On Efficiently Capturing Scientific Properties in Distributed Big Data without Moving the Data:

A Case Study in Distributed Structural Biology using MapReduce

Boyu Zhang*, Trilce Estrada†, Pietro Cicotti‡, Michela Taufer*
*University of Delaware
{bzhang, taufer}@udel.edu
†University of New Mexico
{estrada}@cs.unm.edu
‡San Diego Supercomputer Center
{pcicotti}@sdsc.edu

***Abstract*—In this paper, we present two variations of a general analysis algorithm for large datasets residing in distributed memory systems. Both variations avoid the need to move data among nodes because they extract relevant data properties locally and concurrently and transform the analysis problem (e.g., clustering or classification) into a search for property aggregates. We test the two variations using the SDSC's supercomputer Gordon, the MapReduce-MPI library, and a structural biology dataset of 100 million protein-ligand records. We evaluate both variations for their sensitivity to data distribution and load imbalance. Our observations indicate that the first variation is sensitive to data content and distribution while the second variation is not. Moreover, the second variation can self-heal load imbalance and it outperforms the first in all the fifteen cases considered.**

I. Introduction

Driven by the constant increased in "born digital" data, data-driven computing provides new challenges in data management and analysis in the life sciences and in many other disciplines ranging from logistics (traffic analysis) to sociology (social media). These challenges are very different than the traditional simulation challenges because dealing with millions of individual records and terabytes of data. State-of-the-art analysis methods based on clustering and classifications for traditional simulations often require the comparison of complex data records in an iterative process. When very large amounts of data records are distributed across a large number of nodes of a distributed memory system, the movement and comparison of data can have a significant impact on the speed with which the system can complete an analysis [1]. New methodologies are needed to analyze data when it is distributed across nodes of large distributed memory systems.

We propose a methodology that allows the effective management and analysis of large datasets composed of millions of individual data records in a distributed manner on large distributed memory systems. Our methodology overcomes the difficulties associated with two existing extremes: a centralized data distribution system based on a database or central data store and, at the other end of the extremes, a fully distributed data management system (such as those in used in many MapReduce-based data implementations). The methodology is based on small exchanges of data properties or property aggregates among the nodes of the distributed memory system, rather than the need to move and compare data records during every step of the analysis. The key step consists of a data reshaping that is applied to each individual data record concurrently. This step projects relevant extracted data properties into a type of metadata. We implement two different methods to manage and analyze the reshaped data. In the first variation, called GlobalToLocal, the extracted properties are moved across nodes to build a global view of the dataset; these properties are iteratively and locally analyzed while searching for some class or cluster convergence. In the second variation called LocalToGlobal, partial properties are analyzed locally on each node to generate a property aggregate, which is a scalar number that summarizes local properties of the data. In this variation, instead of exchanging extracted properties, LocalToGlobal only exchanges a few scalar property aggregates across the nodes. Both approaches are based on the general theme of moving computation to the data. We integrate the two variations of our algorithm into the MapReduce paradigm [2] and use them to study a complex structural biology dataset of millions of ligand molecules. We test the two variations using the Gordon SDSC (San Diego Supercomputer Center) supercomputer, the MapReduce-MPI library [3], and a structural biology dataset of 100 million ligand records.

To sum up the key contributions of this paper are threefold:

- We present a general methodology to encode properties of datasets with many data records in a parallel manner when the records are distributed across nodes of a distributed memory system such that we can effectively capture properties while pursuing scalability.
- We present two variations of the methodology and their implementations using the MapReduce paradigm.
- We compare the performance of the two MapReduce implementations for a complex dataset of 100 million ligand records on the SDSC Gordon Supercomputer.

The rest of this paper is organized as follows: Section II provides a brief overview on how traditional data analysis is being affected by new distributed data acquisition techniques. In this section we also outline the general methodology proposed in this paper and how it can benefit a scenario where

data is collected in a decentralized way. Section III shows a case study where our methodology is applied to structural biology datasets residing in large distributed memory systems. Section IV presents our evaluation and results. In particular we thoroughly evaluate performance of two variations of our algorithm. Our results show how the underlying scientific problem can affect performance of one of these variations, while the other preserves its efficient treatment of the data regardless of the logical data distribution. Section V gives an overview of related work, and Section VI concludes the paper with a brief discussion and future directions.

II. Distributed Data Acquisition and Processing

A. Semi-decentralized and fully-decentralized memory systems

Data considered in this paper is composed of a large number of individual data records and is distributed across the nodes of a large distributed memory system. More specifically, we consider semi-decentralized and fully-decentralized distributed memory systems. In a semi-decentralized system, processes report to more than one node, usually to the closest one, and take advantage of locality by reducing expensive data transfer and potential storage pressure. In contrast, in a fully-decentralized system, each node stores its own data, which reduces the need of data transfers and increases the amount of locally stored data. Logically, the entire dataset can converge toward one or multiple scientific properties, or it may not convey any scientific information. Physically, data with similar scientific properties may agglomerate in topologically close nodes or may be dispersed across nodes. Logical and physical tendencies are not known a priori when data is generated in semi- or fully-decentralized systems. In general, scientists must move data across nodes in order to analyze and understand it. This process can be extremely costly in terms of execution time. For the sake of completeness, we define three scenarios that resemble the challenging conditions faced by scientists when dealing with distributed systems with large amounts of data. The scenarios consist of data distributed in:

- a semi-decentralized manner in which data with similar properties are generated by and stored in specific nodes;
- a fully-decentralized, synchronous manner in which data is gathered at regular intervals producing a uniform distribution of data properties across the nodes in a round-robin fashion; and
- a fully-decentralized, asynchronous manner in which every node acts by itself and properties are stored randomly across nodes.

