In-advance Data Analytics for Reducing Time to Discovery

Jialin Liu
Department of Computer Science
Texas Tech University
Lubbock, Texas, USA
Email: jaln.liu@ttu.edu

Yin Lu
Department of Computer Science
Texas Tech University
Lubbock, Texas, USA
Email: yin.lu@ttu.edu

Yong Chen
Department of Computer Science
Texas Tech University
Lubbock, Texas, USA
Email: yong.chen@ttu.edu

***Abstract*—Scientific workflow involves data generation, data analysis, and knowledge discovery. As the data volume exceeds a few terabytes (TB) in a single simulation run, the data movement, which happens among data generation, data analysis, and knowledge discovery, becomes a bottleneck in most scientific big data applications. Our previous work shows that reusing the analysis results can have a significant potential in reducing the overlap between data movement among compute nodes and storage nodes. In this work, we propose a new *in-advance data analytics* method to augment the result reuse. The fundamental idea of this *in-advance data analytics* method and its prototyping system is to predict the potential useful analytics operations by studying the users' analysis pattern. The predicted analysis operation is pro-actively performed on existing data and the analysis results are stored in an in-memory database for result reuse. The evaluation shows that the in-advance data analytics method and its prototyping system gains 1.2X-6.1X speedup in I/O performance improvement with 50% data overlapping and 10%-100% operation recommendation hit rate. The proposed *in-advance data analytics* method brings a new promising data reduction solution for big data applications.**

***Keywords*-in-advance data analytics, big data, data intensive computing, scientific computing**

I. INTRODUCTION

The current big data challenge in scientific application involves two parts, i.e., data storage and data computing. As the computing power tends to increase much more faster than the storage capability, the poor I/O performance has become a critical bottleneck. The main cause of the I/O bottleneck is slow disk-read speeds compared to both the CPU speed and memory bandwidth. As the size of data collected from scientific instruments and generated from simulations keep increasing rapidly, the I/O access cost dominate an application's execution time. Reducing data movement has never been so important to address the I/O bottleneck issue. Recent studies including in-situ/in-transit data processing [19], [3], [10] (processes the data simultaneously at the time the data are generated), data compression [12], [5], [9] (compresses the data before written to storage), and active storage [16], [14], [18] (moves the computation to storage and performs the analysis directly in place where the data are located) are all along this direction.

Among those techniques, the in-situ/in-transit processing usually desires enough prior knowledge of how scientists conduct the analysis, therefore it is sometimes hard for applications that the scientists prefer to perform iterative and diverse post-analysis. The active storage attempts reducing data movement during analysis phase with an assumption that the storage device is able to deliver the computing power that the processing requires. However, many scientific computations desire more computational power than what storage devices can offer, especially when working on big data problem. Data compression reduces data movement by requiring an additional step to compress and decompress the data.

Both of our previous work (segmented analysis [13]) and the Resilient Distributed Datasets (RDD) design in the Spark data processing framework [20], demonstrate that caching the analysis results in memory can further reduce data movement dramatically. The core idea is to re-use the cached results, instead of raw data, to reconstruct the future analysis. However, these current studies only consider reusing the existing results, and the I/O performance depends on whether the results can be hit.

In this paper, we propose a new method, 'In-advance Data Analytics', to predict the future analysis operation based on history information. The fundamental idea is to analyze the existing analytics logs to extract the analytics pattern. When there is a new analytics operation, the future operation is predicted according to the pattern and the data analytics are conducted ahead.

The rest of the paper is organized as follows. Section II presents a motivating example script and application. Section III presents the definition and detection of the analytics pattern. Section IV discusses the system design and related algorithms. We present performance evaluation results in Section V. Section VI discusses related work and compares them with our proposed strategy. We concludes the discussion in Section VII.

