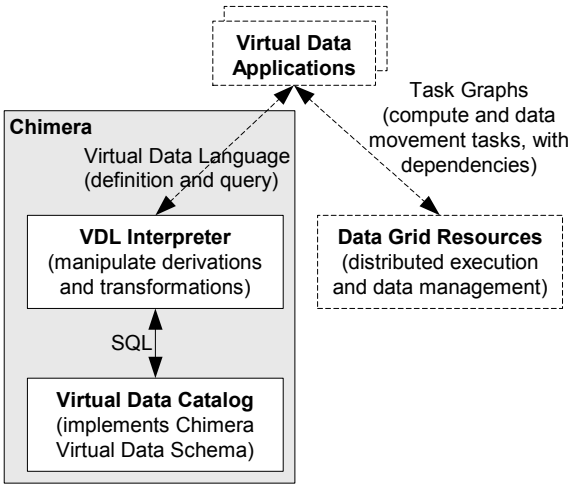

Figure 1: Schematic of the Chimera architecture

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, "re-materialize" 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, etc., 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 the central 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 LFN(s), output LFN(s), transformation name, and/or application name.

Given a logical file name, the Chimera "request planner" can generate a directed acyclic graph (DAG) for use by DAGman [16] to manage the computation. 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, un-managed 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) [3]. 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 [3] and briefly summarized here. For each galaxy, one calculates whether it is likely to be a luminous red galaxy (BRG). If this likelihood is above a threshold, compute a likelihood that it is a brightest cluster galaxy by weighting the BRG likelihood by the number of galaxies in the neighborhood. Finally, ask whether this galaxy is the most likely in the area; if so there is a cluster present centered here, else 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. One then works on a central region, with a buffer zone around that region of angular size of the upper limit. One only locates clusters in the central region, but uses the information from the buffer zone for the calculation.

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

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

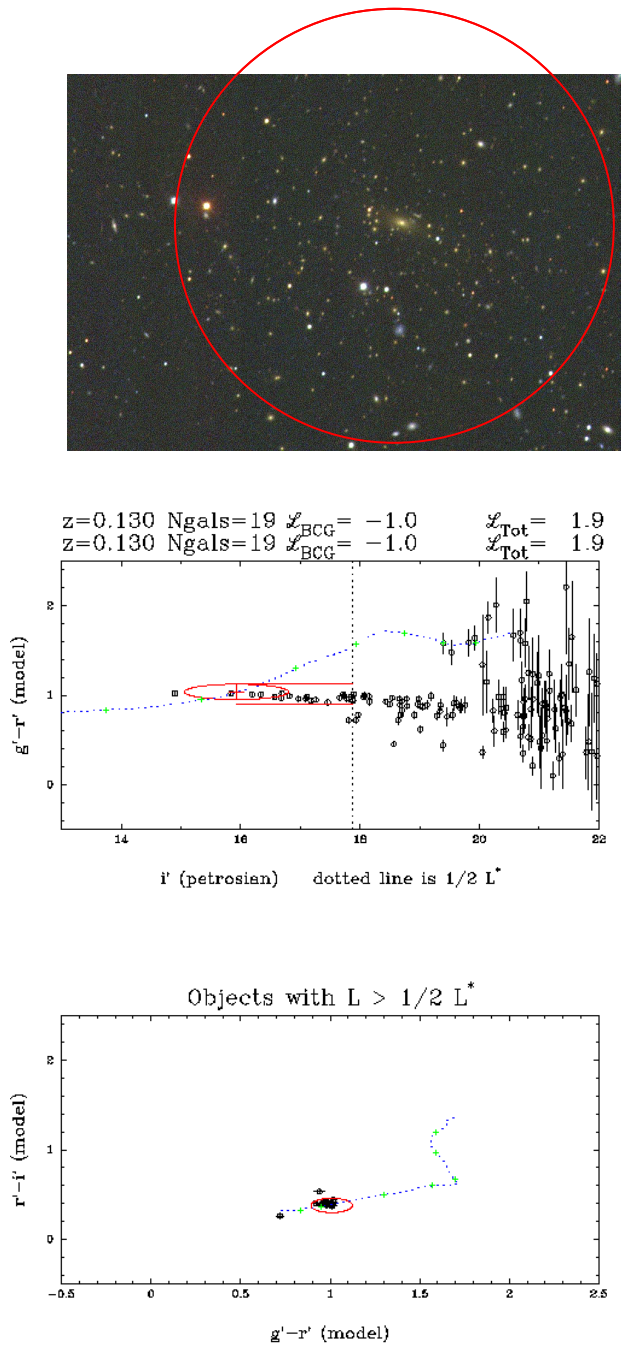

Figure 2: The MaxBCG cluster finding algorithm

4 The GriPhyN Chimera Formulation of Cluster Finding

We describe here our representation of the maxBCG transformations, and then our system design of 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:

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.