B. From data records to properties and property aggregates

We address the data analysis in the three aforementioned scenarios by moving the computation to the data. We accomplish this by keeping the scientific data on the node where it was originally generated and redefine the analysis algorithm to move the analysis to the data. When we analyze scientific data e.g., for clustering or classification, we often focus on specific properties. For example, when dealing with structural biology datasets, properties include molecular geometries or location of a molecule in a pocket. These properties can be captured and encoded across the dataset concurrently by analyzing each single data record independently. In this case, processing large datasets can be decoupled into a preprocessing to extract concise properties that can be represented as metadata, followed by the analysis of the extracted properties to profile scalar property aggregates. Across scientific domains, there is not a single way to extract properties. In this paper, we use a method for capturing molecular geometries in structural biology datasets based on projections and linear interpolations. This method effectively reduces dimensionality of each distributed data record in a parallel manner without any data movement across nodes. Hence, we can effectively and accurately capture data properties while pursuing scalability. The analysis is applied to the extracted properties rather than the whole scientific dataset. Note that we implicitly transform the analysis problem from a clustering or classification problem on scientific data into a search on extracted properties. Now that we no longer need to move the scientific data, the next challenge confronts ways to study the extracted properties in an efficient, distributed manner. The analysis operations applied to our dataset are associative and commutative (e.g., sum or maximum property values), and thus they can be applied to the local extracted properties on each node to generate local scalar property aggregates (SPAs). In the analysis phase either larger extracted properties can be exchanged less frequently or SPAs can be exchanged more frequently across nodes to rebuild a global view of the data scientific meaning. Although extracted properties may be larger than SPAs, both properties and SPAs are smaller than the original scientific data.

III. Case Study: Structural Biology Datasets

A. Scientific datasets

In studies of disease processes, a common problem is to search for small molecules (ligands) that can interact with a larger molecule such as a protein when the protein is involved in a disease state. More specifically, when ligands dock well in a protein, they can potentially be used as a drug to stop or prevent diseases associated with the protein's malfunction. This is a common type of analysis done in drug development research. In this paper, as an example of its applicability, we apply our methodology to structural biology datasets containing large numbers of ligand conformations from protein-ligand docking searches. Computationally, a protein-ligand docking search seeks to find near-native ligand conformations in a large dataset of conformations that are docked in a protein [4]. A conformation is considered near-native if the Root-Mean Square Deviation (RMSD) of the heavy atom coordinates is smaller than or equal to two Angstroms from the experimentally observed conformation. Algorithmically, a docking simulation consists of a sequence of independent docking trials. An independent docking trial starts by generating a series of random initial ligand conformations; each conformation is given multiple random orientations. The resulting conformations are docked into the protein-binding site. Hundreds of thousands of docking attempts are performed concurrently. Molecular dynamics simulated annealing is used to search for low energy conformations of the ligand on the protein pocket. Traditionally, docked conformations with minimum energy are assumed to be near-native. However, it has been shown that this is not always the case [1]. Docked

conformations with minimum energy are not necessarily near-native. In previous work, we showed that compact clusters of docked conformations grouped by their geometry are more likely to be near-native than the individual conformations with lowest energy [1]. Large numbers of ligand conformations were sampled through Docking@Home (D@H) project in the past five years and are used in this paper as the dataset of interest. very large number of docked ligand conformations that have to be compared to each other in terms of their geometry.

Traditionally, a naïve approach used to group similar structural biology conformations is through geometry-based clustering. Such technique requires that data is stored in a centralized location in order to compute the Root-Mean Square Deviation of each ligand with all the other ligands in the dataset. The analysis requires the molecular dataset to be moved entirety into a central server. Figure 1 shows an abstract representation of a centralized data analysis system in which all the ligands have to be moved to a local storage where they can be compared and clustered. This fully centralized approach is not scalable and can result in serious storage and bandwidth pressure on the server side. Thus, the challenge involves ways to efficiently find these dense ligand clusters of geometric representations, especially when the data is acquired in a distributed way. While each conformation is small (of the order of ten Kbytes), the number of conformations that have to be moved across the distributed memory system and have to be compared to each other is extremely large; depending on the type and number of proteins, the conformation dataset can comprise tens or hundreds of millions of ligands. This scenario is expected when thousands of processes perform individual docking simulations and store their results locally. Thus, in this paper we analyze the scenarios where data is generated and partially analyzed in a semi- and fully-decentralized way, taking advantage of the large distributed compute power and storage of Gordon at SDSC.

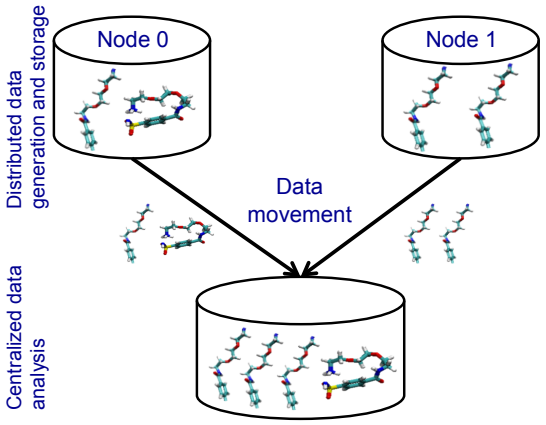


Fig. 1: Traditional centralized comparison and clustering of large datasets generated on a distributed memory system.

