Grid Spider: A Framework for Data Intensive Research with Data Process Memoization Cache

Daichi Yamada, Tomohiro Sonobe, Hiroshi Tezuka and Mary Inaba
Graduate School of Information Science and Technology
The University of Tokyo
Tokyo, Japan
{*yamada.daichi, sonobe.tomohiro, tezuka.hiroshi, mary*}*@ci.i.u-tokyo.ac.jp*

***Abstract*—As computational power grow, the new field of "Data Intensive Computation" has emerged in which vast amounts of data generated by radio telescopes, particle accelerators, electron microscopes, genomics and Earth observation equipment is processed. In most cases, once the data has been accumulated, it is not overwritten. It has also been observed that in many cases the very same software is used to preprocess the very same data, leading to identical results. To address these issues, we propose "Grid Spider", a framework for data intensive scientific research which is optimized to avoid re-computation through the utilization of our file cache mechanism called "Data Process Memoization Cache" or DPMCache. This mechanism requires pre-processing applications to maintain referential transparency. Both the data and the application are registered with Grid Spider prior to processing, and for each execution of the application, Grid Spider records the history of the coupling of the application, the input data fie, and the output data file. To evaluate Grid Spider, we have implemented "GEO Grid Spider II", which is a framework within which geo-scientists can evaluate satellite data archives.**

***Keywords*-data intensive; memoization; file cache; cache replacement;**

I. INTRODUCTION

Computers have been playing an important role in the progress of scientific research from their inception. As computational power has risen, supercomputers have been increasingly utilized to provided the number-crunching necessary for scientific simulations. More recently with the rapid development of storage devices and distributed computing, "Data Intensive Computation" has emerged which analyzes huge volumes of of scientific observation data.

Radio telescopes, particle accelerators, electron microscopes, genomics and Earth observation equipment are typical sources for such data requiring intensive computation [1]. In most cases, the observation equipment tends to be quite expensive, and many researchers and research groups often share the data.

One common characteristic of this sort of data is that it is never overwritten. Once written, it is merely read by the interested researchers. In addition, it has been observed that, in many cases, the very same software such as code libraries for pre-processing is applied. Assuming that the very same software is processing the very same data, is reasonable to conclude that the results will also be the same. If a method of avoiding this kind of re-computation were possible, it would improve the resolution of the results.

To this end, we propose "Grid Spider", a framework tailored to data intensive scientific research. Grid Spider requires applications to maintain their referential transparency through a file cache mechanism called "Data Process Memoization Cache" or DPMCache, thereby avoiding to a high degree re-computation. Grid Spider users are scientific researchers who heavily rely on application programs, but seldom write their own code. Prior to processing, both the data and the application are registered with Grid Spider. Grid Spider then controls the execution, and records the coupling of the application, the input data file, and the output data file. To evaluate Grid Spider, we implemented "GEO Grid Spider II", a spacial data mining framework primarily utilized by geo-scientists evaluating satellite data archives.

The organization of this paper is as follows. In Section II, we propose a file cache mechanism called DPMCache, which provides the core of the Grid Spider framework. We then propose the Grid Spider framework, our core concept. In Section III, we discuss GEO Grid Spider II, which is a framework for the processing of satellite data by geo-scientists. In this section we will introduce some interesting results. In Section IV, we provide an evaluation of the Grid Spider framework through simulations, offer related work in Section V, then draw relevant conclusions in Section VI.

II. DATA PROCESS MEMOIZATION CACHE AND GRIDSPIDER

In 1968, very early in the era of computers, memoization [2] was proposed by Donald Michie to optimize computation by reducing the cost of re-calculation; for each function call, an argument and its result is recorded, and the result is used without the computation in subsequent calls to the same function with the same argument. This technique is used in many fields, and the condition for a function for which memoization works properly is, the function always returns the same value for identical arguments. This technique was primarily used in the functional language, but later in [3], Peter Norvig shows that, if only the function has referential transparency, then external automatic memoization is possible. This technique has been applied in many fields [3] [4]. We noticed that this concept can be also applied to the application programs for data intensive science in which applications and vast data archives are shared by many researchers, and that, once the data is acquired, it is never overwritten. We extend the concept of memoization beyond the domain of a single program execution to multiple program executions using a file cache. For this, we need to allow programs to have referential transparency to store the output of the applications and to record the triplet of (1) the application used, (2) the input file applied, and (3) the output file generated. Since the cost of storage is continuously dropping, it is now feasible to divide an application into several modules and store the intermediate results of each module. This will enhance the share-ability of the application.