II. MOTIVATION

In climate science community, data are either collected from instruments, e.g., satellite, or generated from simulations with climate models, e.g., general circulation model(GCM [2]) and Global Coupled Climate Models(CM2 [6]). The dataset usually covers a high-dimensional grid with multiple parameters. Scientists query and access different subsets of the data to perform the analysis, which involves large amount of data

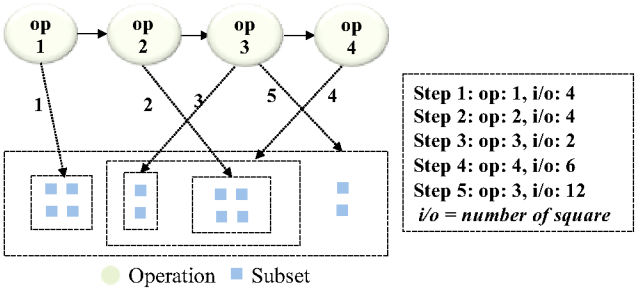

Fig. 1: Data Overlapping among Analytics Tasks

movement. Figure 1 shows a typical workflow, in which different operations/tasks are issued to access different subsets of the data, but the overlapping data among the analytics are usually unaware. For example, in operation 1, when the subset (where the arrow points to) are ready in the compute nodes, we usually delete the data after operation 1 is done. But we can find the later operation, i.e., operation 3 (points to a subset with two square, and also points to the biggest rectangle), will also need the operation 1's data as part of its I/O. Therefore, if we perform operation 3 at the time operation 1 is received, we would reduce the future data movement for operation 3. A motivating real application is illustrated in the following script.

```
Operation 1: Compute Weights
    cdo -f nc -genbil, grid.txt -import_cmsaf
        CFCdm${day}.hdf we.nc
Operation 2: Remap and Selection
    while [$day -le 20100831]
    do
    cdo -f nc -gec, 75
        -remap,grid.txt, we.nc -import_cmsaf
            CFCdm${day}.hdf
         CFCdm${day}.nc
   day= 'expr $day + 1'
   done
Operation 3: Add Values in All Files
   cdo enssum CFCdm201008??.nc cl.nc
```

Listing 1: Operation Sequence for Monthly Number of Clear/Cloudy Days

This example analysis (Listing 1) is to calculate the monthly number of clear and cloudy days [11]. The analysis requires user to conduct a sequence of operations. After each operation, there is an output file for storing the intermediate results. The later operations in this analysis are based on the former operation and its results. As shown in the script, the first operation is to calculate the interpolation weights (operator 'genbil'). These interpolation weights (in 'we.nc') are then reused for every remapping process with the operator 'remap', which maps the daily data to an European domain and select the clear days (i.e. based on the 'lower equal'-operator ('-lec') a map is created where all clear days are set to 1). In the final operation, all intermediate values in 'CFCdm20100801_g75.nc', 'CFCdm20100802_g75.nc', ..., 'CFCdm20100831_g75.nc', are added up using the ensemble sum operator 'enssum'.

This example is a typical case in the scientific data analysis, where the ordered operations are 'pre-defined', and form a fixed sequence to complete the task. The sequence in this example is represented as $seq = \{op1, op2, op3\} = \{genbil, remap, enssum\}$. Such pre-defined analysis sequence will be conducted on different data regions. In other words, scientists may repeat this sequence over different input files and different datasets; for example, **'Seq1 in, out; Seq2 in1, out1; Seq1 in, in1, out2; Seq3 in2, out3; etc'** ('in' denotes input, 'out' denotes 'output', and different numbers represent different files/datasets). Once the pre-defined sequence is detected, we will apply it into the recommendation to proactively conduct the latter operation in a sequence as long as the system receives the former operation in a sequence, such that the data movement (intermediate files) are reduced.

III. Pattern Definition and Detection

As we have shown in the motivation section, the users' analytic activity usually reveals a certain pattern. The In-advance analytics system recommends a potential analysis based on such pattern. The pattern itself, however, varies among users and applications. By analyzing different data analytics in many domain sciences [11], [17], we observe two general patterns which include a *memorable pattern* and a *memoryless pattern*. For the memorable pattern, the users' analytic task is consisted of an ordered operation sequence. A latter operation in this sequence can not proceed without the intermediate results of the former operation in this sequence. The second general pattern, memoryless pattern, represents that future analytics only depend on the current operation, and are not related to the past operations. We detail the discussion below.

A. Memorable Pattern

Our system uses an in-memory database to record each analytic operation, but the challenge for detecting such memorable sequence, is how to find the 'meaningful sequence', which is a serial of operations can be used in fixed order to complete one analysis task. As we have observed in the climate science example, that the inputs of each operation within one sequence are not arbitrary. The latter operation's input is based on the former operation's output, therefore, the overlapping data is the key to detect the meaningful sequence.