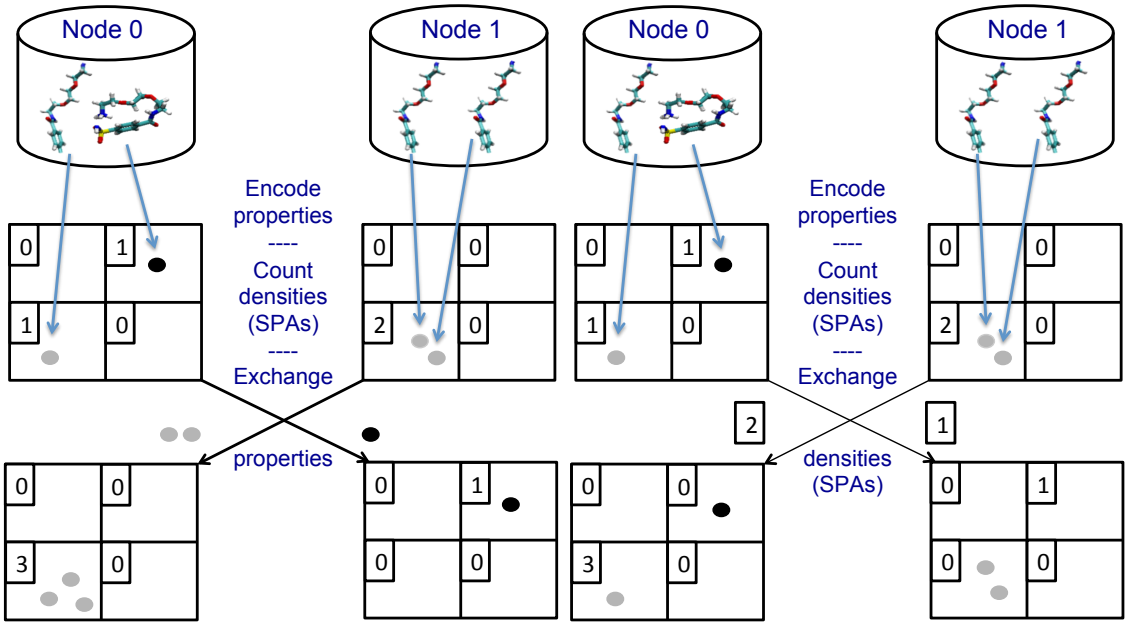
B. Capturing relevant properties

Our preprocessing of docking simulation results requires us to extract the geometrical shape (property) of each docked ligand in the docking pocket of a protein. In this process we perform a space reduction from the scientific data (atom coordinates of the ligand conformations) to the extracted property (compressed values representing ligand geometries in the docking pocket). The main advantage of our space reduction is that it does not rely on calculations involving the whole or partial dataset as do most traditional algorithms, such as k-means and Fuzzy c-means clustering. On the contrary, our space reduction is applied individually to each ligand conformation by transforming each molecule containing p atomic coordinates in the three-dimensional space ($p \times 3$) into a single three-dimensional point (1×3) in the Euclidean space (3-dimensional space of property-encoding points). This process is based on projections and linear interpolations and can be summarized as follows: Given a ligand with p atomic coordinates (x_i, y_i, z_i, with i from 1 to p), we perform a projection of the coordinates in their respective 3 planes (x, y), (y, z), and (z, x). Each projection results in a set of 2D points on the associated 2D plane. For each projection, we compute the best-fit linear regression line over the projected points. We use the coefficient of each model as a coordinate of the 3-dimensional point to encode the conformational geometry of its corresponding ligand. Our space reduction has the desired property of projecting ligands with a similar geometry closer into the newly defined 3-dimensional space. Thus, the converged cluster is the one with the highest density of points.

By dealing with property-encoding points rather than raw atom coordinates, we implicitly transform the analysis problem from a clustering or classification problem into a property aggregate search. Extracted properties can be unpredictably distributed across the 3-dimensional space of property-encoding points and across the nodes of the distributed memory system. Thus, the next challenge we face is how to efficiently count the aggregates of close property-encoding points in a distributed way, that is, how to generate a scalar property aggregate (SPA) that represents local cluster densities. To address this challenge, we use our two general algorithmic variations, i.e., GlobalToLocal and LocalToGlobal. In GlobalToLocal, the extracted properties are exchanged among nodes across the distributed memory system to rebuild a global view of the information content in the scientific data so that similar properties eventually reside on the same node or nodes that are topologically close to each other (Figure 2.a). The atom coordinates of the ligands generating the property-encoding points are not moved from their original system nodes. Property densities, used as SPAs, are then iteratively counted locally while searching for convergence. In LocalToGlobal, instead of exchanging extracted properties, nodes exchange partial densities representing a scalar property aggregate (Figure 2.b). Each node preserves a global vision of the properties associated with its local data and counts its densities before shuffling them with other nodes.