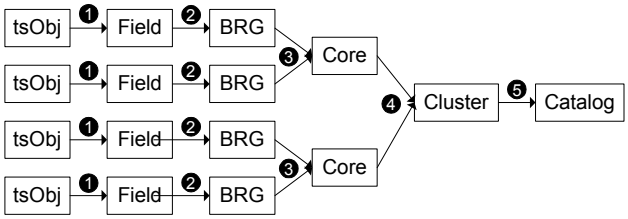

Figure 3: SDSS cluster identification workflow.

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 [15].

```
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 pegase-kcorr.par
      File   I field-%run-%camcol -%field.par
      File   o brg-%run-%camcol-%field.par
End

rc     parameters.par      $root/parameters.par
rc     pegase-kcorr.par    $root/pegase-kcorr.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
```

Because the maxBCG algorithm walks through a buffer 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 a maxBcg module that maps a (RA, Dec) pair to a list of filenames, as illustrated below. The runs are 752 and 756, with field numbers starting at 30 for runs 752 and at 215 for run 756. The field being analyzed is run 752 field 32.

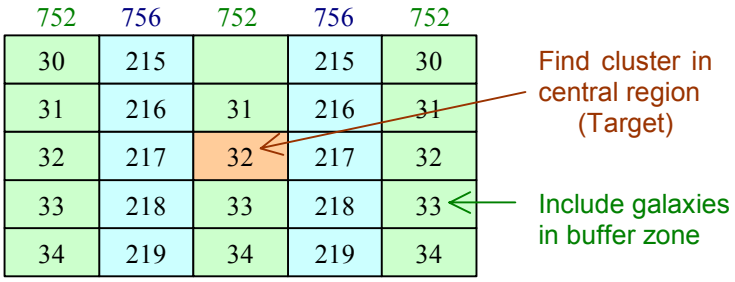

Figure 4: Dynamic buffer region around target galaxy.

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 5.

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 the input files containing critical input parameters. Instead of keeping all possible parameter files in the Chimera database, we built transformations in Chimera that take the primary changeable parameters as arguments and incorporate those parameters into *parameter file templates* which contain numerous seldom-changed parameters, and generate complete parameter files as output. These transformations are then incorporated into the user application DAG to generate the parameter file used by the code. We only need to maintain a few templates, which can be instantiated unlimited times. With this technique, the Chimera database contains (in its derivation records) all parameters required to rerun the derivation.

