APHID: A Practical Architecture for High-Performance, Privacy-Preserving Data Mining

Jimmy Secretan, Anna Koufakou, Michael Georgiopoulos

Abstract—**While the emerging field of privacy preserving data mining (PPDM) will enable many new data mining applications, it suffers from several practical difficulties. PPDM algorithms are difficult to develop and computationally intensive to execute. Developers need convenient abstractions to reduce the costs of engineering PPDM applications. The individual parties involved in the data mining process need a way to bring high-performance, parallel computers to bear on the computationally intensive parts of the PPDM tasks. This paper discusses APHID (Architecture for Private and High-performance Integrated Data mining), a practical software architecture for developing and executing large scale PPDM applications. At one tier, the system supports simplified use of cluster and grid resources, and at another tier, the system abstracts communication for easy PPDM algorithm development. This paper offers a detailed analysis of the challenges in developing PPDM algorithms with existing frameworks, and motivates the design of a new infrastructure based on these challenges.**

Index Terms—**Privacy-Preserving Data Mining, Distributed Data Mining, Cluster Computing, MapReduce**

I. Introduction

Modern organizations manage an unprecedented amount of data, which can be mined to generate valuable knowledge, using several available data mining techniques. While data mining is useful within an organization, it can yield further benefits with the combined data of multiple organizations. And, in fact, many organizations are interested in collaboratively mining their data. This sharing of data, however, creates many potential privacy problems. Many organizations, such as health organizations, have restrictions on data sharing. Businesses may be apprehensive to share trade secrets despite the value of cooperative data mining. At the same time, privacy concerns for individuals are rapidly gaining attention. Instead of dispensing entirely with cooperative data mining, research has instead focused on Privacy Preserving Data Mining (PPDM), which uses various techniques, statistical, cryptographic and others, to facilitate cooperative data mining while protecting the privacy of the organizations or individuals involved.

However, PPDM research is still in its infancy, and few practical systems are currently in place. Even if organizations currently have the legal infrastructure in place for sharing data, there is a lack of developmental support for PPDM systems. Organizations attempting to implement PPDM systems would face a lack of available toolkits, libraries, middleware and architectures that are ready for deployment. The costs involved are potentially high, because of the lack of familiarity with PPDM technology. In addition, because complex computation is often required, high performance and parallel computing technologies are necessary for efficient operation, adding yet another level of complexity to development. The purpose of this research is to provide an architecture and development environment that will allow organizations to easily develop and execute PPDM software. By borrowing from familiar parallel paradigms, the architecture aims to ease the introduction of PPDM technology into the existing database infrastructure. Furthermore, the system seamlessly integrates high performance computing technologies, to ensure an efficient data mining process.

Jimmy Secretan, Anna Koufakou and Michael Georgiopoulos are with the Department of Electrical and Computer Engineering, University of Central Florida, Orlando, FL 32816, USA (phone: 407-882-2016; fax: 407-823-5835; emails: jsecreta@ucf.edu, akoufako@mail.ucf.edu and michaelg@mail.ucf.edu).

Because of extensive communication over relatively slow wide area networks, and because of the large computational requirements of cryptographic and other privacy-oriented technologies, resource requirements for PPDM algorithms can be intense. One study [1] describes taking *29 hours* to build a 408-node decision tree from a 1728 item training set. While there is much research that discusses available algorithms and techniques in PPDM, few studies focus on high-performance computational architectures that support them. Therefore, this research presents a development environment and runtime system specifically geared toward PPDM.

The contributions of this research are: (1) middleware for managing the execution of PPDM algorithms across multiple organizations, (2) the integration of high performance and parallel computing middleware into the PPDM execution environment, and (3) a simple development framework for PPDM software. First in this paper, a brief background on PPDM and high performance computing middleware is given. Existing frameworks are analyzed in terms for their strengths and weaknesses. Then, the model of development used in the system is delineated and justified. Next, the design of the PPDM runtime environment is discussed, followed by the details of the implementation. A PPDM example of the Naïve Bayes classifier is presented, with an associated design and sample code for the system.

II. Privacy Preserving Data Mining

Privacy Preserving Data Mining (PPDM) concentrates on how to coherently combine and mine databases to preserve the privacy of the individual parties' data. PPDM shares a common line of research with Distributed Data Mining (DDM). DDM analyzes the problem of coherently mining data at multiple sites, while considering processing power, network transfer, format compatibility and many other issues encountered in the course of coherently mining disparate data sets. PPDM adds a privacy dimension to these challenges. PPDM methods may address individual privacy (i.e. the privacy of specific customers or patients), or collective privacy, the privacy of information about an organization's records overall (i.e. summary statistics, etc) [2].