C. Counting property densities

For the sake of efficiency, in both variations we reshape the 3-dimensional space of property-encoding points into an octree and search for the densest octants that contain the solution to our analysis problem. Our search for dense octants (or properties) begins with each node generating its octree on its own local data by recursively subdividing the 3-dimensional space into octants (the nodes of the octree). It proceeds with



(a) GlobalToLocal (GL) (b) LocalToGlobal (LG)

Fig. 2: Examples of exchange of properties in the GlobalToLocal variant and exchange of scalar property aggregations in the LocalToGlobal variant.

each process exploring its octree, i.e., moving up or down along the tree branches depending on whether a dense enough octant is found or not. During the search, nodes may exchange either extracted properties or scalar property aggregates based on which variation of the algorithm is used. As shown in Figures 2.a and 2.b, in both variations, each node transforms its local data into the 3-dimensional space as described in the preprocessing step above. Since nodes work on disjointed data, they can perform the space reduction locally and concurrently. The local property-encoding points can populate the space in an unpredictable way, for example, they can reside across multiple subspaces; each node has a disjointed vision of the whole space. In GlobalToLocal, the extracted properties are communicated among nodes so that each one has all the information needed to analyze similar properties globally. The analysis is applied to each node's local properties iteratively. In LocalToGlobal, after the data preprocessing, each node applies the data analysis operation to its local extracted properties and computes scalar property aggregates based on densities. The SPAs are communicated among nodes so that each node has all the partial results that it needs to analyze the density globally. An aggregate operation (i.e., sum) is applied to the partial results. Communications of properties and aggregates are done iteratively: communication in GlobalToLocal happens only once and the communicated package is larger in size while communication in LocalToGlobal happens multiple times and each time the communicated package is smaller in size. The MapReduce paradigm naturally accommodates the capturing of properties from local data and the iterative search for either properties or densities in its Map and Reduce functions respectively. Thus, we integrated our two variants into the MapReduce-MPI framework rather than implementing a new MPI-based framework from scratch.

IV. Tests and Evaluation

A. Platform setup

The platform used in this work is Gordon, a Track 2 supercomputer at SDSC, which is allocated as an XSEDE resource. Gordon is ideal for our work because it features a flash memory storage private to each compute node that can be used to simulate a fully distributed memory system. We use 64 Gordon compute nodes, each of which contains two 8-core 2.6 GHz Intel EM64T Xeon E5 (Sandy Bridge) processors and 64 GB of DDR3-1333 memory, and mounts a single 300 GB SSD. The nodes are connected by a $4 \times 4 \times 4$ 3-dimensional torus with adjacent switches connected by three $4\times$ QDR InfiniBand links. We integrated the GL and LG algorithms in MapReduce-MPI [3].

B. Data setup

We generate five synthetic datasets from real Docking@Home data to fit specific property distributions in the 3-dimensional space (logical distributions); these distributions reflect five scientific conclusions in terms of convergence towards specific ligand geometries. Each of the dataset consists of 100 million ligands and is 250GB in size. We also distribute the datasets across the nodes' storage (physical distribution) in three ways to fit the different manners distributed data can be generated as presented in Section II.

C. Logical distribution

The information latent in data can have different contents and meanings. For example, the collected ligands may more or less strongly converge toward a single similar geometry or may not converge at all. This is reflected in the extracted

properties and the distribution of property-encoding points in the 3-dimensional space. We consider five scenarios with five different scientific conclusions. In the first scenario with one dense cluster (1D), the information content in the scientific data strongly converges toward one ligand conformation; the 3-dimensional points densely populate one small region of the space while most of the remaining space is empty. We select ligands from the D@H dataset whose geometries generate 3-dimensional points with normal distribution around one point in the 3-dimensional space with standard deviation equaling 0.1 for this scenario. In the second scenario with one sparse cluster (1S), the information content in the data more loosely converges toward one ligand conformation; the 3-dimensional points sparsely populate one larger region of the space. The ligand geometries generate points with normal distribution around one point in the 3-dimensional space with standard deviation equaling one. In the third scenario with two dense clusters (2D) and the fourth scenario with two sparse clusters (2S), we extend the first and second scenarios by presenting scientific cases in which information in the data either strongly or loosely converge toward two major conformations rather than one. More specifically, in the third scenario, ligand geometries are mapped onto points with normal distribution around two separate points with standard deviation equaling 0.1; in the fourth scenario, we generate points with normal distribution around two separate points with standard deviation equaling 0.5. In the fifth and last scenario with a uniform distribution of the information (UN), the 3-dimensional space does not convey any main scientific conclusion; no single ligand conformation was repeatedly generated. This scenario can happen, for example, with insufficient sampling or the inaccurate mathematical modeling.

D. Physical distribution

To simulate the distributed datasets when generated in semi- or fully-decentralized systems, we consider three different physical distributions of the data across the nodes of Gordon: Uniform, Round-Robin, and Random. (1) In the uniform physical distributions, property-encoding points that belong to the same subspace in the logical distribution are located in the same physical storage. This scenario happens most likely in a semi-decentralized distribution in which points mapping close properties are collected by the same node or topologically close nodes; hence, there is uniformity of the property-encoding points inside the nodes' storage. (2) In the round-robin physical distributions, points that belong to the same subspace in the logical distribution are stored in separate physical storage in a round-robin manner. This scenario happens most likely in a fully-decentralized, synchronous distribution in which points are collected at each predefined time interval; hence, the data points for each time interval are stored in separate storage across the distributed system. (3) In the random physical distributions, points are randomly stored in the physical storage of all the system nodes. This scenario simulates the fully-decentralized asynchronous manner. In each of the scenarios, the whole dataset is roughly evenly distributed among the 64 physical storage, giving to each one of the physical storage around 1.6 million ligand confirmations.

