Distributed Change Point Detection for Mining Astronomy Data Streams

Kanishka Bhaduri*, Kamalika Das†, Kirk Borne‡, Chris Giannella§, Tushar Mahule¶, Hillol Kargupta¶‖

*Mission Critical Technologies Inc.,

NASA Ames Research Center, MS 269-1, Moffett Field, CA-94035

Email:Kanishka.Bhaduri-1@nasa.gov

†Stinger Ghaffarian Technologies Inc.,

NASA Ames Research Center, MS 269-3, Moffett Field, CA-94035

Email:Kamalika.Das@nasa.gov

‡Computational and Data Sciences Dept., GMU, VA-22030

Email:kborne@gmu.edu

§The MITRE Corporation

300 Sentinel Dr. Suite 600, Annapolis Junction MD 20701

Email:cgiannel@acm.org

¶CSEE Department, UMBC

1000 Hilltop Circle, Baltimore, Maryland, 21250, USA

Email:{tusharm1,hillol}@cs.umbc.edu

‖AGNIK, LLC., Columbia, MD, USA

A shorter version of this paper was published in SIAM Data Mining Conference 2009

Abstract

This paper considers the problem of change detection using local distributed eigen monitoring algorithms for next generation of astronomy petascale data pipelines such as the Large Synoptic Survey Telescopes (LSST). This telescope will take repeat images of the night sky every 20 seconds, thereby generating 30 terabytes of calibrated imagery every night that will need to be co-analyzed with other astronomical data stored at different locations around the world. Change point detection and event classification in such data sets may provide useful insights to unique astronomical phenomenon displaying astrophysically significant variations: quasars, supernovae, variable stars, and potentially hazardous asteroids. However, performing such data mining tasks is a challenging problem for such high-throughput distributed data streams. In this paper we propose a highly scalable and distributed asynchronous algorithm for monitoring the principal components (PC) of such dynamic data streams and discuss a prototype web-based system PADMINI (Peer to Peer Astronomy Data Mining) which implements this algorithm for use by the astronomers. We demonstrate the algorithm on a large set of distributed astronomical data to accomplish well-known astronomy tasks such as measuring variations in the fundamental plane of galaxy parameters. The proposed algorithm is provably correct (i.e. converges to the correct PCs without centralizing any data) and can seamlessly handle changes to the data or the network. Real experiments performed on Sloan Digital Sky Survey (SDSS) catalogue data show the effectiveness of the algorithm.

I. Introduction

Data mining is playing an increasingly important role in astronomy research [28][12][4] involving very large sky surveys such as Sloan Digital Sky Survey SDSS and the 2-Micron All-Sky Survey 2MASS. These sky-surveys are offering a new way to study and analyze the behavior of the astronomical objects. The next generation of sky-surveys are poised to take a step further by incorporating sensors that will stream in large volume of data at a high rate. For example, the Large Synoptic Survey Telescopes (LSST) will take repeat images of the night sky every 20 seconds. This will generate 30 terabytes of calibrated imagery every night that will need to be co-analyzed with other astronomical data stored at different locations around the world. Change point detection and event classification in such data sets may provide useful insights to unique astronomical phenomenon displaying astrophysically significant variations: quasars, supernovae, variable stars, and potentially hazardous asteroids. Analyzing such high-throughput data streams would require large distributed computing environments for offering

scalable performance. The knowledge discovery potential of these future massive data streams will not be achieved unless novel data mining and change detection algorithms are developed to handle decentralized petascale data flows, often from multiple distributed sensors (data producers) and archives (data providers). Several distributed computing frameworks such as [19], [25], [26], [22], and [17] are being developed for such applications. We need distributed data mining algorithms that can operate on such distributed computing environments. These algorithms should be highly scalable, be able to provide good accuracy and should have a low communication overhead.

This paper considers the problem of change detection in the spectral properties of data streams in a distributed environment. It offers an asynchronous, communication-efficient distributed eigen monitoring (DDM) algorithm for monitoring the principle components (PCs) of dynamic astronomical data streams. It particularly considers an important problem in astronomy regarding the variation of fundamental plane structure of galaxies with respect to spatial galactic density and demonstrates the power of DDM algorithms using this example application. This paper presents the algorithm, analytical findings, and results from experiments. Experiments are performed using currently available astronomy data sets from virtual observatories. Our distributed algorithm is a first step in analyzing the astronomy data arriving from such high throughput data streams of the future. The specific contributions of this paper can be summarized as follows:

- To the best of the authors knowledge this is one of the first attempts on developing a completely asynchronous and local algorithm for doing eigen analysis in distributed data streams
- Based on data sets downloaded from astronomy catalogues such as SDSS and 2MASS, we demonstrate how the galactic fundamental plane structure varies with difference in galactic density
- We discuss the architecture, workflow and deployment of an entirely web-based P2P astronomy data mining prototype (PADMINI) that allows astronomers to perform event detection and analysis using P2P data mining technology

Section II describes the astronomy problem. Section III discusses the data mining prototype system PADMINI while Section IV presents the related work. Section V offers the background material and formulates the data mining problem. Section VI describes the centralized version

of the problem while Section VII models the distributed version and explains the eigenstate monitoring algorithm. Section IX presents the experimental results. Finally, Section X concludes this paper.

II. Astronomy Data Stream Challenge Problems

When the LSST astronomy project becomes operational within the next decade, it will pose enormous petascale data challenges. This telescope will take repeat images of the night sky every 20 seconds, throughout every night, for 10 years. Each image will consist of 3 gigapixels, yielding 6 gigabytes of raw imagery every 20 seconds and nearly 30 terabytes of calibrated imagery every night. From this "cosmic cinematography", a new vision of the night sky will emerge – a vision of the temporal domain – a ten-year time series (movie) of the Universe. Astronomers will monitor these repeat images night after night, for 10 years, for everything that has changed – changes in position and intensity (flux) will be monitored, detected, measured, and reported. For those temporal variations that are novel, unexpected, previously unknown, or outside the bounds of our existing classification schemes, astronomers will want to know (usually within 60 seconds of the image exposure) that such an event (a change in the night sky) has occurred. This event alert notification must necessarily include as much information as possible to help the astronomers around the world to prioritize their response to each time-critical event. That information packet will include a probabilistic classification of the event, with some measure of the confidence of the classification. What makes the LSST so incredibly beyond current projects is that most time-domain sky surveys today detect 5-10 events per week; LSST will detect 10 to 100 thousand events per night! Without good classification information in those alert packets, and hence without some means with which to prioritize the huge number of events, the astronomy community would consequently be buried in the data deluge and will miss some of the greatest astronomical discoveries of the next 20 years (perhaps even the next "killer asteroid" heading for Earth – this time, it won't be the dinosaurs that will go extinct!).

To solve the astronomers' massive event classification problem, a collection of high-throughput change detection algorithms will be needed. These algorithms will need to access distributed astronomical databases worldwide to correlate with each of those 100,000 nightly events, in order to model, classify, and prioritize correctly each event rapidly. One known category of temporally varying astronomical object is a variable star. There are dozens of different well known classes of

variable stars, and there are hundreds (even thousands) of known examples of these classes. These stars are not "interesting" in the sense that they should not produce alerts (change detections), even though they are changing in brightness from hour to hour, night to night, week to week – their variability is known, well studied, and well characterized already. However, if one of these stars' class of variability were to change, that would be extremely interesting and be a signal that some very exotic astrophysical processes are involved. Astronomers will definitely want to be notified promptly (with an alert) of these types of variations. Just what is this variation? It is essentially a change in the Fourier components (eigenvectors) of the temporal flux curve (which astronomers call "the light curve").

This problem has several interesting data challenge characteristics: (1) the data streaming rate is enormous (6 gigabytes every 20 seconds); (2) there are roughly 100 million astronomical objects in each of these images, all of which need to monitored for change (i.e., a new variable object, or a known variable object with a new class of variability); (3) 10 to 100 thousand "new" events will be detected each and every night for 10 years; and (4) distributed data collections (accessed through the Virtual Astronomy Observatory's worldwide distribution of databases and data repositories) will need to correlated and mined in conjunction with each new variable object's data from LSST, in order to provide the best classification models and probabilities, and thus to generate the most informed alert notification messages.

Astronomers use sky surveys to study the sky systematically by measuring and collecting data from all objects that are visible within large regions of the sky, in a systematic, controlled, and repeatable fashion. These statistically robust procedures thereby generate very large unbiased samples of numerous classes of astronomical objects. Many astronomy data mining use cases are anticipated with the LSST database [9][10][11], including:

- Provide rapid probabilistic classifications for all 10,000–100,000 LSST events each night;
- Find new "fundamental planes" of parameters (*e.g.*, the fundamental plane of Elliptical galaxies);
- Find new correlations and associations of all kinds from 200+ attributes in the science database;
- Compute N-point correlation functions over a variety of spatial and cosmological parameters;
- Discover voids or zones of avoidance in multi-dimensional parameter spaces (*e.g.*, period

gaps);

- Discover new and exotic classes of astronomical objects, while discovering new properties of known classes;
- Discover new and improved rules for classifying known classes of objects (*e.g.*, photometric redshifts);
- Identify novel, unexpected behavior in the time domain from time series data;
- Hypothesis testing - verify existing (or generate new) astronomical hypotheses with strong statistical confidence, using millions of training samples;
- Serendipity - discover the rare one-in-a-billion type of objects through outlier detection; and
- Quality assurance - identify glitches and image processing errors through deviation detection.

We are developing and testing algorithms that specifically support many of these scientific use cases.