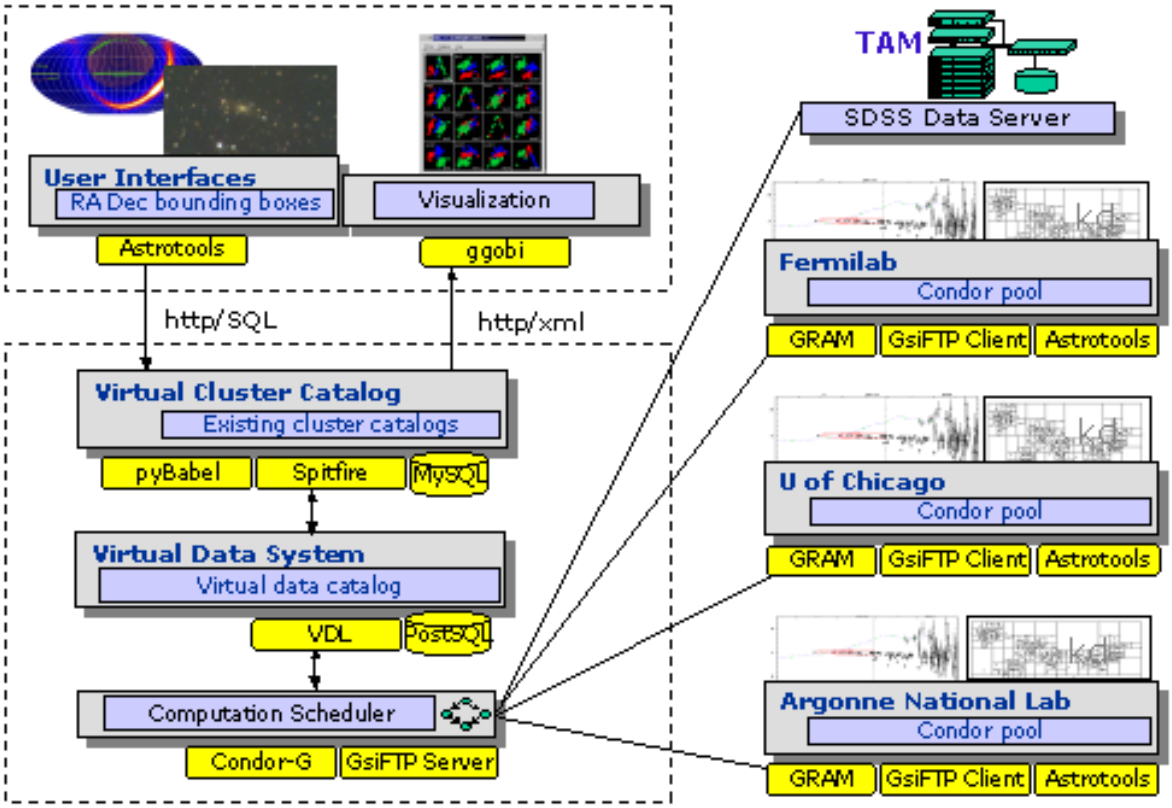

Figure 5: Architecture for integration of Chimera into SDSS environment for cluster-finding.

5 Experimentation and Results

Our experimentation with this process to date has involved the analysis of 220 square-degrees of SDSS data (some 7000 fields, representing 3% of the ultimate survey), using Chimera and the virtual data toolkit.

5.1 Test Grid

Our experimental grid consisted of three condor pools, as shown in Figure 6. We used the Argonne pool as the master, for job submission and persistent data storage. The Chicago pool is used for computation, and the Fermilab pool is used for comparison purposes.

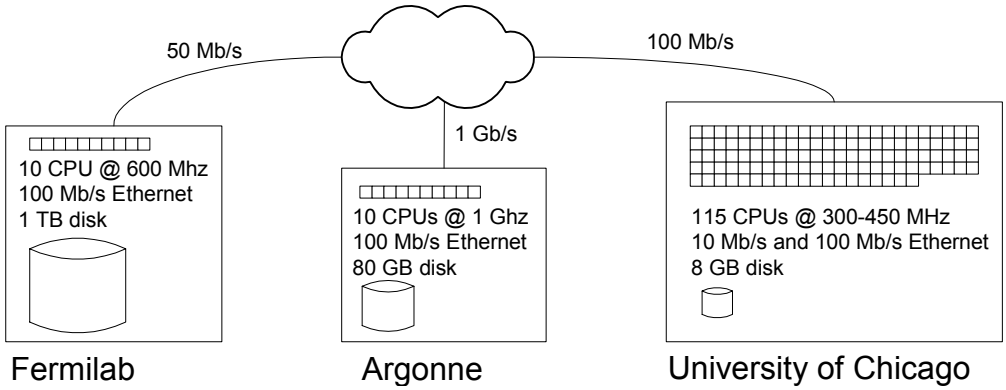

Figure 6: Configurations of the resources (Condor pools, storage, and networks) used in our experiments

5.2 Chimera DAG Creation

A "virtual data generator" script was employed to enter the large number (thousands) of VDL descriptions for each of the five stages of the maxBCG computation into the Chimera database. These scripts read control parameters both from pre-tuned parameter files and from user input. The resulting Chimera-computed DAGs showed a variety of complex shapes. A DAG for a trivial amount of data, 48 fields, is shown in Figure 7, where stage 1, on the bottom, is brgSearch, stage 2 is bcgSearch, stage 3 is bcgCoalesce, and stage4, on the top, is getCatalog.