Figure 3 shows the total time in seconds (aggregated across all processes) of GlobalToLocal (GL) and LocalToGlobal (LG) for four of the five logical distributions (i.e., one dense cluster 1D, two dense clusters 2D, one sparse cluster 1S, and uniform UN) and one of the three physical distributions (i.e., round-robin physical distribution). The other performance results are omitted in the figure for space constraints but convey the same conclusions. All the results are included in the discussion below and the complete set of times is reported in Table I in the appendix. The total time across the 1024 Gordon's cores is cut down in four time components, i.e., Map, Shuffle, Overhead and Reduce. Map includes the time a process spends extracting properties during the preprocessing and searching across subspaces in the tree. Shuffle is the time spent exchanging properties in GlobalToLocal or densities in LocalToGlobal. Overhead is the time introduced by the MapReduce-MPI library for either synchronizing processes at the end of each MapReduce step or for awaiting certain processes to complete their Map or Reduce in case of load imbalance. Reduce is the time to aggregate properties in GlobalToLocal or the time to aggregate densities in LocalToGlobal. In the figure, we report the average execution time in seconds over three runs; the time traces are obtained by using TAU [5]. We instrumented the source code so only a limited number of events are measured by TAU; thus, the overhead introduced by TAU is negligible.

E. Results and discussion

For all but the uniform logical distributions (UN), we observe that GlobalToLocal performance is impacted by the logical distribution of properties in the dataset and a substantial overhead is reported compared to LocalToGlobal, i.e., up to two orders of magnitude larger than LocalToGlobal when properties (points) are densely populated around one or two small regions of interest (one dense cluster 1D and two dense clusters 2D). The one dense cluster 1D and two dense clusters 2D scenarios require relocating all the properties (3-dimensional points) to one or two processes respectively, on which the iterative search is performed locally. When points are sparsely distributed (one sparse cluster 1S and two sparse cluster 2S), GlobalToLocal overhead is still tangible but smaller than for one dense cluster 1D and two dense clusters 2D, i.e., up to one order of magnitude larger than in LocalToGlobal. Note that scenarios like one dense cluster 1D and two dense clusters 2D are scientifically meaningful because the information content of the science strongly converges toward a few conclusions; for GlobalToLocal this is associated to high total times due to the load imbalance and ultimately poor performance. On the other hand, LocalToGlobal performs similarly across scenarios (one dense cluster 1D, one sparse cluster 1S, two dense clusters 2D, and two sparse clusters 2S) and the overheads have lower magnitudes regardless of the embedded scientific results and the physical distributions. When looking at the other time components, both GlobalToLocal and LocalToGlobal total Map times are similar in size; both variations perform similar preprocessing. The GlobalToLocal total Reduce times are two orders of magnitude larger than the equivalent LocalToGlobal times; LocalToGlobal has been simplified to sum single density values (SPAs) rather than counting densities from properties and then summing them up. Dense and sparse clusters have different Shuffle times with dense scenarios taking longer than sparse scenarios. This is because shuffling all the properties to one or two processes for the global analysis, as in the case of dense scenarios, takes more time than shuffling the same properties to a larger subset of processes. The scenarios