Consider this additional example: astronomers currently discover $\sim$100 new supernovae (exploding stars) per year. Since the beginning of human history, significantly fewer than 10,000 supernovae have been recorded. The identification, characterization, classification, and analysis of supernovae are among the key science requirements for LSST's, NASA's, and DOE's plans to explore the ubiquitous cosmic Dark Energy. Since supernovae are the result of a rapid catastrophic explosion of a massive star, it is imperative for astronomers to respond quickly (within minutes) to each new event with rapid follow-up observations in many measurement modes (e.g., light curves, spectroscopy, or imaging of the host galaxy). Historically, with less than 10 new supernovae being discovered each week, such follow-up observations have been feasible. However, LSST promises to produce a list of 1000 new supernovae each night for 10 years! How can such a staggering number of new objects be efficiently detected, characterized, and properly prioritized for follow-up observation? Astronomers will be faced with the enormous challenge of efficient change detection (within 60 seconds) relative to information stored in various geographically distributed databases for all 100 million objects in each image, and then to characterize specifically and correctly those changes and behaviors that help to identify the supernovae from among the 100,000 nightly events. Rapidly mining, correctly classifying, and intelligently prioritizing this huge number of new events for rapid follow-up observation each

night for a decade is an enormous data stream challenge problem for astronomy.

Astronomers cannot wait until the year 2016 (when LSST begins its sky survey operations) for new algorithms to begin being researched. Those algorithms (for distributed mining, change detection, and eigenvector monitoring) will need to be robust, scalable, and validated already at that time. So, it is imperative to begin now to research, test, and validate such data mining paradigms through experiments that replicate the expected conditions of the LSST sky survey. Consequently, we have chosen an astronomical research problem that is both scientifically valid (i.e., a real astronomy research problem today) and that parallels the eigenvector monitoring problem that we have described above. We have chosen to study the principal components of galaxy parameters as a function of an independent variable, similar to the temporal dynamic stream mining described above. In our current experiments, the independent variable is not the time dimension, but local galaxy density. We specifically investigate this problem because it is scientifically current and interesting, thereby producing new astronomical research results, and also because it performs tests of our algorithms specifically on the same types of distributed databases that will be used in the future LSST change detection and event classification problems

The class of elliptical galaxies has been known for 20 years to show dimension reduction among a subset of physical attributes, such that the 3-dimensional distribution of three of those astrophysical parameters reduce to a 2-dimensional plane. The normal to that plane represents the principal eigenvector of the distribution, and it is found that the first two principal components capture significantly more than 90% of the variance among those 3 parameters.

By analyzing existing large astronomy databases (such as the Sloan Digital Sky Survey SDSS and the 2-Micron All-Sky Survey 2MASS), we have generated a very large data set of galaxies. Each galaxy in this large data set was then assigned (labeled with) a new ”local galaxy density” attribute, calculated through a volumetric Voronoi tessellation of the total galaxy distribution in space. Then the entire galaxy data set was horizontally partitioned across several dozen partitions as a function of our independent variable: the local galaxy density.

As a result, we have been able to study eigenvector changes of the fundamental plane of elliptical galaxies as a function of density. Computing these eigenvectors for a very large number of galaxies, one density bin at a time, in a distributed environment, thus mimics the future LSST dynamic data stream mining change detection (eigenvector change) challenge problem described earlier. In addition, this galaxy problem actually has uncovered some new astrophysical results:

we find that the variance captured in the first 2 principal components increases systematically from low-density regions of space to high-density regions of space, and we find that the direction of the principal eigenvector also drifts systematically in the 3-dimensional parameter space from low-density regions to the highest-density regions.

III. PADMINI–A PEER-TO-PEER ASTRONOMY DATA MINING SYSTEM

PADMINI is a web based Peer to Peer data mining system that aims at being a computation tool for the researchers and users related to the field of astronomy and data mining. There are several challenges to centralizing the massive astronomy catalogs (some of which has been elucidated in the previous section) and running traditional data mining algorithms. To solve this data avalanche, PADMINI is powered by a back end peer to peer computation network to provide the required scalability. The back end computation network supports two distributed computation frameworks, namely Distributed Data Mining Toolkit (DDMT) [20] and Hadoop [27]. The web based PADMINI system is available online at `http://padmini.cs.umbc.edu/`. In the next few sections we first describe the different components of the system and then describe the implementation details.

A. System components

The system architecture is shown in Figure 1. It consists of a web server, DDM server, server database, jobs database and the back-end P2P network. Each of the components are discussed in details next.

1) Web server: The web server hosts the main website and is the primary interface for submitting jobs and retrieving results of the submitted jobs (see Figure 2). Each new user signs up for an account on the website and sets up a job to be run on the system. Every user has a dedicated profile page where the user can keep a track of the jobs submitted by him. The current status of the jobs and a projected time for the completion of the jobs are also displayed on the same page. Each job submitted by the user will trigger a distributed algorithm to run on the back-end P2P network. The results of the algorithm will be pushed back to the web server. The user can then download a copy of the results of their jobs. The web service methods exposed by the DDM server are used by the web server to start a job and receive results. The web server is thus the consumer of the web service methods exposed by the DDM server.

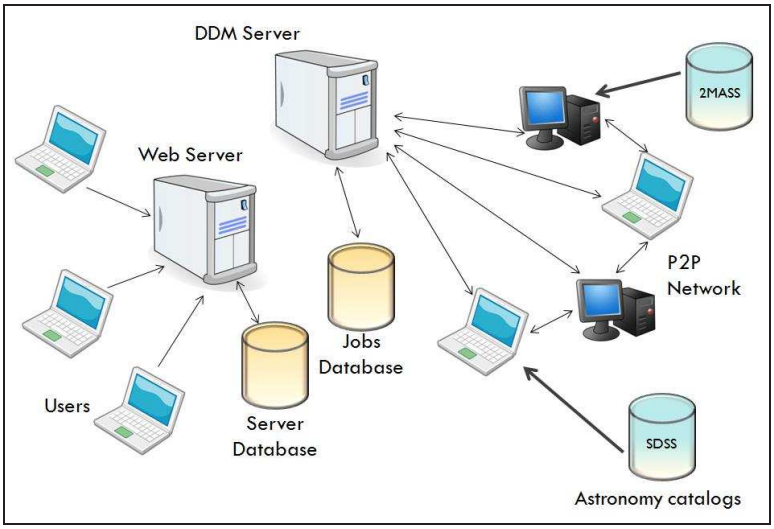


Fig. 1. System diagram showing the different components

Figure 3 shows how a user can select a portion of the sky and set it up as an input for a job. The website provides access to some privileged tasks to the administrator. These tasks include setting up the network, adding or deleting nodes from the network, etc. The administrator page is shown in Figure 4.

Fig. 2. Home page

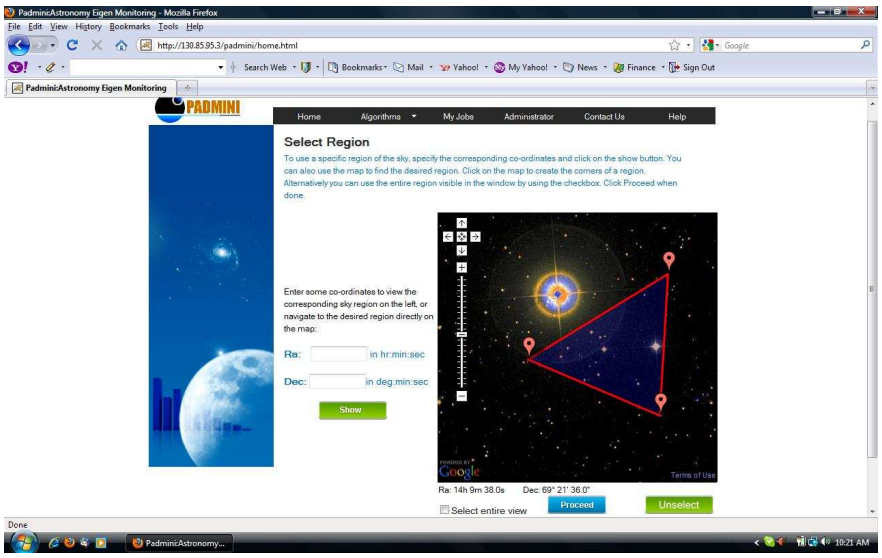


Fig. 3. Astronomy eigen monitoring - selecting a sky region

2) Server database: The server database primarily deals with user and identity management. The database stores the information related to the registered users of the system and the privileges they have. The job activity details of an user are also stored in this database. These include the inputs submitted by the user, the algorithm selected, the output of the job etc. The list of the supported astronomy data catalogs and their attributes that a user can use as inputs are also stored in this database.

3) DDM server: The Distributed Data Mining (DDM) Server is an intermediate tier between the web server and the back-end peer to peer computation network. The multiple job requests coming in from the web server are directed to the DDM Server and stored in a job queue. The jobs are then submitted serially for completion to the back-end computation network. The DDM server exposes a set of methods that can be used to set up jobs and for pushing the results of the completed jobs onto the web server. The web service methods encourage openness. Hence, a new system can be easily built around the available back-end P2P computation network.