We propose a file cache system we have called "Data Process Memoization Cache" or DPMCache. An application consists of a sequence of application modules which have referential transparency, and the output of each module is stored as a cache file until storage space is full (Figure 1). When storage becomes full, cache replacement occurs with a replacement policy that is dynamically changeable for the DPMCache based on the history that its framework records.

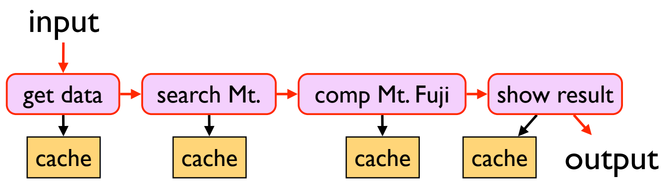

Figure 1. Application list

It is often observed that several applications use identical pre-processing, and in these cases, these applications constitute the application tree (Figure 2). The application tree is a union of the application lists, and the results of shared modules are also shared. For example, processes such as distortion correction and noise removal are required my multiple applications.

We propose the Application Tree Structure Cache Replacement Policy (ATS). The ATS scores each cache file based on the Application Tree Structure. Modules with many branches in an application tree are likely candidates for shared pre-processing. To detect this, we adopt a scoring system in which (1) cache files generated by a module with many branches, (2) cache files created by a module requiring more time, and (3) cache files of a smaller size are all given a higher score. Therefore, cache files with lower scores are selected to be replaced by ATS. Figure 2 shows how ATS

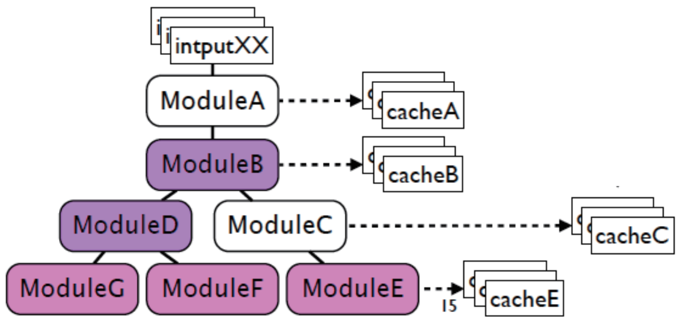

Figure 2. Application Tree

leaves untouched files created by Module B and Module D which score higher due to their many branches.

To utilize this DPMCache scheme, we propose a framework for data intensive scientific research called "Grid Spider" which has the objective of omitting avoidable recomputation by sharing the results of all users of the framework. Grid Spider controls (1) the enforcement of the referential transparency of application modules, (2) the management of data files so that files are unique and are not overwritten, (3) recording the history of history triplets consisting of the module, the input file and the output file, and (4) applying the optimal cache replacement policy to cache files based on the history triplets.

Application modules and input data are registered with Grid Spider in advance, and users construct an application by combining application modules.

Grid Spider is intended to be used in a distributed environment. Figure 3 shows the configuration of the system. The users send an query to the parent node consisting of search areas and applications. The query is then divided into requests that correspond to the divided search areas, and these requests are assigned to child nodes. The child nodes run the application, and they return the results to the parent node. The application tree and the cache files independently correspond to the the child nodes. The assignment of the request depends on the coordinates of the search area, and each child node is assigned its own search areas. If there is a conflict between the assignment nodes, the parent node takes over the load-balancing.

The optimal replacement policy is determined by how the cache is accessed. For example, LRU works well in many cases, but it works poorly when scanning arrays. Therefore, we propose Node Scoring Adaptive Policy or NSAPolicy which dynamically switches the replacement policy in a distributive environment. In our proposal, child nodes send the cache hit rate to the parent node, and the parent node chooses the highest scored policy and applies it to the child nodes. The parent node checks the highest score policy and switches the child nodes's cache replacement policy.

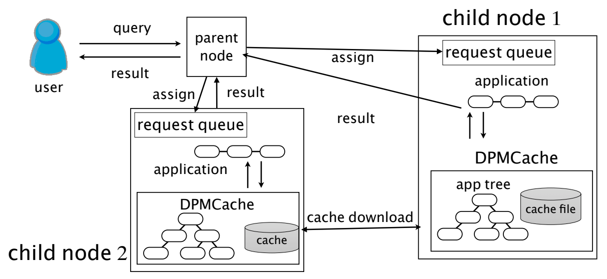

Figure 3. System design

III. GEO GRID SPIDER II

Using Grid Spider, we designed and implemented GEO Grid Spider II (GGSII) as an instance of the Grid Spider framework for geoscientists treating satellite data.