We use a directed graph to represent the sequence and describe the data overlap. In the graph, the nodes represent the operations. Each operation is a node, and nodes are connected when the data accessed in operations has an overlap. Different nodes can represent same operation, because they are conducted on different data or different time. The sequence in Figure 1 is now represented in the following graph (Figure 2, left subfigure).

To build the directed graph from users' historical analytics, we follow two rules, i.e.,

1) Operations are consecutive,

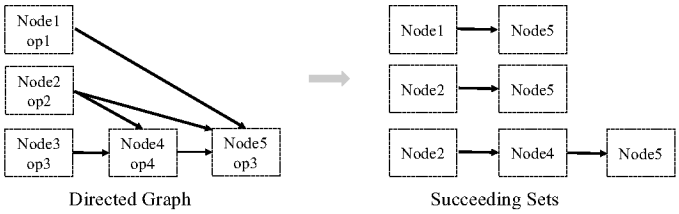

Fig. 2: Succeeding Set Derived from Directed Graph

2) Data has overlapping

New nodes are added in and connected with existing nodes by following these two rules. After building the directed graph from the history analytics, we use depth first search algorithm to derive each sequence, which can be represented as a succeeding set. The opeartion's succeeding set tells what are the operations that happen after this operation and has overlap on I/O. For example,

$$\begin{aligned} &Succeeding(monadd) = \{ymonmul, max\} \\ &Succeeding(mean) = \{min, ydayadd, yhourmul, enssum\} \\ &Succeeding(mean) = \{max, ensstd, enskhistspace\} \end{aligned} \tag{1}$$

in which, Succeeding(X)={operations come after X}.

B. Memoryless Pattern

The previous memorable pattern is commonly observable in users' analytic pattern. In practice, many cases also exist, in which no pre-defined sequence is followed. For example, scientists may neither know much about what is inside the data, nor what analytic can quickly lead to the interesting discovery [15]. Therefore, the analytic activity tends to be random and spanning over the dataset. As the users' analytic activity also tends to be memoryless, random and the future operations only depends on the current analytic operation (no relation with the past operations). Such pattern has the same characteristics with the Markov Chains, thus we represent the pattern using *Markov Chains*. We consider the users' memoryless analytic activity as a stochastic process,

$$\{X^{(n)}, n = 0, 1, 2, ...\} \tag{2}$$

in which, the $X^{(n)}$ can be any operation, e.g., 'cdo mean'.

Suppose the analytic process is a Markov Chain process,we can have

$$P(X^{(n+1)} = i | X^{(n)} = i_{n-1}, ..., X^{(0)} = i_0) = P_{i,j} \tag{3}$$

in which, $P_{i,j}$ represents the probability that the user's next operation will be i given the current analytic operation j. $P_{i,j}$ is independent of time such that future operation only depends on the current operation. Clearly one has

$$P_{i,j} \geq 0, \sum_{i=0}^{\infty} P_{i,j} = 1, j = 0, 1, ... \tag{4}$$

Therefore, we can build a matrix to represent all the operations and its transition probability.

$$P = \begin{pmatrix} P_{0,0} & P_{0,1} & \cdots & P_{0,m} \\ P_{1,0} & P_{1,1} & \cdots & P_{1,m} \\ \vdots & \vdots & \vdots & \vdots \\ P_{m,0} & P_{m,1} & \cdots & P_{m,m} \end{pmatrix} \tag{5}$$

where each element represents the probability of event that operation j comes after operation i.

The above represents one-step transition probability matrix P for a Markov Chain. Using this one-step transition matrix, we can recommend only one future step of analytic operation. For an n-step transition probability, we can derive the P using Equation 6:

$$P_{ij}^{(n)} = P_{ij}^{n} \tag{6}$$

Given an observed analytic sequence $SEQ = \{X\}$, e.g., $seq = \{min, ydayadd, yhourmul, enssum, min, ydayadd, max, ensstd, enskhistspace\}$, we build the Markov chain in three steps. First, we compute the transition frequency F_{ij} by counting the number of transitions from operation i to operation j in one step, e.g., $F_{min,ydayadd} = 2$ and $F_{max,ensstd} = 1$, etc.