4) Jobs database: The jobs database persists the book-keeping information related to the jobs. This includes the list of all the jobs that are submitted by the user, including the ones not yet submitted to the computation network. The status of the running and the waiting jobs and the results of the recently completed jobs is stored here. The database also stores information

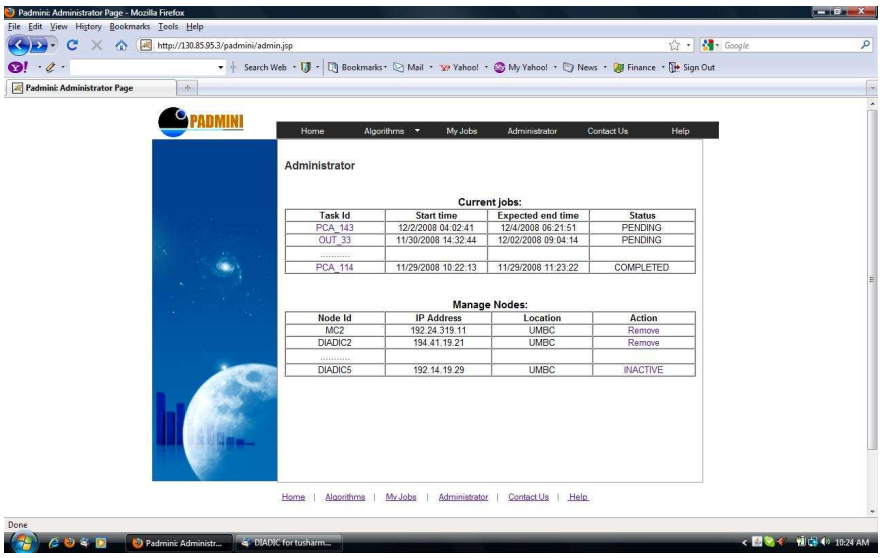


Fig. 4. Administrator page

related to the back-end P2P computation network. This includes the information pertaining to the total number of active nodes, failed nodes etc.

5) P2P network: The peer to peer network forms the backbone of the back-end computation framework. All the peers in this network are configured to support two computation frameworks, namely Distributed Data Mining Toolkit (DDMT) and Hadoop. The type of jobs the user can submit is restricted by the algorithms supported by the system. Some algorithms are implemented using the DDMT while some are built on top of the Hadoop framework. The DDM server picks up a job from the queue and assigns it to be executed on top of the appropriate framework. This information depends on the type of the job and hence is implicitly set by the user.

B. Implementation details

1) Language: The website is developed using HTML, Javascript and JSPs. DDMT is implemented in Java and is based on Java Agent Development (JADE) Framework. The important methods like starting a job, stopping it, providing input etc. have been exposed as web service methods. This enables future systems to be built around the existing computation network. Hadoop provides an extensive Java API using which highly scalable Map Reduce algorithms can be created. For running either DDMT or Hadoop, Java support is the only expected feature from a peer. Thus, the P2P computation network can be easily expanded.

2) Databases: MySQL is used as the database in the web server database as well as in the jobs database. Hibernate is used for object-relational mapping at the web server database end. Classes corresponding to the database tables make sure that operations made on the class objects get reflected and persisted in the database. Such a system not only saves development time, but also guarantees a robust database system.

3) Web service: Axis2 is used as the core engine for web services. Axis2 is built on a new architecture that was designed to be a much more flexible, efficient, and configurable. With the new Object Model defined by Axis2, it is easier to handle SOAP messages. All the web service requests are directed to the DDM Server. The DDM Server then calls the corresponding methods and starts the requested job. Axis2 also has excellent support for sending binary data or files using SOAP messages. This eases moving the inputs and outputs between the web server and the DDM server.

4) User interface: An user needs to sign up on the home page to get an account and start submitting jobs. On signing up, each user gets a personal profile page. Each algorithm supported by the website has a dedicated page on which the user can create and submit a specific job. The user can then track the status of the submitted jobs and also store the results of the most recently completed jobs on the profile page. The Google Maps interface on the PADMINI website aids an astronomer in specifying an area of the sky intuitively and effectively. The controls to select the astronomy catalogs and the supported attributes are also provided. Thus, a job can be specified with only a few clicks and the user does not to need to wait for the results.

In the next section we present the related work.

IV. Related Work

The work related to this area of research can broadly be subdivided into data analysis for large scientific data repository and distributed data mining in a dynamic networks of nodes. We discuss each of them in the following two sections.

A. Analysis of Large Scientific Data Collections

The U.S. National Virtual Observatory (NVO) [37] is a large scale effort to develop an information technology infrastructure enabling easy and robust access to distributed astronomical archives. It will provide services for users to search and gather data across multiple archives

and will provide some basic statistical analysis and visualization functions. The International Virtual Observatory Alliance (IVOA) [30] is the international steering body that federates the work of about two dozen national VOs across the world (including the NVO in the US). The IVOA oversees this large-scale effort to develop an IT infrastructure enabling easy and robust access to distributed astronomical archives worldwide.

There are several instances in the astronomy and space sciences research communities where data mining is being applied to large data collections [19][16][2]. Another recent area of research is distributed data mining [33][31] which deals with the problem of data analysis in environments with distributed data, computing nodes, and users. Distributed eigen-analysis and outlier detection algorithms have been developed for analyzing astronomy data stored at different locations by Dutta *et al.*[23]. Kargupta *et al.* [32] have developed a technique for performing distributed principal component analysis by first projecting the local data along its principal components and then centralizing the projected data. In both these cases, the data is distributed vertically (different full attribute columns reside at different sites), while in this paper, the data is distributed horizontally (different data tuple sets reside at different sites). Moreover, none of the above efforts address the problem of analyzing rapidly changing astronomy data streams.

B. Data Analysis in Large Dynamic Networks

There is a significant amount of recent research considering data analysis in large-scale dynamic networks. Since efficient data analysis algorithms can often be developed based on efficient primitives, approaches have been developed for computing basic operations (*e.g.* average, sum, max, random sampling) on large-scale, dynamic networks. Kempe *et al.* [34] and Boyd *et al.* [13] developed gossip based randomized algorithms. These approaches used an epidemic model of computation. Bawa *et al.* [5] developed an approach based on probabilistic counting. In addition, techniques have been developed for addressing more complex data mining/data problems over large-scale dynamic networks: association rule mining [41], facility location [35], outlier detection [14], decision tree induction [8], ensemble classification [36], support vector machine-based classification [1], k-means clustering [18], top-k query processing [3].

A related line of research concerns the monitoring of various kinds of data models over large numbers of data streams. Sharfman *et al.* [39] develop an algorithm for monitoring arbitrary threshold functions over distributed data streams. And, most relevant to this paper, Wolff *et al.*

[40] developed an algorithm for monitoring the L2 norm. We use this technique to monitor eigen-states of the fundamental plane concerning elliptical galaxies.

Huang *et al.* [29] consider the problem of detecting network-wide volume anomalies via thresholding the length of a data vector (representing current network volume) projected onto a subspace closely related to the dominant principal component subspace of past network volume data vectors. Unlike us, these authors consider the analysis of a vertically distributed data set. Each network node holds a sliding window stream of numbers (representing volume through it with time) and the network-wide volume is represented as a matrix with each column a node stream. Because of the difference in data distribution (vertical vs. horizontal), their approach is not applicable to our problem. We assume that each node is receiving a stream of tuples and the network-wide dataset is matrix formed by the union of all nodes' currently held tuples (each node holds a collection of *rows* of the matrix rather than a single *column* as considered by Huang).

In the next section we present the notations and problem definition that will be used throughout the rest of the paper.

V. Background

In order to analyze the data streams from the next generation of large scale astronomy systems such as the ones constructed by the LSST project, we need scalable infrastructure for computing. It is generally agreed among the astronomy community that the computing infrastructure will be a grid-like environment comprised of a collection of desktop compute nodes, high performance machines, and data sources among others. We need data analysis algorithms that will be able to work in this distributed heterogeneous computing environment. This paper offers distributed eigen-analysis algorithms that can handle data from distributed nodes (either inherently distributed data or artificially distributed in order to scale up the performance).

In the remainder of this section we first define the notations that will be used to discuss our distributed algorithms and then formally state the problem definition.

A. Notations

Let $V = \{P_1, \ldots, P_n\}$ be a set of nodes connected to one another via an underlying communication infrastructure such that the set of P_i's neighbors, Γ_i, is known to P_i. Additionally, at any

time, P_i is given a time-varying data matrix $\mathcal{M}_i$ where the rows correspond to the instances and the columns correspond to attributes or features. Mathematically, $\mathcal{M}_i = [\overrightarrow{x_{i,1}}\overrightarrow{x_{i,2}}\dots]^{\mathrm{T}}$, where each $\overrightarrow{x_{i,\ell}} = [x_{i,\ell 1}x_{i,\ell 2}\dots x_{i,\ell d}] \in \mathbb{R}^d$ is a row vector. The covariance matrix of the data at node P_i, denoted by $\mathcal{C}_i$, is the matrix whose (i,j)-th entry corresponds to the covariance between the i-th and j-th feature (column) of $\mathcal{M}_i$. The global data set of all the nodes' data is $\mathcal{G} = \bigcup_{i=1}^{n} \mathcal{M}_i$.

It can be shown that if the attributes of $\mathcal{G}$ are mean shifted, *i.e.* the mean of each attribute is subtracted from each value of that attribute, the covariance matrix can be written as $\mathcal{C} = \mathcal{G}^{\mathrm{T}}\mathcal{G}$ (we have ignored the scaling by the number of points in $\mathcal{G}$). Also under such conditions, it can be shown that $\mathcal{C} = \sum_{i=1}^{n} \mathcal{C}_i$.

B. Problem Formulation

The identification of certain correlations among parameters has lead to important discoveries in astronomy. For example, the class of elliptical and spiral galaxies (including dwarfs) have been found to occupy a two dimensional space inside a 3-D space of observed parameters, radius, mean surface brightness and velocity dispersion. This 2D plane has been referred to as the Fundamental Plane [24]. In this paper we first describe a PCA computation for detecting variation of fundamental plane with galactic properties such as density and then develop an asynchronous local distributed algorithm for monitoring the eigenvectors of the global covariance matrix ($\mathcal{C}$) of the data. As we discuss in the next section, eigenvectors of the covariance matrix define the fundamental plane of the galaxies.

The problem that we want to solve in this paper can be stated as follows:

Problem Statement: Given a time-varying data matrix $\mathcal{M}_i$ to each node, maintain an up-to-date set of eigenvectors and eigenvalues of the global covariance matrix $\mathcal{C}$ at any time instance.