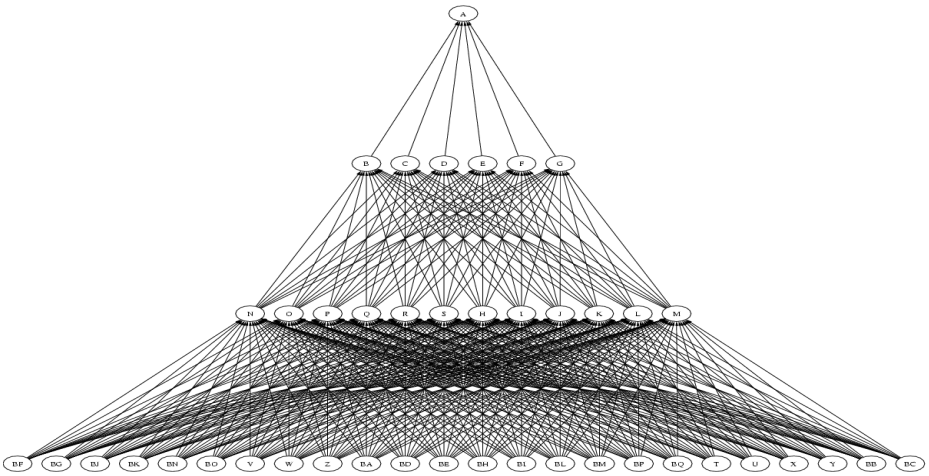

Figure 7: DAG for cluster identification workflow.

Figures 8 and 9 shows the execution trace of this DAG, 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. A larger DAG with approximately 100 nodes is illustrated in Figure 10 and 11.

5.3 Performance Analysis

We found that the processing time for MaxBCG DAGs depends on how one groups fields inside 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, takes 3360 seconds over 10 hosts for the same area. This illustrates an interesting point about the Grid: though special purpose machines and stripped down code may be much faster on a node by node basis, the sheer number of Grid compute resources enables faster overall performance.

Interestingly, the time to completion seems to be minimized by maximizing the number of nodes, within limits. This effect is due in part to increasing the number of hosts, but is also due to an effect that shows up when one looks at the distribution of times to completion of the individual nodes of a stage. These are exponential: 50% of the jobs finish in a time T, the next 40% of the jobs finish in another T, and the remaining 10% of the jobs finish in a third increment T. The overall time to completion is therefore dominated by the last 10% of the jobs that take 3T to finish, and one wins by minimizing T, which one does by maximizing the number of nodes. This effect seems not related to the intrinsic speed of the host (surprisingly), but rather to the complexity of the DAG as the simple brgSearch transformation does not show the effect, but both the bcgSearch and bcgCoalesce transformations with their complicated DAGs, do. Further details of this analysis will be presented in the final paper.