Second, from F, we can get the estimates for $P_{i,j}$ in Equation 5. where

$$P_{i,j} = \begin{cases} \dfrac{F_{i,j}}{\sum_{i=1}^{m} F_{i,j}} & \text{if } \sum_{i=1}^{m} F_{i,j} > 0 \\ 0 & \text{if } \sum_{i=1}^{m} F_{i,j} = 0 \end{cases} \tag{7}$$

Thrid, the n-step transition matrix can be calculated using the same method with one-step matrix (Equation 5, 6, 7), except that the frequency should be the number of transitions from operation i to operation j in n steps, e.g., $F_{min,yhourmul} = 1$ and $F_{min,ydayadd} = 0$ for $n = 2$.

IV. System Design

We introduce the In-Advance Data Analytics system and its workflow in this section. As shown in the Figure 3, the system has three major components, i.e., a recommendation component, an in-advance analytics kernel component, and an in-memory database. The key part of this system is its recommendation component, in which the statistical methods are applied to formalize and detect the analytics pattern. The kernel processes the current task and also performs the recommended analytics for future re-use. The current and recommended analytic results (and sub-results) are stored in a key-value database, which is used to speedup the result retrieval and facilitate the analysis. We discuss each component in the following subsections.

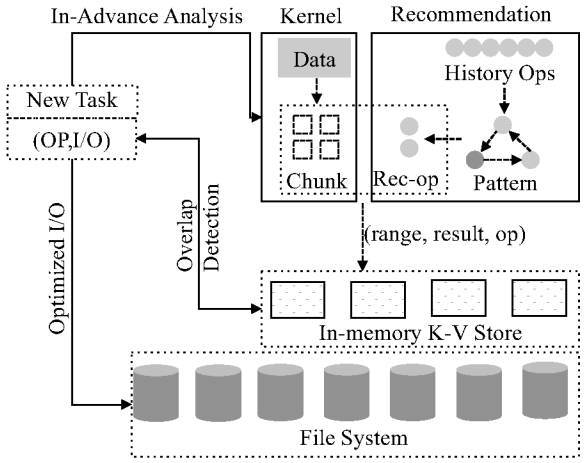

Fig. 3: In-Advance Data Analytics System

A. Pattern Detection and Analysis Recommendation

As we have described in the Section III, the result of pattern detection includes at least the memoryless Markov pattern (since memorable sequence may not exist). Either only one pattern exists, or both patterns exist, i.e., memorable sequence pattern P_s and memoryless Markov pattern P_m, the In-Advance Data Analytics System is designed to choose the operations with memory constraints.

Initially, the system randomly selects one analytic operation from available operations (current test is based on CDO operators [1]), when there is no historical logs. After two iterations (two is the minimum number to construct both patterns), the system begins to construct each pattern, using the methods as discussed in Section III. This pattern construction is conducted periodically, e.g., user can define a $period = 1$, which means pattern construction happens for 'every' new task, such that the existing pattern sequences or transition matrix are updated constantly. Since the recommended operations are performed on the finer-grained segments (the original accessed data is logically chunked into segments, please see Section IV-B), the additional sub-results will consume increasing amount of memory; therefore, for each recommendation, we set a limited size, ls. If the recommended operations will generate more analysis results than the allocated memory, we will not recommend it. In practice, memorable sequences are much more preferable than memoryless patterns, so the recommended operations are chosen from the memorable sequence ($p_s(x)$) first and memoryless pattern ($p_m(x)$) second within this memory limit.

B. In-Advance Analysis Kernel

The recommended analytics require additional computational resources, though the computing is considered "virtually free" for big data problems where data movement dominates the runtime. As shown in Figure 3, the 'Kernel' part contains 'Data' and 'Chunk'. The 'Data' here refers to the current operation's input, which is already fetched from storage nodes to compute nodes. Before removing the data or writing back the update to storage nodes, the In-Advance Data Analytics System recommends the potential analytic operations, i.e., 'Rec-op', and performs on the ready 'Data'. If the future operation has overlapping with existing analytic results, the I/O is reduced by reusing them. The 'Data' are segmented (in this paper, we use the term 'segment' interchangeably with 'chunk') in order to gain finer granularity analytic sub-results and to construct the future analytic results [13]. The finer the sub-results, the more chances the sub-results can be re-used, because the possibility of useable overlap among data is increased. Besides the granularity, how to segment the data plays an important role in the re-using ratio. We adopt a 'dimension-driven segmentation' method [13], and add the 'Rec-op' in the kernel. Therefore, the kernel in this system has more analytic work to perform on the segmented datasets, and generates more analytic sub-results for future re-use.