Typically, we have the following constraints:

- low communication overhead
- dynamic data and topology
- correct result compared to centralized execution

Given this problem statement, we decompose it into two parts: (i) first, given an estimate of the eigenvectors and eigenvalues, we discuss a highly efficient and local algorithm for checking

the 'fitness' of the model to the global data, and (ii) if the data changes to the extent that the current estimates are outdated, we sample data from the network and rebuild the model.

In the next section we show how we have collected and preprocessed the astronomy catalogue data for fundamental plane computation.

VI. Centralized Principal Components Analysis for the Fundamental Plane Computation

For identifying the variability of fundamental plane on the basis of galactic densities, we have used the SDSS and 2MASS data sets available individually through the NVO. Since galactic density is not observed by the NVOs, we have cross-matched the two data sets and computed the densities based on other property values. In this section we describe the data gathering procedure for this approach followed by the PCA computation.

A. Data Preparation

We create a large, aggregate data set by downloading the 2MASS XSC extended source catalog (`http://irsa.ipac.caltech.edu/applications/Gator/`) for the entire sky and cross-match it against the SDSS catalog using the SDSS Crossid tool (`http://cas.sdss.org/astro/en/tools/crossid/upload.asp`) such that we select all unique attributes from the *PhotoObjAll* and *SpecObjAll* tables as well as the *photozd1* attribute from the *Photoz2* table which is an estimated redshift value. We filter the data based on the SDSS identified type to remove all non-galaxy tuples. We then filter the data again based on reasonable redshift (actual or estimated) values ($0.003 \leq z \leq 0.300$).

For creating the new attribute, namely, galactic density, we transform the attributes cx, cy, cz (unit vectors), z, and $photozd1$ to 3D Euclidean coordinates using the transformation

$$(X, Y, Z) = (Distance \times cx, Distance \times cy, Distance \times cz)$$

where $Distance = 2 \times \left[1 - \frac{1}{\sqrt{(1+redshift)}}\right]$. We finally use these Cartesian coordinates to compute the Delaunay Triangulation [21] of each point (galaxy) in 3D space. To remove bias in the density calculation of the Delaunay cells, we identify all boundary points and remove them from the computation. This tessellation procedure is a data transformation step, which converts the spatial location of a galaxy (within the 3-D distribution of galaxies) into a single numeric

attribute (local galaxy density). This parameter has astrophysical significance, even more than the actual spatial location information (*i.e.*, galaxy properties are often modified and governed by the proximity of nearby galaxies, such as in high-density environments), and so we chose to use this new attribute — local galaxy density, as estimated through the tessellation step — because it has strong astrophysical relevance and scientific significance. This is a robust estimator and is as scientifically meaningful as any other attribute in the science database used in these experiments. Now using the output of the Delaunay triangulation the volumes of the Delaunay cells are computed using the expression

$$vol(i) = (1/6) \cdot \left| det(\overrightarrow{a_i} - \overrightarrow{b_i}, \overrightarrow{b_i} - \overrightarrow{c_i}, \overrightarrow{c_i} - \overrightarrow{h_i}) \right|,$$

where $\overrightarrow{a_i}, \overrightarrow{b_i}, \overrightarrow{c_i}$ and $\overrightarrow{h_i}$ are the vertices of the tetrahedron corresponding to the i-th point in 3D Euclidean space. The volume corresponding to the i-th point is the sum of the volumes of all tetrahedrons that contain the particular point. Using the Delaunay Tessellation Field Estimator (DTFE) formulation [38], the density of the i-th cell is then computed as follows:

$$den(i) = (D+1) \times \frac{m_i}{vol(i)}$$

where $m_i = 1$, since we have one object (galaxy) per cell and $D = 3$ for triangulation in 3D-space .

B. Binning and PCA

The astronomy question that we want to address here is whether the fundamental plane structure of galaxies in low density regions differ from that of galaxies in high density regions. For this we take the above data set containing 155650 tuples and associate with each tuple, a measure of its local galactic density. Our final aggregated data set has the following attributes from SDSS: Petrosian I band angular effective radius (*Iaer*), redshift (*rs*), and velocity dispersion (*vd*); and has the following attribute from 2MASS: K band mean surface brightness (*Kmsb*). We produce a new attribute, logarithm Petrosian I band effective radius (*log(Ier)*), as *log(Iaer*rs)* and a new attribute, logarithm velocity dispersion (*log(vd)*), by applying the logarithm to *vd*. We finally append the galactic density (*cellDensity*) associated with each of the tuples as the last attribute of out aggregated data set. We divide the tuples into 30 bins based on increasing cell density, such that there are equal number of tuples in each bin. For each bin we carry

out the fundamental plane calculation or principal component analysis and observe that the percent of variance captured by the first two PCs is very high. This implies that the galaxies can be represented by the plane defined by the first two eigen vectors. It is also observed that this percentage increases for bins with higher mean galactic density. We report these results in Section IX.

As discussed earlier, analysis of very large astronomy catalogs can pose serious scalability issues, especially when considering streaming data from multiple sources. In the next section we describe a distributed architecture for addressing these issues and then show how the centralized eigen analysis of the covariance matrix can be formulated as a distributed computation and how it can be solved in a communication efficient manner.

VII. Distributed Principal Component Analysis

When resources become a constraint for doing data mining on massive data sets such as astronomical catalogs, distributed data mining provides a communication efficient solution. For the problem discussed in the last section, we can formulate a distributed architecture where after cross matching the data using a centralized cross matching tool, we can store the meta data information in a central location. Such a service-oriented architecture would facilitate astronomers to query multiple databases and do data mining on large data sets without downloading the data to their local computing resources. The data set is downloaded in parts at a number of computing nodes (that are either dedicated computers connected through communication channels or part of a large grid) based on the meta data information maintained at the central server site. In such a computational environment, distributed data mining algorithms can run in the background seamlessly for providing fast and efficient solutions to the astronomers by distributing the task among a number of nodes. Figure 1 represents one such architecture in which the user submits jobs through the web server and the DDM server will execute these jobs using the underlying P2P architecture.

Another distributed data mining scenario for large scale astronomy databases is the one described in Section II for the LSST project where high throughput data streams need to be modeled and monitored for changes in an efficient manner. In the next few sections we describe a distributed formulation of our centralized eigen analysis and present a eigenstate monitoring algorithm for this purpose.

A. Problem formulation: Distributed Covariance Computation

For the distributed setup, the entire data is not located at a central location. The data set of node P_i is $\mathcal{M}_i$. Note that, $\mathcal{G} = \bigcup_{i=1}^{n} \mathcal{M}_i$. It is true that, if the mean of each column of $\mathcal{G}$ is subtracted from each value of $\mathcal{G}$, *i.e.* $\mathcal{G}_1$ is mean-reduced $\mathcal{G}$, then the covariance matrix of $\mathcal{G}$ *i.e.* $\mathcal{C}$ can be written as $\mathcal{C} = \frac{1}{\#\text{points in } \mathcal{G}} \mathcal{G}_1^{\mathrm{T}} \mathcal{G}_1$. Now, in the distributed setup it is true that:

$$\mathcal{C} = \frac{1}{\#\text{points in } \mathcal{G}} \mathcal{G}_1^{\mathrm{T}} \mathcal{G}_1 = \frac{1}{\#\text{points in } \mathcal{G}} \sum_{i=1}^{n} \mathcal{M}_{1i}^{\mathrm{T}} \mathcal{M}_{1i}$$

where $\mathcal{M}_{1i}$ is mean reduced $\mathcal{M}_i$. Thus it turns out that if data is horizontally partitioned among n nodes and each column of data is mean shifted using the global mean, the covariance matrix is completely decomposable. With this formulation, we now describe certain notations for discussing our distributed eigen monitoring algorithm.

B. Preliminaries

The goal of the PC monitoring algorithm is to track changes to the eigenvectors of the global covariance data matrix $\mathcal{C}$. The crux lies in each node maintaining a current set of eigenvectors which it believes to be globally correct. We call it the *knowledge* of a node. Also each node checks if it is in *agreement* with all of its immediate neighbors with respect to the knowledge. It can be shown that if this is true for all nodes in the network, then the local eigenvectors of each node is indeed the correct global solution. Note that from our earlier discussion, $\mathcal{G}^T\mathcal{G} = \mathcal{C}$ when $\mathcal{G}$ is mean shifted. In the distributed setup, the mean of the global data is not known to each node. Therefore we decompose the PC monitoring algorithm in to (1) mean monitoring which maintains the correct mean of $\mathcal{G}$, and (2) eigenvector monitoring of $\mathcal{G}^T\mathcal{G}$. Given an eigenvalue or a mean as a model, each algorithm monitors changes to the corresponding model with respect to the global data using only its knowledge and agreement. If the data changes such that the models no longer fit the data, the algorithms raise a flag at each node. At this point, a sample of the data is centralized, new models are built and then disseminated to the network. The monitoring algorithms are then restarted with the new models and the process continues. Below we formally define these quantities and describe the algorithms.

1) Notations and Assumptions: In this section we present certain notations necessary for the algorithms.

In the algorithm, each node sends messages to its immediate neighbors to converge to a globally correct solution. As already discussed, there are three kinds of messages: (i) *monitoring* messages which are used by the algorithm to check if the model is up-to-date, (ii) *data* messages which are used to sample data for rebuilding a model, and (iii) *model* messages which are used to disseminate the newly built model in the entire network. In this section we will discuss messages of the first type only. The other two will be discussed in the later sections since they are algorithm specific.

Let the model supplied to each of the monitoring algorithms be denoted by L. For the mean monitoring algorithm, the model is a mean vector $\overrightarrow{\mu}$; for the eigenvector monitoring, the model is a set of eigenvectors ($\overrightarrow{V}$) and eigenvalues (Θ). Let $\mathcal{E}_i(\mathcal{M}_i, L)$ be the error between the model L and the data of node P_i. Explicit computation of $\mathcal{E}_i$ is problem specific and hence described in respective algorithm descriptions. The nodes jointly track if $\mathcal{E}^{\mathcal{G}} = \bigcup_{i=1}^{n} \mathcal{E}_i$ is less than a user-defined threshold ϵ.