First, to support PPDM, there are data perturbation methods, which obfuscate the original data with random perturbations, but doing it so that the original distributions of the data can be easily recovered. Another, similar area involves using signal processing techniques to make approximations of the data distributions, which more effectively preserve privacy than access to the raw data. A different approach leverages Secure Multi-party Computation (SMC) to compute the data mining functions in a way that is more exact and private, albeit at the expense of computational efficiency. Because of these potential advantages, PPDM through SMC is the focus of the proposed architecture, although much of the architecture could be applied to other PPDM techniques.

SMC has as one of its pillars Yao's work to solve the Millionaire's Problem [3]. The Millionaire's Problem is as follows: two millionaires wish to find who has more money, but neither wants to disclose his/her individual amount. Yao proved that there was a secure way to solve this problem by representing it as a circuit, sharing random portions of the outputs. Later it was proved in [4], that any function could be securely computed using this kind of arrangement. However, using Yao circuits is typically inefficient. The problem must be represented as a circuit, which may be large, especially for complex algorithms. Yao circuits are typically only practical for small sub-problems within the PPDM process. sub problems. This gives rise to more specific solutions based on cryptographic primitives such as secure sums, secure set unions, and secure scalar products [5].

Many machine learning algorithms have been recast into privacy preserving versions, including decision trees [6], the Naïve Bayes classifier [7], k-Means Clustering [8], Support Vector Machines [9], k-NN [10], and Association Rule Mining [11], just to name a few. It is to support the implementation of these and future algorithms that APHID was developed.

III. Related Work

To develop a practical architecture for PPDM, we must first observe what technologies are available to support the process. In this section, we examine those technologies. First, the field of high-performance data mining will provide the necessary infrastructure and frameworks for the computationally intensive portion of the PPDM process. Next an analysis of existing DDM middleware and web-services approaches to data mining will show what principles are worth keeping in a framework, and which are lacking.

A. High Performance Data Mining on Clusters and Grids

Systems that provide convenient abstractions for simple development within a parallel environment have received a great deal of interest in recent years. One of the most popular, MapReduce [12] is a simplified parallel program paradigm for large scale, data intensive parallel computing jobs. By constraining the parallel programming model to only the *map* function and the *reduce* function, the MapReduce infrastructure can simplify parallel programming. MapReduce versions of many machine learning algorithms have been developed including k-means, Logistic Regression, Naïve Bayes, linear Support Vector Machines, and many others [13].

Concepts within grid computing hope to make the use of computational resources, even those across organizational and administrative boundaries, as easy as drawing resources from the power grid. These systems include Networks of Workstations (NOW) architectures, which take advantage of idle machines to lower system costs. However, developing the software to support these architectures can be challenging, as they are often less reliable, less available, and have fewer resources than their dedicated cluster counterparts. In [14] a system for data mining in NOWs is developed, built on a simple primitive called *D*istributed DOALL. *Distributed DOALL* can be applied to loops that have no loop carried dependencies or conflicts, loops which are frequently encountered in data mining.

Because these technologies are oriented toward trusted local environments, they cannot support the PPDM process *per se*. However, they are capable of supporting computationally intensive sub-tasks which are found in PPDM.

B. Distributed Data Mining

Often, separate organizations or multiple sites of a single organization want to collaboratively mine databases which are in different geographic locations. Distributed Data Mining (DDM) research develops techniques and architectures to mine databases distributed across the Internet, while working within the constraints of limited bandwidth and computing.

A system called the Knowledge Grid (K-Grid), is a comprehensive architecture and tool suite for DDM discussed in [15] and [16]. The core layer of K-Grid is implemented on top of Globus [17] services. K-Grid follows a *collection-of-services* approach: that is, the functions of the middleware are available as numerous services, which the application employs. The core layer is responsible for managing the metadata about data sources, algorithms and mining tools, as well matching the requirements of the data mining tools with the necessary grid resources. The high level layer of the K-grid software is responsible for orchestrating the execution algorithms and the production of models.