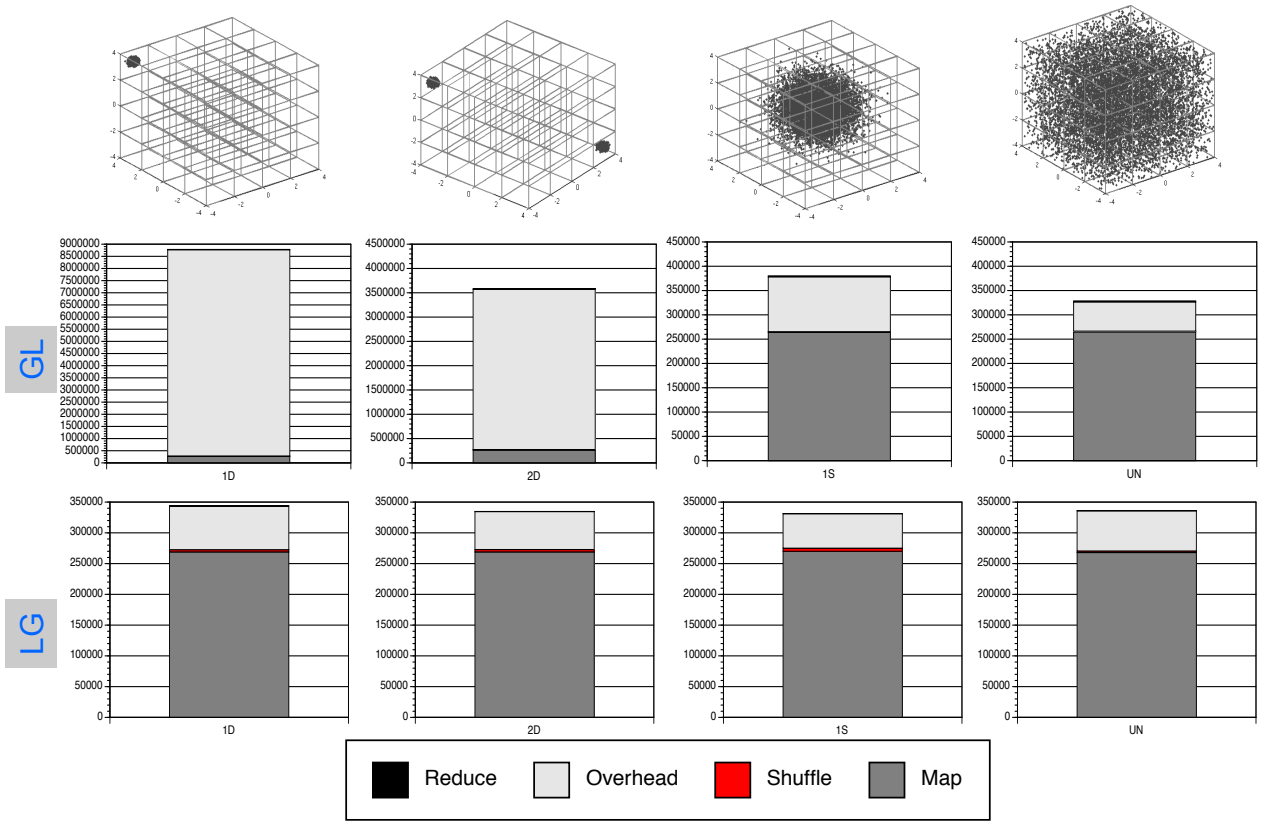


Fig. 3: Total times in seconds across processes broken down into distinctive components: Map, Shuffle, Overhead, Reduce, and Total for GlobalToLocal (GL) and LocalToGlobal(LG). Note that 1D and 2D for GlobalToLocal have different scales.

with a logical uniform distribution always outperform the other scenarios in terms of performance independently from the physical distribution of the data but these scenarios are also the less desirable to work with from the science point of view since they do not convey any conclusion. When comparing the time components across physical distributions, we observe that semi-decentralized scenarios outperform fully-decentralized scenarios in GlobalToLocal and have very similar performance for LocalToGlobal.

To better understand the reasons behind the observed overheads, in Figure 4 we present the typical time pattern per process for the round-robin and random physical distributions; each bar is a collection of 1024 thin bars, one per each process. For each subfigure, each process time in the two bars is normalized with respect to the longest process time for both GlobalToLocal (GL) and LocalToGlobal (LG). This allows us to better compare the two variations. Figure 4 shows that overheads have two components: (1) a much larger component that is present in GlobalToLocal only and is due to load imbalance and (2) a smaller component that is present in both variations and is due to a collective communication embedded into MapReduce-MPI when synchronizing all the processes. This is visible in the figure where thin black lines (corresponding to the Reduce phase) at the top of the bars are observed in one dense cluster 1D and two dense clusters 2D for one and two processes respectively that perform most of the reduce task (i.e., the aggregate operation on the extracted properties). During this time, the remaining processes sit idle until the Reduce step is completed. For the one sparse cluster 1S and two sparse clusters 2S scenarios and GlobalToLocal, we still observe load imbalance (in the form of drifting lines at the top of the bars) but in a smaller proportion compared to one dense cluster 1D and two dense clusters 2D. This time, multiple processes across multiple nodes perform the aggregation operation on more dispersed properties. The synchronization time for the 1024 processes is similar for the two variations, is negligible compared to the other overhead component in GlobalToLocal, and can take up to 20% of the total time in LocalToGlobal. Both Figures 3 and 4 convey the important findings that LocalToGlobal is able to perform implicit load balancing regardless of the logical and physical data distributions.

V. Related work

The MapReduce paradigm has been used in the past to analyze large data. For example, the paradigm was adapted to analyze long MD trajectories in parallel with the tool HiMach [6]. Work in [6] includes the design and use of a MapReduce-style programming interface to exclusively analyze statistical data of long trajectories. Our approach looks at more general similarities in large datasets. Other efforts have investigated well-known clustering methods, such as k-means clustering and hierarchical clustering, which were adapted to fit into the MapReduce framework [7], [8]. However, the resulting