Any monitoring message sent by node P_i to P_j contains information that P_i has gathered about the network which P_j may not know. In our case, the message sent by P_i to P_j consists of a set of vectors or matrix[1] $\mathcal{S}_{i,j}$ with each row corresponding to observations and each column corresponding to features. Note that if each node broadcasts $\mathcal{S}_{i,j} = \mathcal{M}_i$, then each node would obviously be able to compute the correct result. However this would be communication intensive. Our next few sets of vectors allow us to compute the correct result in a more communication efficient manner.

- **Knowledge**: This is all the information that P_i has about the error:
$$\mathcal{K}_i = \mathcal{E}_i \cup \bigcup_{P_j \in \Gamma_i} \mathcal{S}_{j,i}$$
- **Agreement**: This is what P_i and P_j have in common:
$$\mathcal{A}_{i,j} = \mathcal{S}_{i,j} \cup \mathcal{S}_{j,i}$$
- **Held**: This is what P_i has not yet communicated to P_j
$$\mathcal{H}_{i,j} = \mathcal{K}_i \setminus \mathcal{A}_{i,j}$$

These sets of vectors can be arbitrarily large. It can be shown that if vectors sent by P_i to P_j are never sent back to P_i, we can do the same computations using only the average vector

[1]we use them interchangeably here

of these sets and the size of the set. One way of ensuring this is to assume that communication takes place over a communication tree – an assumption we make here (see [40] and [8] for a discussion of how this assumption can be accommodated or, if desired, removed).

The following are the notations used for the set statistics — (1) *average*: $\overline{\mathcal{K}_i}$, $\overline{\mathcal{A}_{i,j}}$, $\overline{\mathcal{H}_{i,j}}$, $\overline{\mathcal{S}_{i,j}}$, $\overline{\mathcal{S}_{j,i}}$, $\overline{\mathcal{E}_i}$, and $\overline{\mathcal{E}^{\mathcal{G}}}$, and (2) *sizes*: $|\mathcal{S}_{i,j}|$, $|\mathcal{S}_{j,i}|$, $|\mathcal{K}_i|$, $|\mathcal{A}_{i,j}|$, $|\mathcal{H}_{i,j}|$, $|\mathcal{E}_i|$, and $\left|\mathcal{E}^{\mathcal{G}}\right|$. With these notations, we can now write

- $|\mathcal{K}_i| = |\mathcal{E}_i| + \sum_{P_j \in \Gamma_i} |\mathcal{S}_{j,i}|$
- $|\mathcal{A}_{i,j}| = |\mathcal{S}_{i,j}| + |\mathcal{S}_{j,i}|$
- $|\mathcal{H}_{i,j}| = |\mathcal{K}_i| - |\mathcal{A}_{i,j}|$

Similarly for the average of the sets we can write,

- $\overline{\mathcal{K}_i} = \frac{1}{|\mathcal{K}_i|}\left(|\mathcal{E}_i|\,\overline{\mathcal{E}_i} + \sum_{P_j \in \Gamma_i} |\mathcal{S}_{j,i}|\,\overline{\mathcal{S}_{j,i}}\right)$
- $\overline{\mathcal{A}_{i,j}} = \frac{1}{|\mathcal{A}_{i,j}|}\left(|\mathcal{S}_{i,j}|\,\overline{\mathcal{S}_{i,j}} + |\mathcal{S}_{j,i}|\,\overline{\mathcal{S}_{j,i}}\right)$
- $\overline{\mathcal{H}_{i,j}} = \frac{1}{|\mathcal{H}_{i,j}|}\left(|\mathcal{K}_i|\,\overline{\mathcal{K}_i} - |\mathcal{A}_{i,j}|\,\overline{\mathcal{A}_{i,j}}\right)$

Note that, for any node, these computations are local. For all the monitoring algorithms we assume that message transmission is reliable and ordered.

Since $\overline{\mathcal{E}^{\mathcal{G}}}$ is a vector in $\mathbb{R}^d$, the goal reduces to checking if $\left|\left|\overline{\mathcal{E}^{\mathcal{G}}}\right|\right| < \epsilon$. However, the quantity $\overline{\mathcal{E}^{\mathcal{G}}}$ is not available at any node. In the next section we state a key result which allow us to perform the same computation using only local vectors of a node.

2) Stopping Rule: The main idea of the stopping rule is to describe a condition for each node P_i based on $\overline{\mathcal{K}_i}$, $\overline{\mathcal{A}_{i,j}}$, and $\overline{\mathcal{H}_{i,j}}$, which guarantee that $\overline{\mathcal{E}^{\mathcal{G}}}$ is greater than or less than ϵ. In order to apply this theorem, we need to split the entire domain into non-overlapping convex regions such that the quantity $\left|\left|\overrightarrow{\mathcal{E}^{\mathcal{G}}}\right|\right| < \epsilon$ has the same value inside each of these convex regions. We denote the set of all such convex regions by C_ω. Geometrically, checking if the L2-norm of a vector is less than ϵ is equivalent to checking if $\left|\left|\overrightarrow{\mathcal{E}^{\mathcal{G}}}\right|\right|$ lies inside a circle of radius ϵ. Note that, by construction, the region in which the output is 0 *i.e.* inside the circle is a convex region. Let us denote it by R_c. The outside of the circle can easily be divided into convex regions by drawing random tangent lines to form half-spaces denoted by (R_{h_1}, R_{h_2}, ...). The areas uncovered by C_ω denote the *tie* regions.

As stated by the Theorem below, if the following condition holds, the node can stop sending messages and determine the correct output based solely on its local averages.

Theorem 7.1: [40] Let $P_1, \ldots, P_n$ be a set of nodes connected to each other over a spanning tree $G(V, E)$. Let $\mathcal{E}^{\mathcal{G}}$, $\mathcal{K}_i$, $\mathcal{A}_{i,j}$, and $\mathcal{H}_{i,j}$ be as defined in the previous section. Let R be any region in C_ω. If at time t no messages traverse the network, and for each P_i, $\overline{\mathcal{K}_i} \in R$ and for every $P_j \in \Gamma_i$, $\overline{\mathcal{A}_{i,j}} \in R$ and either $\overline{\mathcal{H}_{i,j}} \in R$ or $\mathcal{H}_{i,j} = \emptyset$, then $\overline{\mathcal{E}^{\mathcal{G}}} \in R$.

Proof: For proof the readers are referred to [40]. ∎

Using this theorem, each node can check if $\left\|\overline{\mathcal{K}_i}\right\| < \epsilon$. If the result holds for every node, then we are guaranteed to get the correct result. If there is any disagreement, it would be between any two neighbors. In that case, messages will be exchanged and they will converge to the same result. In either case, eventual global correctness is guaranteed.

C. Algorithm

Both the mean monitoring algorithm and the eigenvector monitoring rely on the results of Theorem 7.1 to output the correct result. For the eigenvector monitoring, the model supplied to each node are the eigenvector $\overrightarrow{V}$ and eigenvalue Θ. Assuming that the mean of the data is zero, the goal is to check if:

$$\begin{aligned} \left\|\mathcal{C} \cdot \overrightarrow{V} - \Theta\overrightarrow{V}\right\| &\leq \epsilon \\ \Longrightarrow \left\|\frac{1}{|\mathcal{G}|}\left[\mathcal{G}^{\mathrm{T}}\mathcal{G}\right] \cdot \overrightarrow{V} - \Theta\overrightarrow{V}\right\| &\leq \epsilon \\ \Longrightarrow \left\|\frac{1}{\sum_i |\mathcal{M}_i|}\sum_i \left[\mathcal{M}_i^{\mathrm{T}}\mathcal{M}_i\right] \cdot \overrightarrow{V} - \Theta\overrightarrow{V}\right\| &\leq \epsilon \end{aligned}$$

Thus given $\overrightarrow{V}$ and Θ, each node can locally compute the vector $\left(\left[\mathcal{M}_i^{\mathrm{T}}\mathcal{M}_i\right] \cdot \overrightarrow{V} - \Theta\overrightarrow{V}\right)$. Let this instance of problem be denoted by I_1. We can write:

- $I_1.\overline{\mathcal{E}_i} = \frac{\left(\left[\mathcal{M}_i^{\mathrm{T}}\mathcal{M}_i\right] \cdot \overrightarrow{V} - \Theta\overrightarrow{V}\right)}{|\mathcal{M}_i|}$
- $I_1.\left|\mathcal{E}_i\right| = |\mathcal{M}_i|$

Thus for this problem, each node computes the vector $I_1.\overline{\mathcal{E}_i}$ which is then used as input to the eigenvector monitoring algorithm.

Similarly for the mean monitoring algorithm, the model supplied to each node is the mean $\overrightarrow{\mu} \in \mathbb{R}^d$. In this case, each node subtracts the mean $\overrightarrow{\mu}$ from its local average input vector $\mathcal{M}_i$.