V. Evaluation

A. Experimental Setup

We ran all the experiments on the Hrothgar, a 640-node (7680 cores) Linux cluster. Each node in the cluster contains two Intel Xeon 2.8 GHz 6-core processors with 24 GB of memory. The nodes are connected with DDR Infiniband. We generated several 4-D climate datasets using PnetCDF, which are 769MBs, 179GBs, 358GBs, and 448GBs.

B. Evaluation and Results Analysis

Frist, we evaluated the hit rate of the recommendation algorithm and the I/O performance of In-Advance Data Analytics system with/without the recommendation. For the evaluation of the recommendation, we have two patterns, i.e., memorable pattern and memoryless pattern, to be measured. Our system predicts all operations within a pattern sequence as long as this pattern is detected. For memoryless pattern, we observe the hit rate with different number of prediction steps. The number of prediction steps refers to the number of operations to be selected based on the transition probability from high to low.

Regarding the training and test set, we have a set of real scripts collected from climate community [11]. The majority of operations in the scripts are memorable/fixed patterns. We also generated two synthetic scripts that simulate the random analytics. The two training sets have 100 and 1000 operations separately. In practice, though the different patterns are mixed, our current setup evaluates them separately (using different training data) is close to the real situations.

The number of unique operations is 52, and the test set is another 100 and 1000 operations set. The same set of experiments are repeated 180 times and the result plotted is their average. Figure 4 shows the hit rate along different prediction steps. As shown in the Figure 4, the more the training set, the more accurate the prediction. However, Figure 4 shows that the real script is immune to the number of prediction steps, and the real scripts get higher hit rate even with one predicton step. This is because the real script contains

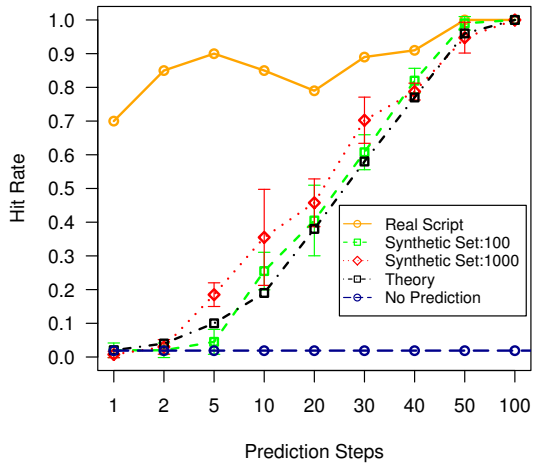

Fig. 4: Hit Rate with Different Prediction Steps

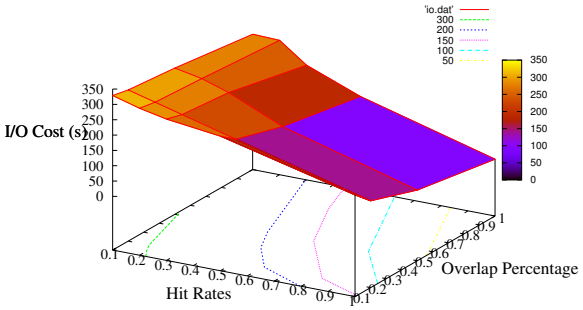

Fig. 5: I/O Performance with Recommendation

more meaningful analytics pattern, and once detected, later occured operations are easily predicted. On the other hand, the randomly generated training sets have few meaningful patterns, and only the prediction steps can affect their hit rate. The experimental results confirm our analysis. Besides, in Figure 4, the 'Theory' line shows a theoretical hit rate, i.e., the number of steps divided by total unique operations. The Markov algorithm outperformed the pure random selection due to the accumulation of the transition probability. This transition probability matters more in practice. At last, 'No Prediction' clearly has no capability to predict the future, which is used in the previous work [13].

The hit rate is only one factor that can have effect on the I/O performance though. A high hit rate still can not guarantee a high I/O performance, but it is always better to achieve a good hit rate. Therefore, we would use the Markov algorithm to predict 30 or more steps (users can decide) to gain 50%+ hit rate. For memorable pattern, we only use one prediction step, which is enough to achieve high hit rate.