One of the most comprehensive DDM systems is the DataMiningGrid [18] software. It provides functionality for tasks such as data manipulation, resource brokering, and parameter sweeps. It was developed to encourage grid transparency, the adaptation of current code to be grid enabled, through a service oriented architecture (SOA). The authors emphasize that extensibility is important to DataMiningGrid, and that developers should be able to add new components without adversely affecting the existing large components and implementations in the system.

Systems like K-Grid and DataMiningGrid provide excellent frameworks for unifying the data mining resources of several organizations. However, they are not specifically designed to ease the complexity of developing distributed algorithms, but mostly for devising multi-stage distributed data mining processes. Furthermore, they offer no additional support for PPDM algorithms.

While DDM frameworks cannot be immediately adapted to the needs of PPDM, they still share much in the way of concerns and challenges. Performance, scalability, portability and cost of development are concerns for both disciplines. Therefore, we can emulate some of the techniques of DDM systems to apply to a PPDM framework.

The authors of [19] advocate having the most flexible architecture possible to support DDM. They mention that new algorithms should be included easily, the system should integrate relational and data mining operators, and the system should integrate interoperability and resource management mechanisms. They argue that DDM architectures should be built not to support a specific kind of data mining paradigm (classification, ARM, clustering, etc), but should instead offer a broad base of support, much like an operating system.

The system described in [20] emphasizes scalability and portability. The system uses Java for the language, RMI for the intercommunication, XML for much of the storage, and JDBC for database connections. While this is put together into a flexible and efficient framework, the fact that a language choice is imposed may limit a company's ability to adapt current infrastructure to DDM environments, and therefore may hinder adoption.

SOAs are gaining popularity in DDM. In this paradigm, data sets and algorithms can be viewed as independently hosted services, called whenever they are needed. The system in [21] uses an execution framework in conjunction with a registry of algorithms and databases to complete a large-scale data mining task, by matching tasks to be executed to available services. SOAs decouple DDM systems, by allowing various parts of the applications to be hosted in different environments.

C. Architectures to Support PPDM

Systems to support an SMC-based PPDM process have begun to appear in the literature. TRIMF [22] proposes a runtime environment to support privacy preserving data access and mining. Built on top of a service oriented architecture and communicating over JXTA peer-to-peer technology [23], TRIUMF aims to provide an ensemble of related services for PPDM. TRIMF also supports fine-grained access control where each party can specify which data is accessible and to whom. While TRIUMF can enable efficient PPDM processes, and scale to many parties, it does not suggest a framework with which to implement PPDM algorithms.

In [24] the authors suggest a hierarchical structure combining P2P and grid concepts in order to efficiently support PPDM. Peers within virtual organizations (VO) communicate locally, and then use super-peers to communicate among the VOs. While an architecture resembling this has tremendous potential for facilitating large-scale PPDM, it is not clear exactly how a system like this would operate, and what kind of programming model it would use.

The system described in [25] suggests that PPDM specific services be offered as services built on the K-Grid architecture [15], [16]. While the authors do not provide details on how this would be implemented, or communication framework developers would use, we do adapt the approach of providing PPDM services in our system.

Domain specific languages for SMC have the potential to free the user from the details of the execution and implementation. Fairplay [26] is a domain specific language for secure computation. Fairplay generates secure circuits in a Secure Hardware Description Language (SHDL) and then executes those circuits.

While domain specific languages can potentially ease the development of PPDM algorithms, they also have drawbacks. The new domain specific systems present a problematic learning curve to developers who are trained in the use of standard languages. In addition, systems that automatically generate SMC algorithms, are also implicitly parallelizing the computation; automated parallelization can only find mechanical ways of distributed the computation and does not make decisions to refactor the overall computation in more efficient ways. The use of specialized languages can inhibit developers from leveraging existing code for PPDM projects. Finally, the SMC languages surveyed do not offer the ability to leverage high-performance computing resources available to organizations. Therefore, their scaling can be limited for large data mining applications.