The goal is to check if:

$$\begin{aligned} \left\|\overline{\mathcal{G}} - \overrightarrow{\mu}\right\| &\leq \epsilon \\ \left\|\frac{1}{\sum_i |\mathcal{M}_i|}\sum_i \overline{\mathcal{M}_i}\,|\mathcal{M}_i| - \overrightarrow{\mu}\right\| &\leq \epsilon \\ \left\|\frac{1}{\sum_i |\mathcal{M}_i|}\sum_i |\mathcal{M}_i|\left(\overline{\mathcal{M}_i} - \overrightarrow{\mu}\right)\right\| &\leq \epsilon \end{aligned}$$

Note that the quantity $|\mathcal{M}_i|\left(\overline{\mathcal{M}_i} - \overrightarrow{\mu}\right)$ can be locally computed by a node. For this problem instance denoted by I_2, the following are the inputs:

- $I_2.\overline{\mathcal{E}_i} = \left(\overline{\mathcal{M}_i} - \overrightarrow{\mu}\right)$
- $I_2.\,|\mathcal{E}_i| = |\mathcal{M}_i|$

Algorithm 1 presents the pseudo-code of the monitoring algorithm while Alg. 2 presents the pseudo-code for the algorithm which builds the model. The inputs to the monitoring algorithm are $\mathcal{M}_i$, $\mathcal{E}_i$ (depending on how it is defined), Γ_i, ϵ and C_ω and L. For each problem instance I_1 and I_2, each node initializes its local vectors $\overline{\mathcal{K}_i}$, $\overline{\mathcal{A}_{i,j}}$ and $\overline{\mathcal{H}_{i,j}}$. Below we describe the monitoring algorithm with respect to only one instance I_1 (and hence drop the instance index I_1). The other case is identical. The algorithm is entirely event-driven. Events can be one of the following: (i) change in local data $\mathcal{M}_i$, (ii) on receiving a message, and (iii) change in Γ_i. In any of these cases, the node checks if the condition of the theorem holds. Based on the value of its knowledge $\overline{\mathcal{K}_i}$, the node selects the active region $R_\ell \in C_\omega$ such that $\overline{\mathcal{K}_i} \in R_\ell$. If no such region exists, $R_\ell = \emptyset$. If $R = \emptyset$, then $\overline{\mathcal{K}_i}$ lies in the tie region and hence P_i has to send all its data. On the other hand, if $R_\ell \neq \emptyset$ the node can rely on the result of Theorem 7.1 to decide whether to send a message. If for all $P_j \in \Gamma_i$, both $\overline{\mathcal{A}_{i,j}} \in R_\ell$ and $\overline{\mathcal{H}_{i,j}} \in R_\ell$, P_i does nothing; else it needs to set $\overline{\mathcal{S}_{i,j}}$ and $|\mathcal{S}_{i,j}|$. Based on the conditions of the Theorem, note that these are the only two cases when a node needs to send a message. Whenever it receives a message ($\overline{\mathcal{S}}$ and $|\mathcal{S}|$), it sets $\overline{\mathcal{S}_{j,i}} \leftarrow \overline{\mathcal{S}}$ and $|\mathcal{S}_{j,i}| \leftarrow |\mathcal{S}|$. This may trigger another round of communication since its $\overline{\mathcal{K}_i}$ can now change.

To prevent message explosion, in our event-based system we employ a "leaky bucket" mechanism which ensures that no two messages are sent in a period shorter than a constant L. Note that this mechanism does not enforce synchronization or affect correctness; at most it might delay convergence. This technique has been used elsewhere also [40][7].

Algorithm 1: Monitoring eigeivector/eigenvalues.

Input: ϵ, C_ω, $\mathcal{E}_i$, Γ_i and L
Output: 0 if $\left|\left|\overline{\mathcal{K}_i}\right|\right| < \epsilon$, 1 otherwise
Initialization: Initialize vectors;
if MessageRecvdFrom $\left(P_j, \overline{\mathcal{S}}, |\mathcal{S}|\right)$ **then**
  $\overline{\mathcal{S}_{j,i}} \leftarrow \overline{\mathcal{S}}$;
  $|\mathcal{S}_{j,i}| \leftarrow |\mathcal{S}|$;
  Update vectors;
if $\mathcal{M}_i$, Γ_i *or* $\mathcal{K}_i$ *changes* **then**
  forall the $P_j \in \Gamma_i$ **do**
    if $LastMsgSent > L$ *time units ago* **then**
      if $R_\ell = \emptyset$ **then**
        $\overline{\mathcal{S}_{i,j}} \leftarrow \frac{|\mathcal{K}_i|\overline{\mathcal{K}_i} - |\mathcal{S}_{j,i}|\overline{\mathcal{S}_{j,i}}}{|\mathcal{K}_i| - |\mathcal{S}_{j,i}|}$;
        $|\mathcal{S}_{i,j}| \leftarrow |\mathcal{K}_i| - |\mathcal{S}_{j,i}|$;
      if $\overline{\mathcal{A}_{i,j}} \notin R_\ell$ *or* $\overline{\mathcal{H}_{i,j}} \notin R_\ell$ **then**
        Set $\overline{\mathcal{S}_{i,j}}$ and $|\mathcal{S}_{i,j}|$ such that $\overline{\mathcal{A}_{i,j}}$ and $\overline{\mathcal{H}_{i,j}} \in R_\ell$;
      MessageSentTo$\left(P_j, \overline{\mathcal{S}_{i,j}}, |\mathcal{S}_{i,j}|\right)$;
      $LastMsgSent \leftarrow$ CurrentTime;
      Update all vectors;
    else Wait L time units and then check again;

The monitoring algorithm raises a flag whenever either $\left\|I_1.\overline{\mathcal{K}_i}\right\| > \epsilon$ or $\left\|I_2.\overline{\mathcal{K}_i}\right\| > \epsilon$. Once the flag is set to 1, the nodes engage in a convergecast-broadcast process to accumulate data up the root of the tree, recompute the model and disseminate it in the network.

For the mean monitoring algorithm in the convergecast phase, whenever a flag is raised, each leaf node in the tree forwards its local mean up the root of the tree. In this phase, each node maintains a user selected alert mitigation constant, τ which ensures that an alert is stable for a given period of time τ for it to send the data. Experimental results show that this is crucial in preventing a false alarm from progressing, thereby saving resources. In order to implement this, whenever the monitoring algorithm raises a flag, the node notes the time, and sets a timer to τ time units. Now, if the timer expires, or a data message is received from one of its neighbors, P_i first checks if there is an existing alert. If it has been recorded τ or more time units ago, the node does one of the following. If it has received messages from all its neighbors, it recomputes the new mean, sends it to all its neighbors and restarts its monitoring algorithm with the new mean.

On the other hand, if it has received the mean from all but one of the neighbors, it combines its data with all of its neighbors' data and then sends it to the neighbor from which it has not received any data. Other than any of these cases, a node does nothing.

For the eigenvector monitoring, in place of sending a local mean vector, each node forwards the covariance matrix $\mathcal{C}_i$. Any intermediate node accumulates the covariance matrix of its children, adds it local matrix and sends it to its parent up the tree. The root computes the new eigenvectors and eigenvalues. The first eigenstate is passed to the monitoring algorithm.

D. Correctness and Complexity Analysis

The eigen monitoring algorithm is eventually correct.

Theorem 7.2: The eigen monitoring algorithm is **eventually correct**.

Proof: For the eigen monitoring algorithm, the computation will continue for each node unless one of the following happens:

- for every node, $\overline{\mathcal{K}_i} = \overline{\mathcal{E}^{\mathcal{G}}}$
- for every P_i and every neighbor P_j, $\mathcal{K}_i$, $\mathcal{A}_{i,j}$,and $\mathcal{H}_{i,j}$ are in the same convex region $R_\ell \in C_\omega$.

In the former case, every node obviously computes the correct output since the knowledge of each node becomes equal to the global knowledge. In the latter case, Theorem 7.1 dictates that $\overline{\mathcal{E}^{\mathcal{G}}} \in R_\ell$. Note that by construction, the output of the monitoring function (in this case L2-norm) is invariant inside R_ℓ. In other words, the binary function $\left\|\overline{\mathcal{E}^{\mathcal{G}}}\right\| < \epsilon$ and $\left\|\overline{\mathcal{K}_i}\right\| < \epsilon$ will have the same output inside R_ℓ. Therefore in either of the cases, the eigen monitoring algorithm is correct. ∎ ■

Determining the communication complexity of local algorithms in dynamic environments is still an open research issue. Researches have proposed definitions of locality [7][40]. Note that for an exact algorithm as the eigen monitoring algorithm, the worst case communication complexity is $O(sizeofnetwork)$. This can happen, for example, when the each node has a vector in a different convex region and the global average is in another different region. However, as shown in this paper and also by several authors [40][7] there are several problem instances for which the resource consumption becomes independent of the size of the network. Interested readers are referred to [6] for a detailed discussion on communication complexity and locality of such algorithms.

Algorithm 2: P2P Eigen-monitoring Algorithm.