To observe how the I/O can be improved with the recommendation, we run the In-Advance Data Analytics system over a 179GB synthetic climate dataset. In this test, we initialized one memcached instance, with 1024 MB capacity. The training scripts for the recommendation is the previous set. We submitted analytics operations randomly to the In-Advance Data Analytics system and measured the the I/O time with and without the recommendation. The tests were conducted with 96 processes and 100 total OSTs. Figure 5 shows that the I/O cost becomes less as the recommendation providing higher hit rate and the data has more overlapping. This observation confirms two necessary conditions, in which the data movement is reduced only if both the computation and I/O are overlapped. Observing the surface in the figure, we can also find that the I/O performance is not guaranteed given certain pair of hit rate and overlap percentage. For instance, the case with hit rate=0.2, overlap percentage=0.9, should be better than the case with hit rate=0.1 and overlap percentage=0.9, but it turns out the opposite. This situation occurs because that the hit rate is a probability that the recommended operation is hit, and does not mean it will be hit. Therefore, we need to use our recommendation algorithm to predict more steps of operations and cache more results for future. On the other hand, the overlap percentage is proportional to the I/O performance. This observation shows the importance of our system's finer segmentation, which helps in detecting more data overlap.

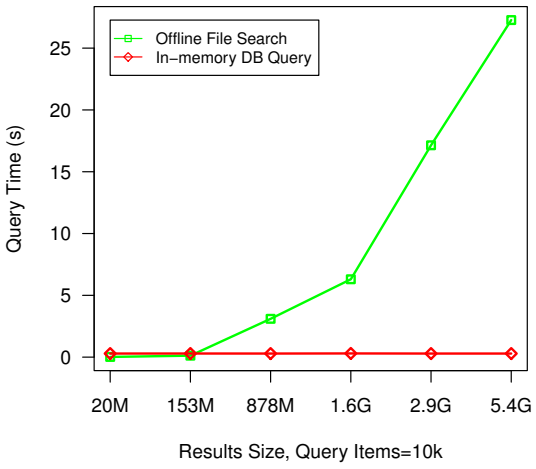

Fig. 6: Results Query Performance Comparison

In group two, we evaluated the query efficiency using the distributed in-memory database. Figure 6 shows the performance comparison with and without the in-memory database. This group of tests used one instance memcached and allocated 6GBs memory. We generated different size of analysis results

and stored them in file and the database separately. We performed 10,000 times of query and plot the total response time. To be precise, in each test, the results stored in file are all read into memory and the query is performed in memory. On the other hand, the in-memory database is started from the first test and keeps running as a distributed server. We can observe that using in-memory database, the query response is constant and is only proportional to the number of query items. While the traditional 'offline file search' keeps increasing the response time as more results accumulated. Our evaluation shows that the distributed in-memory database is promising to help reducing the data movement for big data analytic problems.

VI. Related Work

Compared to caching raw data, caching result is a relatively new research area. In cloud computing, caching results is a technique to achieve fault tolerance (RDD [20])cite linkedin and incremental computing [7], [4]. The idea of RDD is to keep the partitioned operation and recompute the data using lineage for fast fault tolerance. In contrast, Our in-advance system is designed to reuse the analysis results by detecting the computation and I/O overlapping. Knowledge discovery is another area that is related to our work. Knowledge discovery is 'the nontrivial extraction of implicit, previously unknown, and potentially useful information from data' [8]. It focuses on using various machine learning or statistical methods to explore the data for unknown knowledge. Our work is similar in the way of predicting the useful results, but the difference is that we design a lightweight system that observes the user's analysis habit and tries to make a recommendation for scientists.

VII. conclusion

In this study, we have introduced a new *in-advance data analytics* method for reducing data movement for big data analysis and big data applications. The proposed in-advance data analytics leverages a prediction method that uses minimal computing resources to generate useful analysis results in advance. As data movement dominates the run time of big data analysis, and computing is virtually free for big data problem, the in-advance data analytics can be a promising solution that fully leverages data locality and reduces the data movement and the time to solution.

VIII. Acknowledgments

This research is sponsored in part by the National Science Foundation under grant CNS-1338078 and CNS-1162488. We also acknowledge the High Performance Computing Center (HPCC) at Texas Tech University for providing resources.

References

[1] Climate data operator. https://code.zmaw.de/projects/cdo.

[2] General circulation model. http://en.wikipedia.org/wiki/Community_Climate_System_Model.

[3] H. Abbasi, G. Eisenhauer, M. Wolf, K. Schwan, and S. Klasky. Just in time: adding value to the io pipelines of high performance applications with jitstaging. In *HPDC*, pages 27–36, 2011.