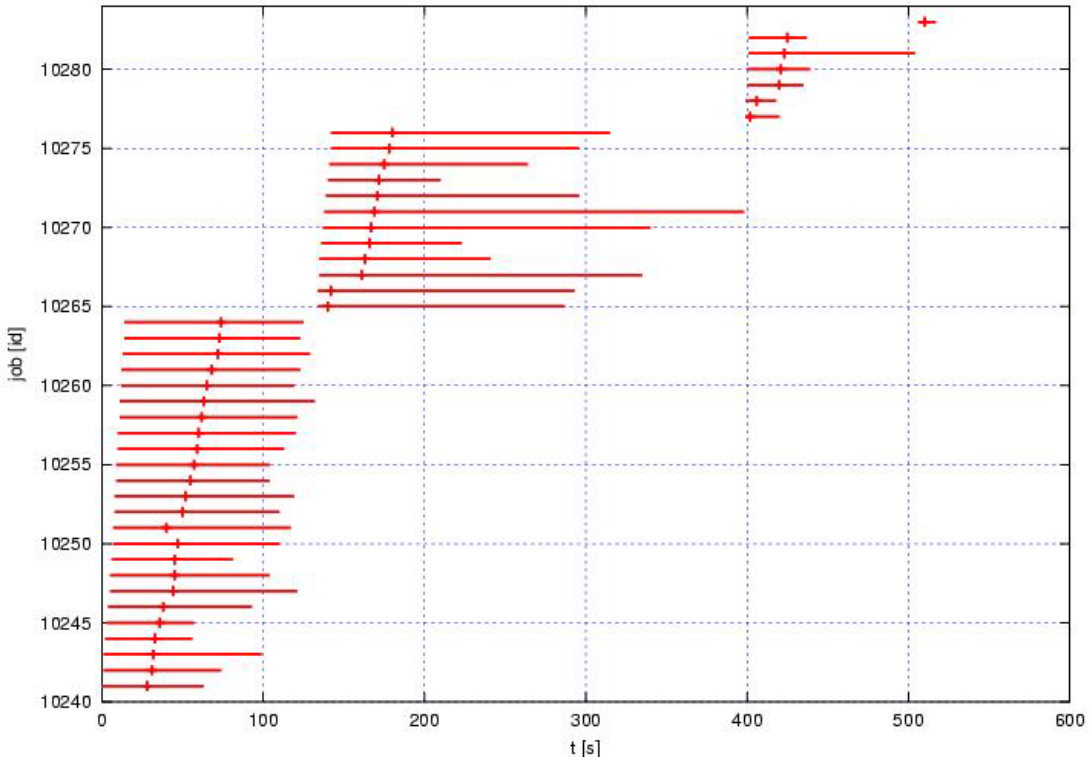

Figure 8 Queuing and execution trace for each stage (by job ID)