Input: ϵ, C_ω, $\mathcal{M}_i$, Γ_i, L, τ
Output: (i) $\overrightarrow{V}, \Theta$ such that $\left\|\mathcal{C} \cdot \overrightarrow{V} - \Theta \cdot \overrightarrow{V}\right\| < \epsilon$, (ii) $\overrightarrow{\mu}$ such that $\left\|\overline{\mathcal{G}} - \overrightarrow{\mu}\right\| < \epsilon$
Initialization
begin
  Initialize vectors;
  MsgType = MessageRecvdFrom(P_j);
if $MsgType = Monitoring\_Msg$ **then** Pass Message to appropriate Monitoring Algorithm;
if $MsgType = New\_Model\_Msg$ **then**
  Update $\overrightarrow{V}$, Θ, $\overrightarrow{\mu}$;
  Forward new model to all neighbors;
  Datasent=false;
  Restart Monitoring Algorithm with new models;
if $MsgType = Dataset\_Msg$ **then**
  if Received from all but one neighbor **then**
    $flag$=Output Monitoring Algorithm();
    if $Datasent = false$ *and* $flag = 1$ **then**
      if DataAlert stable for τ time **then**
        D_1=$\mathcal{C}_i$+ $Recvd\_Covariance$;
        D_2=$\overline{\mathcal{M}_i}$+ $Recvd\_Mean$;
        $Datasent$=true;
        Send D_1, D_2 to remaining neighbor
      else *DataAlert*=CurrentTime;
  if Received from all neighbors **then**
    D_1=$\mathcal{C}_i$+ $Recvd\_Covariance$;
    D_2=$\overline{\mathcal{M}_i}$+ $Recvd\_Mean$;
    $(\overrightarrow{V}, \Theta)$=EigAnalysis($D_1$);
    $\overrightarrow{\mu}$ =Mean(D_2);
    Forward new $\overrightarrow{V}, \Theta, \overrightarrow{\mu}$ to all neighbors;
    Datasent=false;
    Restart Monitoring Algorithm with new models;
if $\mathcal{M}_i$, Γ_i *or* $\mathcal{K}_i$ *changes* **then**
  Run Monitoring Algorithm;
  flag=Output_Monitoring_Algorithm();
  if flag=*1 and* P_j=*IsLeaf()* **then**
    Execute the same conditions as $MsgType = Dataset\_Msg$

VIII. Applications

In this section we show several applications of using a distributed eigen monitoring algorithm for

IX. Experimental Results

In this section we demonstrate the experimental results of both the centralized fundamental plane analysis and the distributed eigen monitoring algorithm. The centralized experiments show how the fundamental plane changes with variations in galactic density, while the distributed experiments show the performance of the eigen monitoring algorithm for a streaming scenario of the same experiment. Our goal is to demonstrate that, using our distributed eigen monitoring algorithm to compute the principal components and monitor them in a streaming scenario, we can find very similar results as were obtained by applying a centralized PCA. Even though our goal was not to make a new discovery in astronomy, the results are astronomically noteworthy. We argue that our distributed algorithm could have found very similar results to the centralized approach at a fraction of the communication cost. Also, we want to emphasize that this distributed eigen monitoring algorithm can be applied to a number of change-detection applications in high-throughput streaming scenarios (such as the LSST) for important astronomical discoveries of many types. The importance and novelty of this algorithm compared to existing distributed PCA algorithms is that, this is an exact algorithm that deterministically converges to the correct result.

A. Fundamental Plane Results

As noted in Section VI-A, we divide the entire dataset into 30 bins. The bins are arranged from low to high density. In this section we present the results of our fundamental plane experiments for only the elliptical galaxies for 30 bins.

Our first experiment (Figure 5) shows the variance captured by the first PC (PC1) as the density of the galaxies increase. The x-axis shows the mean density of each bin in $\log$-scale. As seen in the figure, the variance captured by PC1 increases monotonically with increase in mean galactic density.

Figure 6 provides the most significant scientific result. It demonstrates the dependence of the variance captured by the first 2 PC's with respect to $\log$ of bin density. As seen, the variance increases monotonically from almost 95% to 98% with increase in galactic bin density. This clearly demonstrates a new astrophysical effect, beyond that traditionally reported in the astronomical literature. This results from the application of distributed data mining (DDM) on a significantly (by 1000 times) larger set of data. More such remarkable discoveries can be

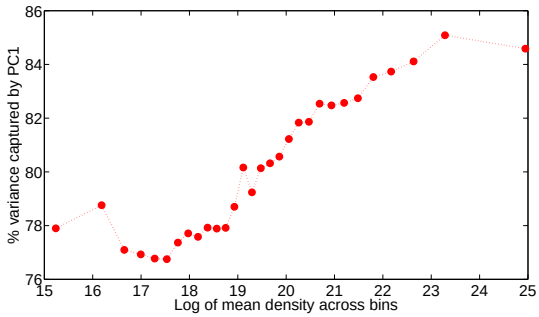


Fig. 5. Variance captured by PC 1 w.r.t. log of mean density of each bin. Bin 1 has the lowest mean density and Bin 30 the highest. The variance captured by PC1 increases monotonically with increasing bin density.

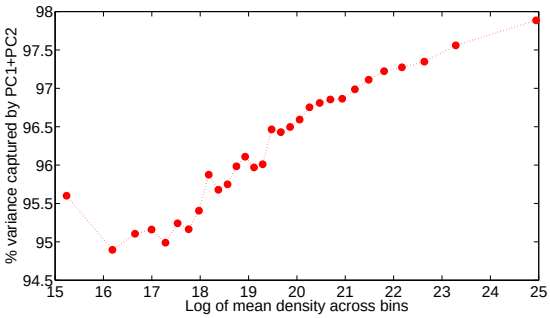
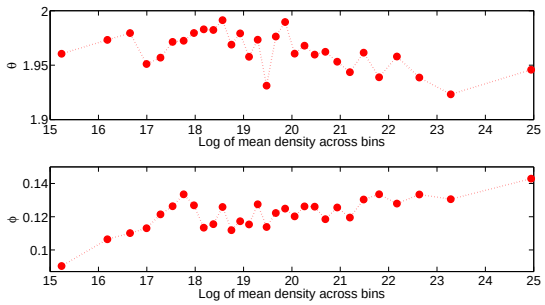


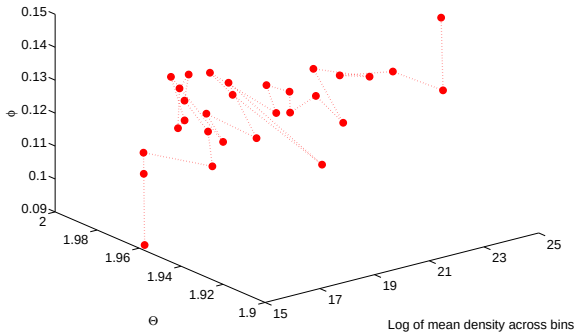
Fig. 6. Variance captured by PCs 1 and 2 w.r.t. log of mean density of each bin. Bin 1 has the lowest mean density and Bin 30 the highest.

anticipated when DDM algorithms of the type reported here are applied to massive scientific (and non-scientific) data streams of the future.

To analyze more deeply the nature of the variation of the first two PCs with respect to increasing galactic density, we plot the direction of the normal to the plane defined by the first 2 PCs *i.e.* pc1 and pc2. Since each of these PC's are vectors in 3-d, so is the normal to the plane. The normal vector is represented by its two directional angles: the spherical polar angles θ and ϕ. Figure 7 shows a plot of θ and ϕ for 30 bins. Figure 7(a) shows the variation of θ and ϕ independently with $\log$ of mean galactic density. Figure 7(b) shows the variation of both with $\log$ of mean density. The systematic trend in the change of direction of the normal vector seen in Figure 7(b) is a new astronomy result. This represents exactly the type of change detection from eigen monitoring that will need to be applied to massive scientific data streams, including large astronomy applications (LSST) and large-scale geo-distributed sensor networks, in order



(a) Variation of θ and ϕ independently w.r.t. log of bin density for 30 bins



(b) Variation of θ and ϕ w.r.t. log of bin density for 30 bins

Fig. 7. Plot of variation of θ and ϕ independently with bin number. The bins are numbered in increasing order of density.

to facilitate knowledge discovery from these petascale data collections.

B. Results of Distributed PCA Algorithm

The distributed PCA implementation makes use of the Distributed Data Mining Toolkit (DDMT) [20] — a distributed data mining development environment from DIADIC research lab at UMBC. DDMT uses topological information which can be generate by BRITE [15], a universal topology generator from Boston University. In our simulations we used topologies generated according to the *Barabasi Albert (BA)* model. On top of the network generated by BRITE, we overlayed a spanning tree. We have experimented with network size ranging from 50 to 1000 nodes.

We have divided the data of the centralized experiments into 5 bins (instead of 30) sorted by galactic density. Each bin represents the data distribution at a certain time in the streaming scenario and the distribution changes every 200,000 simulation ticks which we call an epoch.

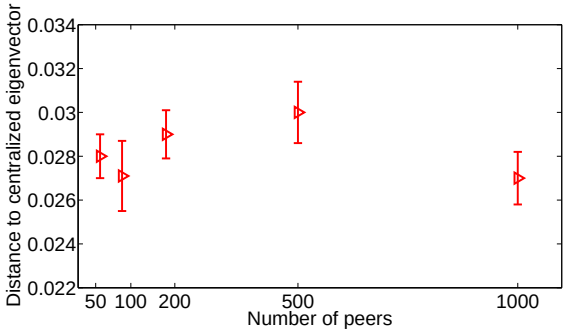


Fig. 8. Quality vs. number of nodes. Quality remains the same thereby showing good accuracy.

This implies that every 200,000 simulation ticks we supply the nodes with a new bin of data. We stream the data at a rate of 10% of the bin size for every 10,000 simulation ticks. The two quantities measured in our experiments are the *quality* of the result and the *cost* of the algorithm.

We have used the following default values for the algorithm: size of leaky bucket $L = 500$, error threshold $\epsilon = 1.0$, alert mitigation constant $\tau = 500$. Due to shortage of space we do not present an exhaustive analysis of the effect of all these parameters. We plan to report these in an extended version.

For the eigen monitoring algorithm, quality is the average L2 norm distance between the principal eigen vector and the and the computed eigen vector in the distributed scenario over all the bins. Since we compute the principal eigen vector for each bin separately, we plot the average L2 norm distance between the centralized and distributed eigen vectors for every experiment. The experiment was repeated for 10 independent trials. Figure 8 shows the scalability results for the accuracy achieved by our algorithm. As shown in the figure, the proposed eigen monitoring algorithm produces results which are quite close to their centralized counterpart. Moreover, we can also observe that the quality does not degrade with increasing network size. Because our algorithm is provably correct, the number of nodes has no influence on the quality of the result.