On GGSII we can easily implement search applications using geographical information. Figure 4 shows the example results of a search for circular objects on global aerial maps such as round irrigated farm plots. Figure 5 shows a sample result from a search of face-like objects. This application was implemented by modifying the sample code in OpenCV. Thus, we can easily incorporate other applications into the framework as modules.

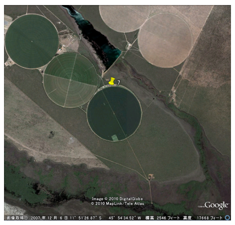

Figure 4. Irrigated farm plots in Brazil

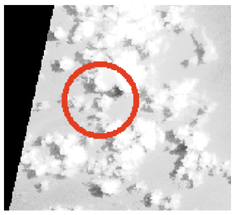

Figure 5. Face search application (Venezuelan coast)

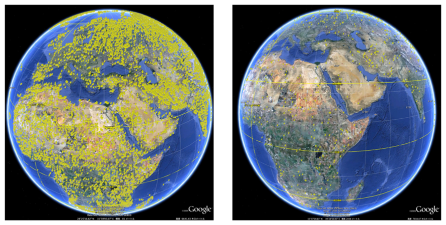

Figure 6. Detected mountains with higher threshold (left) and lower threshold (right)

Figure 6 shows the locations of the similar mountains detected by the algorithm [5]. The left photo depicts the results of the first trial, and the right photo depicts results after the threshold adjustment. In cases where threshold and parameter tuning is desired, the GS framework works quite well.

To make this framework useful to active geoscientists, we consulted them and considered several use cases. In the consideration of distributed system implementation we categorized these use cases into three groups: (1) **world wide** : involving global searches for objects with specified features; (2) **concentrated** : involving focused searches after disasters such as large earthquakes or volcano eruptions; (3) **mixed** pattern : combining both global and focused elements.

IV. EVALUATION

For the evaluation of GGSII, we first performed the simulation, then implemented the system for clusters.

We compared the replacement policies (1) First In First Out or FIFO, (2) Least Recently Used or LRU, (3) Segment LRU, (4) Bimodal Insertion Policy or BIP, (5) the proposal method ATS, and (6) the proposal method NSAPolicy.

A. Simulation

We implemented a event-driven type simulator in Java where the parent-node/child-node ratio is 1:20.

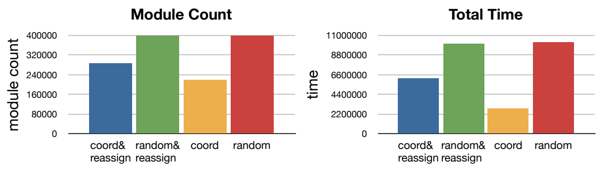

Figure 7. World Wide, Global survey

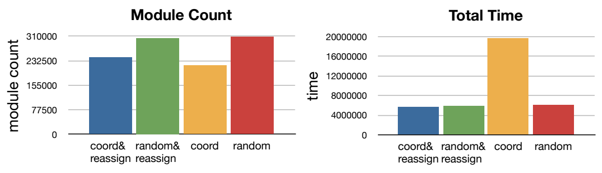

Figure 8. Concentrated, disaster

For the distributed system, we compared the combination of location specific distribution and random distribution against the combination of with reassignment and without reassignment. Figures 7 and 8 show the module counts and the total time for the global distribution use case and the local disaster distribution use case.

Random distribution proved unproductive due to the request being assigned to a node lacking a cache file. Location-specific distribution was more productive in respect to cache hits, but experienced distribution imbalance, especially in disaster cases. We adopted the combination of first location specific distribution migration for load-balancing.

Figure 9 shows the changes by total cache size. This graph show the use case of a world wide survey. ATS performs well.