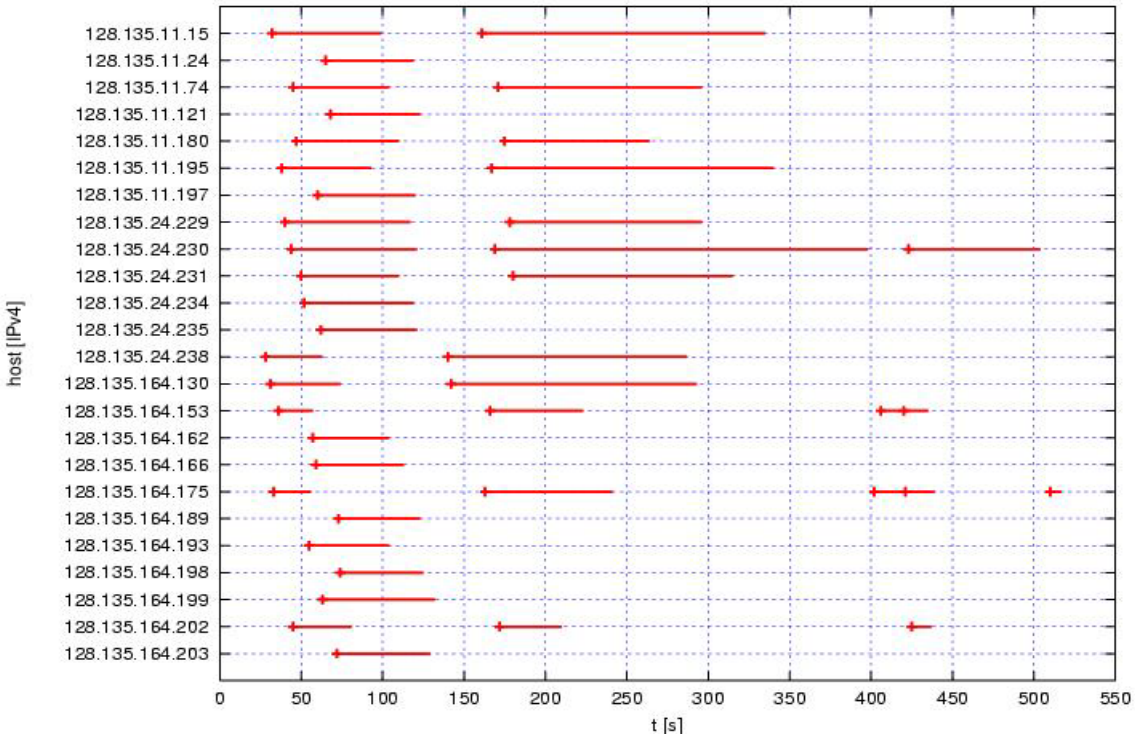

Figure 9: Queuing and execution trace, by compute host

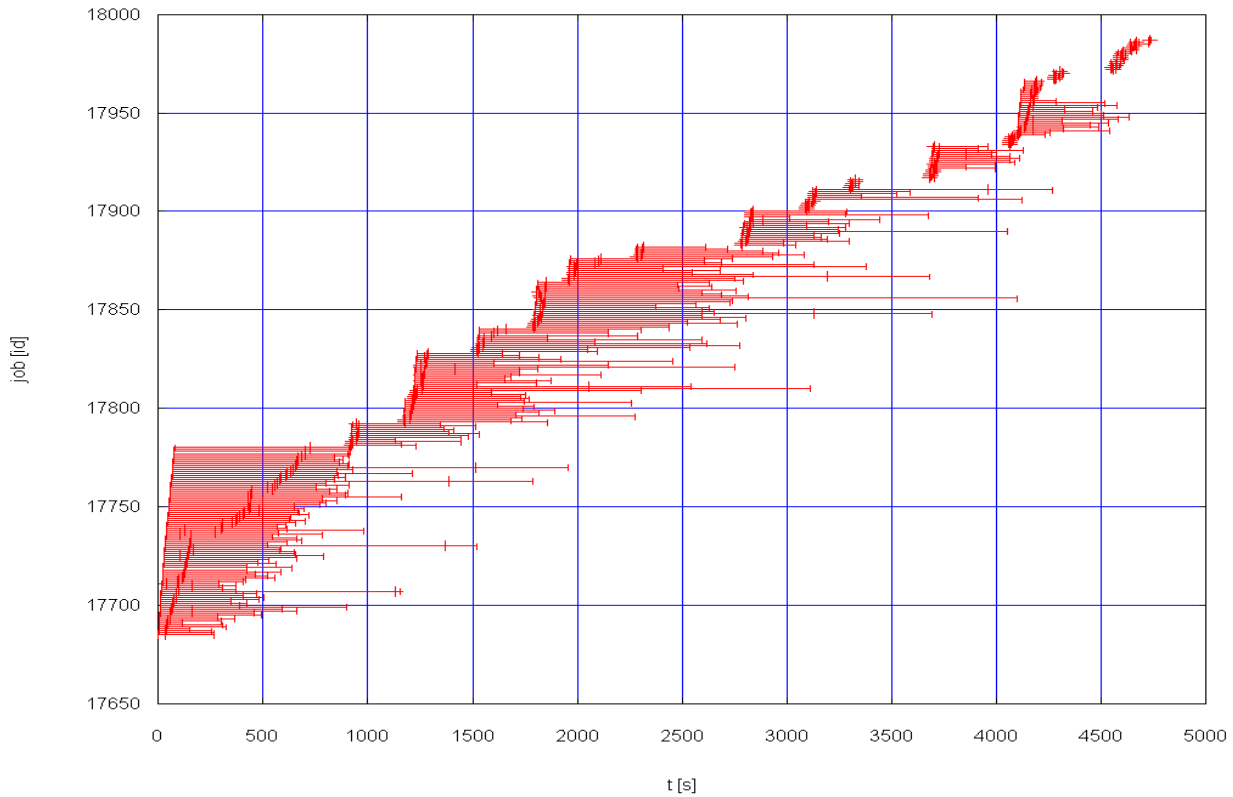

Figure 10 Computation of DAG for 100 Fields, by Job ID

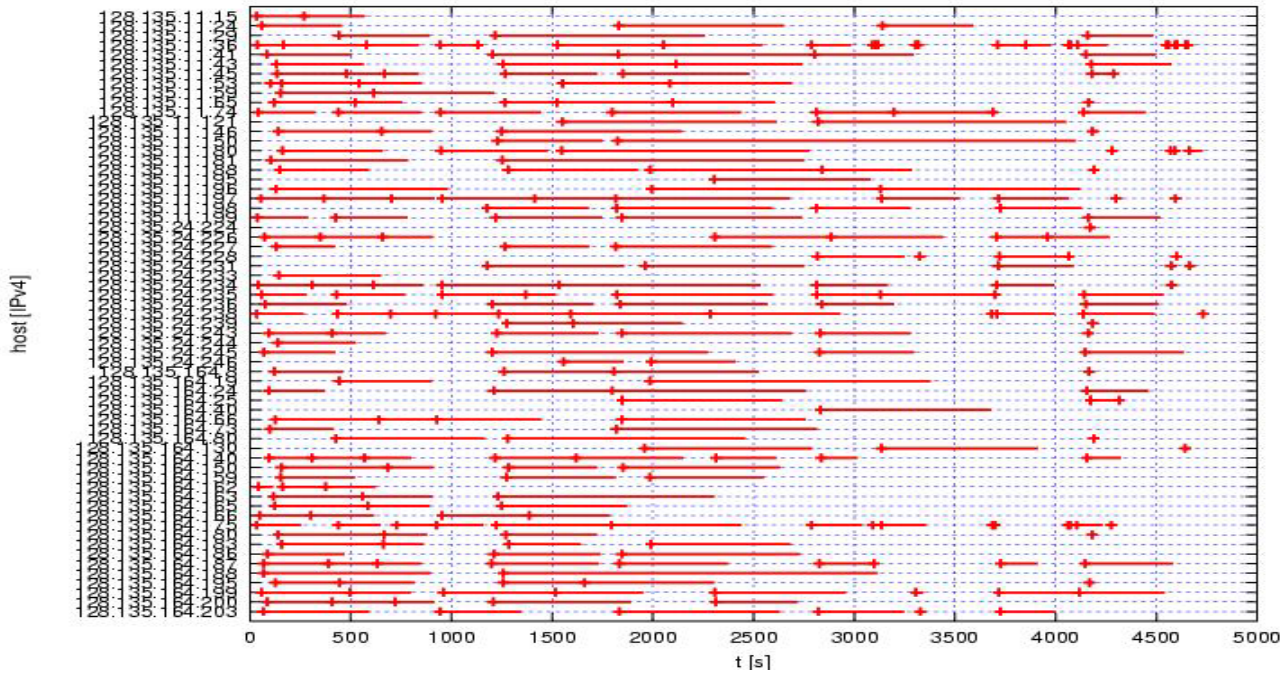

Figure 11: Computation of DAG for 100 Fields, by compute host

5.4 The 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. The power-law relationship is of great interest, as it allows one to constrain various features of the power spectrum of matter in the universe [11, 17]. Equally important for our purpose are the Chimera database containing the transformations that create cluster catalogs given galaxy catalogs, and the derivations that show exactly with which parameters and over what area this particular catalog was created.

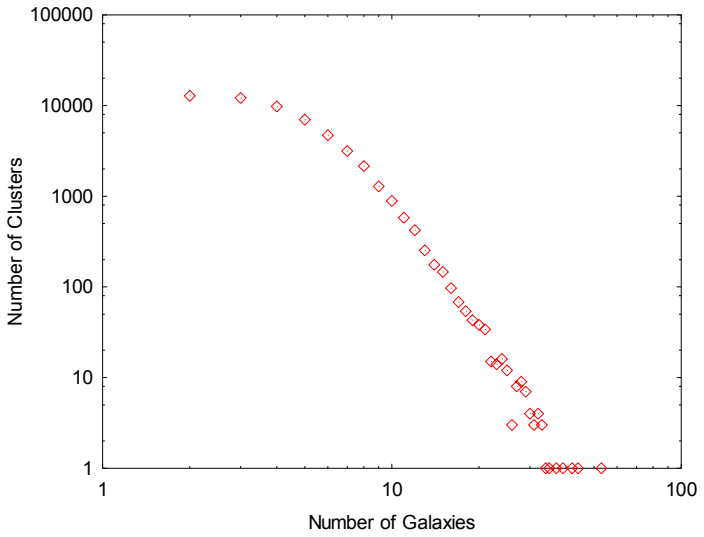

Figure 12 Cluster Distribution

6 Conclusion

We have described a large-scale experiment into the applicability of virtual data concepts to real science problems. This effort has been a great success, 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;
- The benefits of 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 ease with which data could be regenerated when code or dependent data objects change is seen as a huge benefit—as significant as the application of "makefiles" to the process of compiling executable

applications. 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. And a solution has been prototyped 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-mass through generating scripts proved effective, in essence mapping out a large experimental data space that can then be populated via both interactive and batch-oriented processes. Future work will explore the use of automated triggers to produce data as experiments generate new data.

Our studies of the effect of DAG shape on run time performance are preliminary, but suggest a fruitful area of Grid research, in which automated request-planning optimizations can be explored. We plan more thorough results in the final paper.

In future work (and in time for a final SC submission), 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 demand analysis of the images themselves and hence is dominated by data-transfer, andcomputation 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 [15].

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 towards improving the productivity of data-intensive science in the 21st century.

Acknowledgments

We gratefully acknowledge helpful discussions with many colleagues. 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 (Data Grid Toolkit).