Figures 9 and 10 show the number of messages exchanged per node when the number of nodes is increased from 50 to 1000. As shown in Figure 9, the normalized L2 messages per node is approximately 0.3. Normalized message per node means the number of messages sent by a node per unit of leaky bucket. Note that for an algorithm which uses broadcast as the communication model, its normalized messages will be 2.0, assuming two neighbors per node on average. Thus the proposed algorithm is highly efficient with respect to communication. Also

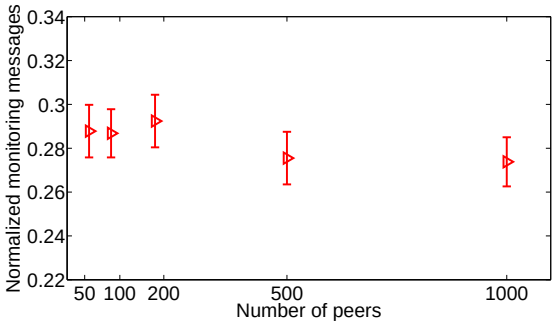


Fig. 9. L2 messages vs. number of nodes. Number of messages remain constant showing excellent scalability.

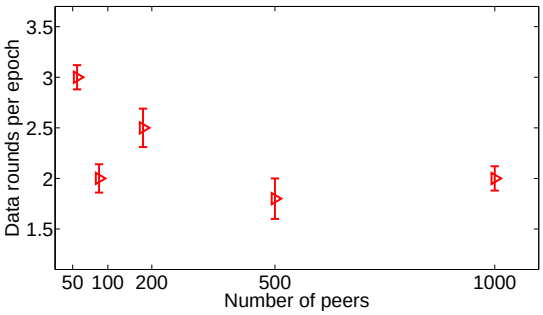


Fig. 10. Number of convergecast rounds per epoch vs. number of nodes. In most cases the convergecast round is less than 3 per epoch.

as shown, the L2 messages remain a constant even if the number of nodes is increased. This demonstrates the excellent scalability of the algorithm.

Finally, we also plot the number of times data is collected per epoch. In most cases, the number of such convergecast rounds is 3 per epoch. Note that this can be reduced further by using a larger alert mitigation constant τ, larger error threshold ϵ or larger local data set size.

X. Conclusion

This paper presents a local and completely asynchronous algorithm for monitoring the eigenstates of distributed and streaming data. The algorithm is efficient and exact in the sense that once computation terminates, each node in the network computes the globally correct model. We have taken a relatively well understood problem in astronomy — that of galactic fundamental plane computation and shown how our distributed algorithm can be used to arrive at the same results without any data centralization. We argue that this might become extremely useful when

petabyte scale data repositories such as the LSST project start to generate high throughput data streams which need to be co-analyzed with other data repositories located at diverse geographic location. For such large scale tasks, distributing the data and running the algorithm on a number of nodes might prove to be cost effective. Our algorithm is a first step to achieving this goal. Experiments on current SDSS and 2MASS dataset show that the proposed algorithm is efficient, accurate, and highly scalable.

ACKNOWLEDGMENTS

This research is supported by the NASA Grant NNX07AV70G and the AFOSR MURI Grant 2008-11. K. Das completed the research for this paper at University of Maryland, Baltimore County. C. Giannella completed the research for this paper while being an Assistant Professor of Computer Science at Loyola College in Maryland and New Mexico State University. The paper was approved for Public Release, unlimited distribution, by The MITRE Corporation: 10-0814; c2010-The MITRE Corporation, all rights reserved. The authors would also like to thank Sugandha Arora and Wesley Griffin for helping with the experimental setup.

REFERENCES

[1] H. Ang, V. Gopalkrishnan, S. Hoi, and W. Ng. Cascade RSVM in Peer-to-Peer Networks. In *Proceedings of PKDD'08*, pages 55–70, 2008.

[2] The AUTON Project. `http://www.autonlab.org/autonweb/showProject/3/`.

[3] W. Balke, W. Nejdl, W. Siberski, and U. Thaden. Progressive Distributed Top-K Retrieval in Peer-to-Peer Networks. In *Proceedings of ICDE'05*, pages 174–185, 2005.

[4] N. M. Ball and R. J. Brunner. Data Mining and Machine Learning in Astronomy. arXiv:0906.2173v1, 2009.

[5] M. Bawa, A. Gionis, H. Garcia-Molina, and R. Motwani. The Price of Validity in Dynamic Networks. *Journal of Computer and System Sciences*, 73(3):245–264, 2007.

[6] K. Bhaduri. *Efficient Local Algorithms for Distributed Data Mining in Large Scale Peer to Peer Environments: A Deterministic Approach*. PhD thesis, University of Maryland, Baltimore County, March 2008.

[7] K. Bhaduri and H. Kargupta. An Scalable Local Algorithm for Distributed Multivariate Regression. *Statistical Analysis and Data Mining*, 1(3):177–194, November 2008.

[8] K. Bhaduri, R. Wolff, C. Giannella, and H. Kargupta. Distributed Decision Tree Induction in Peer-to-Peer Systems. *Statistical Analysis and Data Mining*, 1(2):85–103, June 2008.

[9] K. Borne. A Machine Learning Classification Broker for the LSST Transient Database. *Astronomische Nachrichten*, 329:255, 2008.

[10] K. Borne. Data Science Challenges from Distributed Petascale Astronomical Sky Surveys. In *Proceedings of DOE Workshop on Mathematical Analysis of Petascale Data*, 2008.

[11] K. Borne. The LSST Data Mining Research Agenda. In *Proceedings of AIP*, volume 1087, pages 347–351, 2008.

[12] K. Borne. Scientific Data Mining in Astronomy. In *Next Generation of Data Mining*, chapter 5, pages 91–114. CRC press, 2009.

[13] S. Boyd, A. Ghosh, B. Prabhakar, and D. Shah. Gossip Algorithms: Design, Analysis, and Applications. In *IEEE Infocom*, volume 3, pages 1653–1664, 2005.

[14] J. Branch, B. Szymanski, C. Giannella, R. Wolff, and H. Kargupta. In-Network Outlier Detection in Wireless Sensor Networks. In *Proceedings of ICDCS'06*, page 51, 2006.

[15] Boston University Representative Internet Topology Generator. `http://www.cs.bu.edu/brite/`.

[16] The ClassX Project: Classifying the High-Energy Universe. `http://heasarc.gsfc.nasa.gov/classx/`.

[17] DAME: DAta Mining and Exploration. http://voneural.na.infn.it/.

[18] S. Datta, C. Giannella, and H. Kargupta. Approximate Distributed K-Means Clustering over a Peer-to-Peer Network. *IEEE Transactions on Knowledge and Data Engineering*, 21(10):1372–1388, 2009.

[19] Digital Dig - Data Mining in Astronomy. `http://www.astrosociety.org/pubs/ezine/datamining.html`.

[20] The Distributed Data Mining Toolkit. `http://www.umbc.edu/ddm/wiki/software/DDMT`.

[21] CGAL: Delaunay Triangulation. http://www.cgal.org/.

[22] Data Mining Grid. `http://www.datamininggrid.org/`.

[23] H. Dutta, C. Giannella, K. Borne, and H. Kargupta. Distributed Top-K Outlier Detection from Astronomy Catalogs using the DEMAC System. In *Proceedings of SDM'07*, 2007.

[24] Elliptical Galaxies: Merger Simulations and the Fundamental Plane. `http://irs.ub.rug.nl/ppn/244277443`.

[25] Framework for Mining and Analysis of Space Science Data. `http://www.itsc.uah.edu/f-mass/`.

[26] GRIST: Grid Data Mining for Astronomy. `http://grist.caltech.edu`.

[27] Hadoop Home Page. `http://hadoop.apache.org/`.

[28] T. Hinke and J. Novotny. Data Mining on NASA's Information Power Grid. In *Proceedings of HPDC'00*, page 292, 2000.

[29] L. Huang, X. Nguyen, M. Garofalakis, M. Jordan, A. Joseph, and N. Taft. Distributed PCA and Network Anomaly Detection. Technical Report UCB/EECS-2006-99, EECS Department, University of California, Berkeley, 2006.

[30] International Virtual Observatory. `http://www.ivoa.net`.

[31] H. Kargupta and P. Chan, editors. *Advances in Distributed and Parallel Knowledge Discovery*. MIT Press, 2000.

[32] H. Kargupta, W. Huang, K. Sivakumar, and E. L. Johnson. Distributed Clustering Using Collective Principal Component Analysis. *Knowledge and Information Systems*, 3(4):422–448, 2001.

[33] H. Kargupta and K. Sivakumar. *Existential Pleasures of Distributed Data Mining. Data Mining: Next Generation Challenges and Future Directions*. AAAI/MIT press, 2004.

[34] D. Kempe, A. Dobra, and J. Gehrke. Computing Aggregate Information Using Gossip. In *Proceedings of FOCS'03*, pages 482–491, 2003.

[35] D. Krivitski, A. Schuster, and R. Wolff. A Local Facility Location Algorithm for Large-Scale Distributed Systems. *Journal of Grid Computing*, 5(4):361–378, 2007.

[36] P. Luo, H. Xiong, K. Lu, and Z. Shi. Distributed Classification in Peer-to-Peer Networks. In *Proceedings of KDD'07*, pages 968–976, 2007.

[37] US National Virtual Observatory. `http://www.us-vo.org/`.

[38] W.E. Schaap. *The Delaunay Tessellation Field Estimator*. PhD thesis, University of Groningen, 2007.

[39] I. Sharfman, A. Schuster, and D. Keren. A Geometric Approach to Monitoring Threshold Functions Over Distributed Data Streams. *ACM Transactions on Database Systems*, 32(4):23, 2007.

[40] R. Wolff, K. Bhaduri, and H. Kargupta. A Generic Local Algorithm for Mining Data Streams in Large Distributed Systems. *IEEE Transactions on Knowledge and Data Engineering*, 21(4):465–478, 2009.

[41] R. Wolff and A. Schuster. Association Rule Mining in Peer-to-Peer Systems. *IEEE Transactions on Systems, Man and Cybernetics, Part B*, 34:2426–2438, 2004.