[4] P. Bhatotia, A. Wieder, R. Rodrigues, U. A. Acar, and R. Pasquin. Incoop: Mapreduce for incremental computations. In *Proceedings of the 2Nd ACM Symposium on Cloud Computing*, SOCC '11, pages 7:1–7:14, New York, NY, USA, 2011. ACM.

[5] K. M. Curewitz, P. Krishnan, and J. S. Vitter. Practical prefetching via data compression. In P. Buneman and S. Jajodia, editors, *Proceedings of the 1993 ACM SIGMOD International Conference on Management of Data*, pages 257–266, Washington, D.C., 26–28 May 1993.

[6] T. L. Delworth. Gfdl's cm2 global coupled climate models. *J. Climate*, 19:643674, 2006.

[7] J. Ekanayake, H. Li, B. Zhang, T. Gunarathne, S. hee Bae, J. Qiu, and G. Fox. Twister: A runtime for iterative mapreduce. In *In The First International Workshop on MapReduce and its Applications*, 2010.

[8] W. J. Frawley, G. Piatetsky-Shapiro, and C. J. Matheus. Knowledge discovery in databases: An overview. *AI Mag.*, 13(3):57–70, Sept. 1992.

[9] M. Gardner, W. chun Feng, J. Archuleta, H. Lin, and X. Ma. Parallel genomic sequence-searching on an ad-hoc grid: Experiences, lessons learned, and implications. In *SC'2006 Conference*, Tampa, FL, Nov. 2006.

[10] J. Gray, D. T. Liu, M. A. Nieto-Santisteban, A. S. Szalay, G. Heber, and D. DeWitt. Scientific data management in the coming decade. Technical Report MSR-TR-2005-10, Microsoft Research (MSR), Jan. 2005.

[11] F. Kaspar, U. Schulzweida, and R. Muller. Climate data operators as a user-friendly processing tool for cmsaf's satellite-derived climate monitoring products. In *the Proc. of the EUMETSAT Meteorological Satellite Conference*, 2010.

[12] S. Lakshminarasimhan, J. Jenkins, I. Arkatkar, Z. Gong, H. Kolla, S.-H. Ku, S. Ethier, J. Chen, C.-S. Chang, S. Klasky, R. Latham, R. B. Ross, and N. F. Samatova. ISABELA-QA: query-driven analytics with ISABELA-compressed extreme-scale scientific data. In *SC 2011, Seattle, WA, USA, November 12-18, 2011*, page 31, 2011.

[13] J. Liu, S. Byna, and Y. Chen. Segmented analysis for reducing data movement. In *the Proc. of the IEEE International Conference on Big Data,(Bigdata'13)*, 2013.

[14] X. Ma and A. L. N. Reddy. MVSS: An active storage architecture. *IEEE Transactions on Parallel and Distributed Systems (TPDS)*, PDS-14(10):993–1005, Oct. 2003.

[15] A. Parameswaran, N. Polyzotis, and H. Garcia-Molina. Seedb: Visualizing database queries efficiently. Technical report, Stanford University, 2013.

[16] J. Piernas, J. Nieplocha, and E. J. Felix. Evaluation of active storage strategies for the lustre parallel file system. In *SC'07*. ACM/IEEE, Reno, NV, Nov. 2007.

[17] D. Wang, C. Zender, and S. Jenks. Efficient clustered server-side data analysis workflows using swamp. *Earth Science Informatics*, 2(3):141–155, 2009.

[18] R. Wickremesinghe, J. S. Chase, and J. S. Vitter. Distributed computing with load-managed active, storage. In *Proc. 11th IEEE International Symposium on High Performance Distributed Computing (11th HPDC'02)*, pages 13–23, July 2002.

[19] H. Yu, C. Wang, R. W. Grout, J. H. Chen, and K.-L. Ma. In situ visualization for large-scale combustion simulations. *IEEE Computer Graphics and Applications*, 30(3):45–57, May/June 2010.

[20] M. Zaharia, M. Chowdhury, T. Das, A. Dave, J. Ma, M. McCauley, M. Franklin, S. Shenker, and I. Stoica. Resilient distributed datasets: A fault-tolerant abstraction for in-memory cluster computing. Technical Report UCB/EECS-2011-82, EECS Department, University of California, Berkeley, Jul 2011.