References

1. The DataGrid Architecture. EU DataGrid ProjectDataGrid-12-D12.4-333671-3-0, 2001, www.eu-datagrid.org.
2. Particle Physics Data Grid Project (PPDG), www.ppdg.net.
3. Annis, J., Kent, S., Castander, 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)
4. Avery, P. and Foster, I. The GriPhyN Project: Towards Petascale Virtual Data Grids. Technical Report GriPhyN-2001-15, 2001, www.griphyn.org.
5. Buneman, P., Khanna, S. and Tan, W.-C., Why and Where: A Characterization of Data Provenance. In *International Conference on Database Theory*, (2001)
6. Chervenak, A., Foster, I., Kesselman, C., Salisbury, C. and Tuecke, S. The Data Grid: Towards an Architecture for the Distributed Management and Analysis of Large Scientific Data Sets. *J. Network and Computer Applications* (23). 187-200. 2001.

7. Cui, Y., Widom, J. and Wiener, J.L. Tracing the Lineage of View Data in a Warehousing Environment. *ACM Transactions on Database Systems*, *25* (2). 179–227. 2000.

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

9. Deelman, E., Foster, I., Kesselman, C. and Livny, M. Representing Virtual Data: A Catalog Architecture for Location and Materialization Transparency. Technical Report GriPhyN-2001-14, 2001, www.griphyn.org.

10. Deelman, E., Kesselman, C. and Mehta, G. Transformation Catalog Design for GriPhyN. Technical Report GriPhyN-2001-17, 2001, www.griphyn.org.

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

12. Foster, I., Alpert, E., Chervenak, A., Drach, B., Kesselman, C., Nefedova, V., Middleton, D., Shoshani, A., Sim, A. and Williams, D., The Earth System Grid II: Turning Climate Datasets Into Community Resources. In *Annual Meeting of the American Meteorological Society*, (2002)

13. Foster, I. and Kesselman, C. Globus: A Toolkit-Based Grid Architecture. In Foster, I. and Kesselman, C. eds. *The Grid: Blueprint for a New Computing Infrastructure*, Morgan Kaufmann, 1999, 259-278.

14. Foster, I. and Kesselman, C. (eds.). *The Grid: Blueprint for a New Computing Infrastructure*. Morgan Kaufmann, 1999.

15. Foster, I., Voeckler, J., Wilde, M. and Zhao, Y., Chimera: A Virtual Data System for Representing, Querying, and Automating Data Derivation. In *14th Conference on Scientific and Statistical Database Management*, (2002). www.griphyn.org

16. Frey, J., Tannenbaum, T., Foster, I., Livny, M. and Tuecke, S., Condor-G: A Computation Management Agent for Multi-Institutional Grids. In *10th International Symposium on High Performance Distributed Computing*, (2001), IEEE Press, 55-66

17. Gramann, M. and Suhhonenko, I. The Power Spectrum of Clusters of Galaxies and the Press-Schechter Approximation'. *Astrophysical Journal*, *519*. 433. 1999.

18. Hoschek, W., Jaen-Martinez, J., Samar, A., Stockinger, H. and Stockinger, K., Data Management in an International Data Grid Project. In *International Workshop on Grid Computing*, (2000), Springer Verlag Press

19. Ioannidis, Y.E. and Livny, M. Conceptual Schemas: Multi-faceted Tools for Desktop Scientific Experiment Management. *International Journal of Cooperative Information Systems*, *1* (3). 451-474. 1992.

20. Ioannidis, Y.E., Livny, M., Gupta, S. and Ponnekanti, N., ZOO : A Desktop Experiment Management Environment. In *22th International Conference on Very Large Data Bases*, (1996), Morgan Kaufmann, 274-285

21. Litzkow, M., Livny, M. and Mutka, M. Condor - A Hunter of Idle Workstations. In *Proc. 8th Intl Conf. on Distributed Computing Systems*, 1988, 104-111.

22. Stockinger, H., Samar, A., Allcock, W., Foster, I., Holtman, K. and Tierney, B., File and Object Replication in Data Grids. In *10th IEEE Intl. Symp. on High Performance Distributed Computing*, (2001), IEEE Press, 76-86

23. Szalay, A. and Gray, J. The World-Wide Telescope. *Science*, *293*. 2037-2040. 2001.

24. Szalay, A.S., Kunszt, P.Z., Thakar, A., Gray, J., Slutz, D. and Brunner, R.J. Designing and Mining Multi-Terabyte Astronomy Archives: The Sloan Digital Sky Survey. *SIGMOD Record*, *29* (2). 451-462. 2000.

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