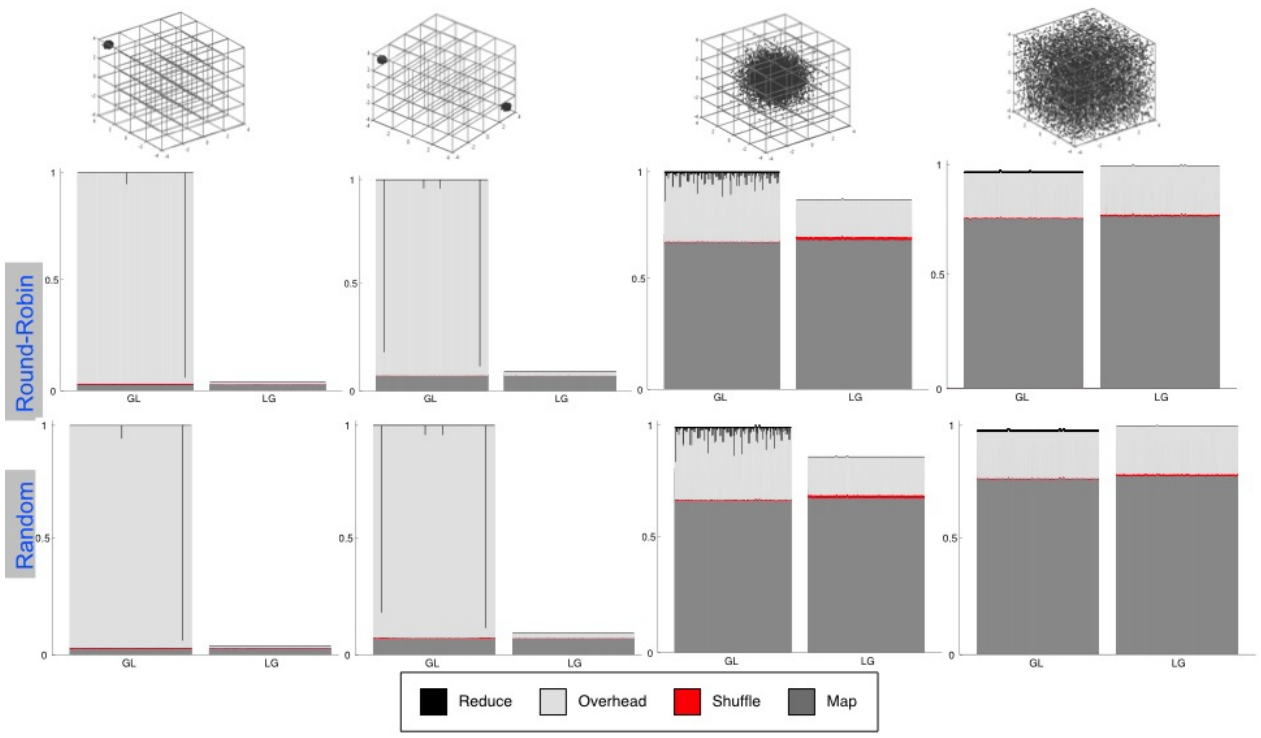


Fig. 4: Per process times normalized w.r.t. the longest process per experiment for the Round-Robin and Random physical distributions.

Uniform										
	1D		1S		UN		2D		2S	
	GL	LG	GL	LG	GL	LG	GL	LG	GL	LG
M	2.64E+05	2.68E+05	2.65E+05	2.68E+05	2.65E+05	2.68E+05	2.65E+05	2.68E+05	2.64E+05	2.69E+05
S	2.35E+04	2.65E+03	1.40E+03	2.74E+03	1.32E+03	1.82E+02	9.88E+03	2.71E+03	2.63E+03	1.56E+03
O	7.77E+06	5.86E+04	1.01E+05	6.18E+04	6.69E+04	5.39E+04	3.07E+06	5.91E+04	1.26E+05	5.45E+04
R	8.63E+03	6.95E+00	1.64E+03	8.38E+00	1.72E+03	1.32E+00	6.50E+03	7.98E+00	2.06E+03	6.43E+00
T	8.06E+06	3.29E+05	3.69E+05	3.33E+05	3.35E+05	3.22E+05	3.35E+06	3.30E+05	3.95E+05	3.25E+05
Round-Robin										
	1D		1S		UN		2D		2S	
	GL	LG	GL	LG	GL	LG	GL	LG	GL	LG
M	2.64E+05	2.69E+05	2.64E+05	2.70E+05	2.65E+05	2.68E+05	2.64E+05	2.69E+05	2.65E+05	2.68E+05
S	1.91E+04	3.65E+03	1.33E+03	5.04E+03	1.47E+03	2.50E+03	1.04E+04	3.85E+03	2.68E+03	3.98E+03
O	8.49E+06	7.10E+04	1.13E+05	5.62E+04	6.02E+04	6.55E+04	3.30E+06	6.18E+04	1.51E+05	6.04E+04
R	9.35E+03	1.06E+01	1.90E+03	1.31E+01	1.93E+03	6.16E+00	7.09E+03	1.11E+01	2.56E+03	1.13E+01
T	8.78E+06	3.43E+05	3.80E+05	3.31E+05	3.28E+05	3.36E+05	3.59E+06	3.35E+05	4.21E+05	3.33E+05
Random										
	1D		1S		UN		2D		2S	
	GL	LG	GL	LG	GL	LG	GL	LG	GL	LG
M	2.66E+05	2.69E+05	2.65E+05	2.69E+05	2.65E+05	2.69E+05	2.66E+05	2.69E+05	2.64E+05	2.69E+05
S	1.94E+04	3.73E+03	1.35E+03	4.92E+03	1.25E+03	2.53E+03	1.02E+04	3.77E+03	3.17E+03	3.98E+03
O	8.63E+06	6.16E+04	1.14E+05	5.72E+04	6.09E+04	6.28E+04	3.62E+06	5.67E+04	1.66E+05	6.28E+04
R	9.52E+03	1.06E+01	2.03E+03	1.30E+01	1.96E+03	6.12E+00	7.81E+03	1.09E+01	2.99E+03	1.07E+01
T	8.93E+06	3.35E+05	3.82E+05	3.32E+05	3.29E+05	3.34E+05	3.90E+06	3.30E+05	4.37E+05	3.36E+05

TABLE I: Total times in seconds across processes broken down into distinctive components: Map (M), Shuffle (S), Overhead (O), Reduce (R), and Total (T).

implementations suffer from the limitations of the clustering algorithms, which do not scale, despite being formulated in the MapReduce paradigm. A similar clustering approach based on the density of single points in an N-dimensional space was presented in the work of Cordeiro el al. [9]. The three algorithms presented in Cordeiro's work rely on local clustering of sub-regions and the merging of local results into a global solution (which can potentially suffer from accuracy issues), whereas our proposed approach considers the whole dataset and performs a single-pass analysis on it. Contrary to [9], when using the LG variation we no longer observe a correlation between space density and clustering efficiency. Milletti and Vulpetti [10] perform a geometry-based abstraction of molecular structures. Specifically, they profile protein-binding pockets by using a shape-context-based method. This method describes the coarse distribution of the

pocket shape with respect to a given point in its geometry. The shape-context method is linear in the number of atoms considered to generate a pocket descriptor and to perform the inhibition map prediction. However, the overall methodology presented in [10] is not practical for the clustering of other datasets. To the best of our knowledge, our approach is the first to emphasize a local, single pass of data to extract global properties or densities and, by doing so, to avoid major data movements. In our work, scalability plays a key role and is particularly well supported by the LG variant. Our approach expands our previous work [1]. Here we use MapReduce-MPI rather than Hadoop to move away from the overhead of the Hadoop Distributed File System (HDFS). HDFS acts as the central storage space for input and output data, adding additional space and operation time to move in and out data. We also extend the previous algorithm into two variations for fully distributed environments. In the variations, data no longer needs to be moved a priori with HDFS, making our approach completely scalable. These variations allow us to study the performance impact of exchanging extracted properties in contrast to exchanging property densities; this was not achievable in Hadoop due to the coarse grain control on data placement of HDFS.