IV. APHID

Reviewing the literature and available software to support PPDM, it is clear that there is a significant barrier to developing PPDM applications. The first of these impediments is that there are no standardized libraries to support PPDM. Secondly, organizations would need a middleware framework to support PPDM, which is not sufficiently provided in current systems [27], [15], [18]. Even with the availability of such frameworks, simple development environments are lacking; it is especially difficult to integrate the PPDM level of mining with the use of local high-performance computing resources (e.g. grid, clusters and specialized hardware). APHID (Architecture for Private and High-performance Integrated Data mining) seeks to overcome these limitations. The design is influenced by several desiderata, which have been explicitly identified in the literature or found lacking in other systems. The system must have:

- Low development cost [20]
- A runtime environment for executing algorithms (as in [27], [15], [18], [22], [25])
- The capability to leverage high-performance computing resources whenever possible (as in [12], [14] and others)
- Flexibility to support an array of PPDM algorithms (emphasized in [19])
- Support many popular languages (addressing the emphasis for portability in [20])

- Simple abstractions to mitigate the complexity of PPDM development

To support low development cost and language independence, DDM/PPDM functions are provided as a collection of web services (as suggested in [15], [18], [22], [25]), which can be called by the application program. To begin with, web services libraries are available for almost every popular language in use, so the services can be implemented in any language or platform, and consumed by a different language or platform. Provide a set of frequently used services reduces development effort and hence cost, for implementing a new algorithm. In charge of these services is a set of master services, the Main Execution Layer (MEL), discussed in section IV-C. MEL orchestrates the execution of the PPDM algorithm, providing the necessary runtime environment.

APHID is explicitly built on a two-tier system of PPDM, which differentiates it from other systems. On the first tier, different organizations (also called parties in the PPDM context) communicate with each other, typically using secure, privacy preserving communications. The second tier includes grids and clusters within a particular party. Treating these tiers distinctly helps the developer to manage the complexities inherent in each level (see figure 1 for an illustration).

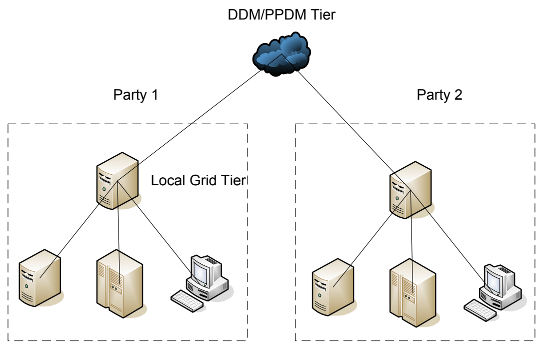

Fig. 1. A two tier PPDM architecture.

The interface to the local high performance machines is also provided as a set of web services for the individual functions in the algorithm. Therefore, an algorithm can be developed once and shared among all of the parties, with the developers at each individual party providing only what is necessary to interface with the party's database and high performance machines. These services support the requirement of leveraging HPC resources.

Figure 2 shows the stack of systems comprising a typical APHID installation within a single party. Organizational data to be mined is frequently stored in a relational database server. Because a relational database manager is typically insufficient for flexible data mining, and because these servers are often intimately involved in core business processes, this data is converted and transfered to a high-performance distributed file system (e.g. HDFS [28], or grid-based storage). This synchronization should be done periodically and at off peak times, before it is needed for a data mining process.

The PPDM process begins with a request from a client, typically as part of a larger application, for the output of a specific PPDM algorithm (e.g. a classified test point, a classifier model, a set of clusters, or a set of association rules). The algorithms are available by unique services representing the algorithm, a partitioning (vertical, horizontal, or arbitrary) and a specific implementation.

While the MEL is responsible for initiating the PPDM process, it will frequently need to use SMC-PPDM primitives (e.g. secure sum, secure scalar products) and perform compute and memory intensive operations on training data. The PPDM services layer and the High-Performance Computing (HPC) respectively support these needs. The PPDM services layer acts as a gateway to external parties. The HPC services layer is a generic interface that interacts with a pluggable set of cluster and grid runtime systems (e.g. MapReduce) to perform the local mining of the database which will become part of the larger PPDM algorithm. It will store and access training databases, and submit compute-intensive jobs through the appropriate channels. Having these broad collections of service-based functions available, meets APHID's requirements for flexibility.

Before describing the functions of each layer in detail, a notation of analysis must first be established. Then, the development model around which APHID is structured is described. Finally, each of the three layers is described in detail, aided by an example of the code that would be found at each layer. The code is for the horizontally-partitioned Naïve Bayes classifier for categorical attributes [7].

A. Notation

To aid in our analysis, we first establish a notation. During a PPDM operation there are K parties involved, numbered $P^1...P^k...P^K$. In the case of horizontal partitioning, the D-dimensional training data, $\mathbf{X}$, are divided among the parties, in some way, such that each party P^k owns several D-dimensional instances of the training data, with that set designated as $\mathbf{X^k}$ and individual points labeled X_r^k.