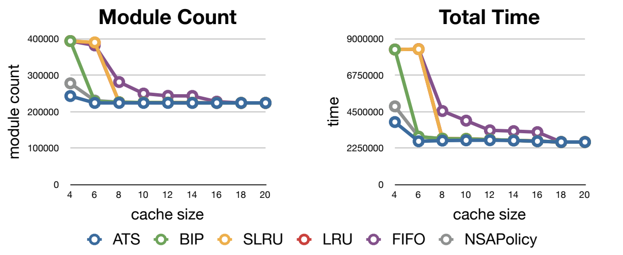

Figure 9. Changes by the total cache size for the intensive global case

B. Experiment on cluster system

We implemented the GGSII on a cluster of eight PowerEdge R410s with an Intel(R) Xeon(R) CPU E553 with 2.40GHz of 4 core x 2, with a 12GB memory and a 430GB HDD. For data, we used Shuttle Radar Topography Mission data or SRTM, and Digital Elevation Model or DEM. Each file was 2.8MB, and the 14,168 files corresponding to almost 80% of the Earth's surface totaled about 40GB. We repeatedly ran the application to detect mountains similar to Mt. Fuji as described in section III, and changed the parameters for each execution.

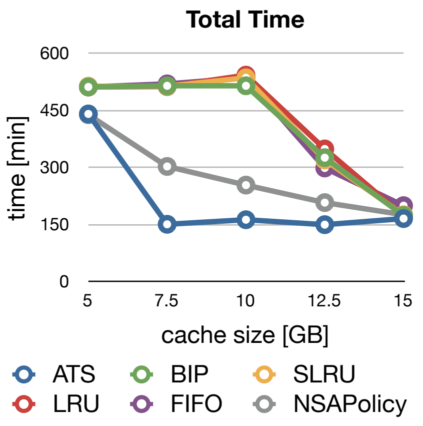

Figure 10. Comparison of query processing time (survey pattern)

Figure 10 shows the results from an experiment similar to the one depicted in Figure 9. ATS performs well to a level comparable to the simulation results.

V. Related Work

A. GEO Grid

Global Earth Observation Grid or GEO Grid [6] is a grid system that was developed by the National Institute of Advanced Industrial Science and Technology (AIST). The application provides through GEO Grid a birds-eye view displayed in the viewer in 3D and in the Volcanic Gravity Flow Simulation. For example, the GEO Grid task force released the crustal movement data and the propagatin animation of the tremors [9] recorded during the 2011 Tohoku Earthquake on Japan's Pacific coast.

B. Cache Consistency

Maintaining cache consistency is important to a cache system in a distributed environment. "If-modified-since" [8] is used on Web systems, and is used to disable one cache when the record is updated on a second cache on a distributed file system such as CODA[7].

In addition, these systems often use read-only data so consistency is maintained. For Gird Spider, the input is write-once, and the cache is write-once because the module has referential transparency. Therefore, cache coherency is automatically maintained.

VI. Concluding Remarks

We are proposing the file cache scheme DPMCache and a framework for data intensive research called Grid Spider. To demonstrate the usefulness of Grid Spider, we implemented GEO Grid Spider II which focuses on satellite data archives and geophysicists. We evaluated the system using both simulation and real implementation, and compared the replacement policies. We have shown the effectiveness of our replacement policy which utilizes the history of the execution of the applications.

For future research, we intend to focus on the time in queue of disk IO, and the communication between each node. In addition, we plan to utilize the scheduling algorithms to reduce processing time.

References

[1] Microsoft Research, "The Fourth Paradigm: Data-Intensive Scientific Discovery", in T. Hey, S. Tansley, K. Tolle, 2009.

[2] D. Michie, "Memo Functions and Machine Learning", Nature, No. 218, pp. 19-22, 1968.

[3] P. Norvig, "Techniques for Automatic Memoization with Applications to Context-Free Parsing", Comput. Linguistics, Vol. 17 No. 1, pp. 91-98, 1991.

[4] J. Mayfield, et al., "Using Automatic Memoization as a Software Engineering Tool in Real-World AI Systems", Proc. of the 11th Conf. on AIA, 1995.

[5] R. Wakuta and T. Sonobe, "SPGF Search Places by Geographical Features all around the world Search and verification system", SAINT2010, 2010.

[6] S. Sekiguchi et al. "Design principles and IT overviews of the GEO Grid", Syst. J., IEEE, 2008.

[7] J. Kistler and M. Satyanarayanan, "Disconnected Operation in the Coda File System." ACM Trans. Comp. Syst., 1992

[8] R. Fielding et al., "Hypertext Transfer Protocol – HTTP/1.1", RFC2616, June 1999

[9] GEO Grid disaster task force, http://disaster-e.geogrid.org/, Jan 2012