VI. Conclusions and Future Work

This paper focuses on avoiding data movement when analyzing scientific datasets composed of a large number of complex data records distributed across multiple nodes of a large distributed memory system. We present two variations of a general algorithm for data analysis (i.e., clustering or classification) and apply it to structural biology datasets. Both variations perform a local, single preprocessing pass of the data to extract relevant properties, i.e., the geometry of a ligand conformation in a large dataset of 100 million confirmations. In the GlobalToLocal variation we exchange extracted properties rather than data, in the LocalToGlobal variation, we exchange scalar property aggregates. The latter are usually smaller than properties but require a higher number of exchanges.

We evaluate the performance of the two variations on 1024 cores under fifteen different scenarios by combining five different clustering configurations and three different physical distributions of the data. From the performance point of view, we observe that GlobalToLocal is very sensitive to the information content of the analyzed datasets, while LocalToGlobal is not. We observe that when exchanging scalar property aggregate, our approach delivers scalable performance and is largely insensitive to the contents of the datasets. In other words, the time to perform analyses and extract the information in the dataset is not impacted by the number of clusters, their shapes, or the way the data is distributed across the local storage. We also observe that by moving the computation to the data location as in LocalToGlobal, any load imbalance resulting from the way the scientific content of datasets are randomly generated and distributed is minimized. Hence, LocalToGlobal overall outperforms GlobalToLocal. Our variations are applicable to diverse scientific datasets beyond the one considered in this paper and can result in scientific analyses that are not sensitive to data content and distribution.

Future work includes adapting our methodology to capture a broader set of properties in large datasets from other scientific domains such as astronomy for which we want to analyze spectra to classify galaxy in terms of ages. We believe that while we here demonstrated our approach for one specific dataset in structural biology and implemented in MapReduce-MPI, it is applicable to a many scientific analyses and other distributed frameworks.

Acknowledgment

This work is supported in part by NSF grants: DMS 0800272/0800266, IIS 0968350, and CNS 1318417. The authors want to thank Dr. Craig A. Stewart from Indiana University for his valuable comments on our work and manuscript. All the datasets will be made available before the conference.

References

[1] T. Estrada, B. Zhang, P. Cicotti, R. Armen, and M. Taufer, "Accurate analysis of large datasets of protein-ligand binding geometries using advanced clustering methods," *Computers in Biology and Medicine*, vol. 42, no. 7, pp. 758–771, 2012.

[2] J. Dean and S. Ghemawat, "MapReduce: simplified data processing on large clusters," *Communications of the ACM*, vol. 51, no. 1, pp. 107–113, 2008.

[3] S. J. Plimpton and K. D. Devine, "MapReduce in MPI for large-scale graph algorithms," *Parallel Computing*, vol. 37, no. 9, Sep. 2011.

[4] A. Jain, "Bias, reporting, and sharing: Computational evaluations of docking methods," *Journal of Computer-Aided Molecular Design*, vol. 22, no. 3-4, pp. 201–212, 2008.

[5] S. S. Shende and A. D. Malony, "The TAU parallel performance system," *International Journal of High Performance Computing Applications*, vol. 20, no. 2, pp. 287–311, 2006.

[6] T. Tu, C. A. Rendleman, D. W. Borhani, R. O. Dror, J. Gullingsrud, M. O. Jensen, J. L. Klepeis, P. Maragakis, P. Miller, K. A. Stafford, and S. E. Shaw, "A scalable parallel framework for analyzing terascale molecular dynamics simulation trajectories," in *Proceedings of the 2008 ACM/IEEE Conference on Supercomputing (SC)*, 2008, pp. 1–12.

[7] H. G. Li, G. Q. Wu, X. G. Hu, J. Zhang, L. Li, and X. Wu, "K-means clustering with bagging and mapreduce," in *Proceedings of the 44th Hawaii International Conference on System Sciences (HICSS)*, 2011, pp. 1–8.

[8] A. Ene, S. Im, and B. Moseley, "Fast clustering using mapreduce," in *Proceedings of the 17th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD)*, 2011, pp. 681–689.

[9] R. L. F. Cordeiro, C. T. Jr, A. J. M. Traina, J. López, U. Kang, and C. Faloutsos, "Clustering very large multi-dimensional datasets with MapReduce," in *Proceedings of the 17th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD)*, 2011, pp. 690–698.

[10] F. Milletti and A. Vulpetti, "Predicting polypharmacology by binding site similarity: From kinases to the protein universe," *Journal of Chemical Information and Modeling*, vol. 50, no. 8, pp. 1418–1431, 2010.

Appendix

Table I shows the total time in seconds of GlobalToLocal (GL) and LocalToGlobal (LG) across fifteen experiments - i.e., for the combination of five logical distributions (i.e., one dense cluster '1D', one sparse cluster '1S', uniform 'UN', two dense clusters '2D', and two sparse clusters '2S') and three physical distributions (i.e., Uniform, Round-Robin, and Random).