B. Development Model

Before focusing more closely on each layer of APHID, a development model must first be established to provide a simple yet powerful abstraction for PPDM development. The cornerstones of APHID development are a program style similar to many HPC frameworks, and policy-attached shared variables, which mitigate complexity and cost.

1) Program Structure: In order to bridge computations on a grid or cluster with DDM/PPDM computations, a simplified interface is needed. Programming hundreds or thousands of machines of a cluster, typically on local networks, along with remote DDM/PPDM sites, typically connected on the Internet, has the potential to significantly confuse a developer. To simplify development, at the cluster/grid level, parallel development environments like MapReduce are used. At the DDM/PPDM level, an Single Program Multiple Data (SPDM) style is used. SPMD is the same programming style used in implementations of the Message Passing Interface

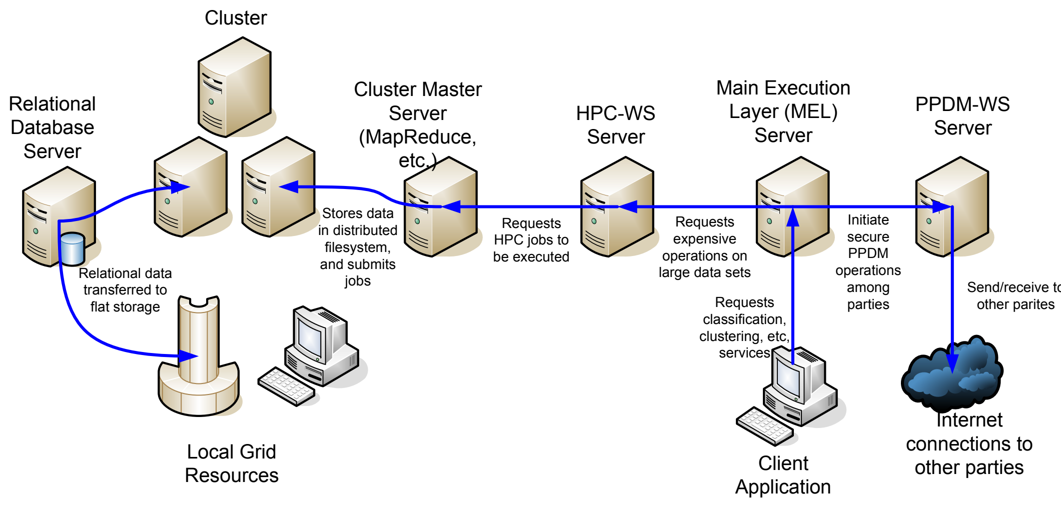
Cluster
Relational Database Server
Cluster Master Server (MapReduce, etc.)
HPC-WS Server
Main Execution Layer (MEL) Server
PPDM-WS Server
Stores data in distributed filesystem, and submits jobs
Requests HPC jobs to be executed
Requests expensive operations on large data sets
Initiate secure PPDM operations among parties
Send/receive to other parites
Relational data transferred to flat storage
Requests classification, clustering, etc. services
Local Grid Resources
Client Application
Internet connections to other parties

Fig. 2. A typical APHID system stack.

(MPI) [29], which is a popular development environment for distributed programming. The SPMD style is appropriate because all parties should be able to examine the operations involved in a PPDM algorithm. For each PPDM algorithm, there should exist one copy of the code that all parties can examine, thereby ensuring security.

2) Shared Variables: One technique APHID employs to simplify PPDM application development is the use of shared variables. These variables work similarly to those of traditional shared memory systems, with the exception that they have a particular *policy* attached. A policy determines how and by whom the value may be accessed.

The available policies are shown in table I. The first policy Intra-Party (IP) creates an intra-party shared variable only, which is simply a handle that allows the data to be passed when needed from machine to machine in the stack. The second policy, Fully Shared (FS), creates variables for which the unmodified value can be passed among parties. Finally the Secret Shared (SS) policy creates variables for which disparate shares are split among several parties. One example of this kind of sharing is the result of a shared secure scalar product, as in [30]. At the end of this operation, the two participating parties each have shares of the final scalar product value, which are each indistinguishable from random but whose sum is the full result of the operation.

TABLE I
TYPES OF VARIABLE SHARING POLICIES.

Policy	Description
Intra-Party (IP)	Only shared within a party, among layers of the PPDM stack.
Fully Shared (FS)	Represents a variable with shared read and/or write access between at least two parties.
Secret Shared (SS)	Represents a variable where independent shares are given to two or more parties, which combined yield the final result.

The shared variables with policies afford several advantages. First, it makes the sharing and broadcasting of values among parties relatively transparent. An example is given in figure 3. All examples are given in language neutral pseudocode to reflect that they should be implementable in any popular language. Line 1 declares a shared variable, and line 2 attaches a policy: in this case, the V_{shared} is fully shared between party P_1 and party P_2. In lines 3–4, executed by party P_1, a value of 1 is written to V_{shared}. When in lines 3–4, the value of V_{shared} is read by P_2, P_2 automatically requests and caches that value.

```
1 float $V_{shared}$;
2 setPolicy($V_{shared}$, $Policy_{FS}(P_1, P_2)$);
3 if $P == P_1$ then
4     $V_{shared}$ = 1;
5 else if $P == P_2$ then
6     $R = 1 + V_{shared}$;
7 else if $P == P_3$ then
8     $R = 2V_{shared}$;
```

Fig. 3. Example shared variable usage

One of the primary advantages behind shared variables with policies is that they offer automatic checking for authorization. Figure 3 also gives an example of this scenario. If code is accidentally written such that it tries to access a variable locked onto another party (lines 7–8). Upon trying to retrieve this value, the runtime will throw a security exception, and the execution will typically be halted.

C. Main Execution Layer

The Main Execution Layer (MEL) is itself a collection of services. These are the high level services that compromise the full data mining algorithms themselves (e.g. Naïve Bayes, k-NN, SVM, etc.) which are then easily integrated into higher-level applications. The MEL also consists of the processes that are responsible for directing the execution of the DDM/PPDM algorithm.

```
Input: Points in $\mathbf{X} = X_r \forall r = 1...n$
Output: Probabilities $p_{yz}$ of an instance with class $y$
        and attribute value $z$.
1  Vector< $float$ > $\mathbf{c^k}$;
2  setPolicy($\mathbf{c^k}$,$Policy_{IP}$);
3  $\mathbf{c^k}$ = categoryAttributeCounts($\mathbf{X^k}$);
4  Vector< $float$ > $\mathbf{c}$;
5  setPolicy($\mathbf{c}$,$Policy_{FS}(P^1)$);
6  $\mathbf{c}$ = secureSum($P^*$, $\mathbf{c^k}$);
7  Vector< $float$ > $\mathbf{n^k}$;
8  setPolicy($\mathbf{n^k}$,$Policy_{IP}$);
9  $\mathbf{n^k}$ = categoryCounts($\mathbf{X^k}$);
10 Vector< $float$ > $\mathbf{n}$;
11 setPolicy($\mathbf{n}$, $Policy_{FS}(P_1)$);
12 $\mathbf{n}$ = secureSum($P^*$, $n^k$);
13 foreach $(y, z)$ do
14     $p_{yz} = c_{yz}/n_y$;
```

Fig. 4. Main service to implement PPDM Naïve Bayes classifier [7]

The algorithm in figure 4 represents the portion of the Naïve Bayes algorithm which the MEL is responsible for executing. In lines 1–2 of the algorithm, a variable is created, only accessible to the local PPDM stack, which stores a vector of categorical/attribute pairs and how frequently they occur. The service $categoricalAttributeCounts$ one line 3 is called from HPC services. This value is then passed to PPDM services (line 6), where is added to the final value c by secure sum, through a variable which can only be accessed by P^1. In lines 7–12, similar actions are taken to obtain the overall counts of points in each category, which are again summed and given to P^1. Finally, the output probabilities p_{yz} are determined using the calculated values.

D. High-Performance Computing Services

For interfacing with the training databases, and for resource intensive computing conducted during the PPDM process, the High-Performance Computing web services (HPC-WS) provide a generic interface to this functionality. The HPC layer can be adapted by each party to interface with their specific HPC installation, which can include clusters, grids and specialized hardware.

The algorithm given in figure 5 represents a portion of the Naïve Bayes algorithm which executes in the HPC layer. This code is in the form of a MapReduce program, which would interface with a cluster with the training database $\mathbf{X}$ distributed in cluster storage. The *map* procedure in lines 1–4 take each training point as the value, and its index r as the key. The map phase simply counts (y, z) pairs of attributes and class labels for each training point. In the *reduce* phase (lines 6–8), these are summed together into a hashtable responsible for keeping the counts of (y, z) pairs. A vector of (y, z) pairs and their associated counts are returned to the calling application within the MEL.

```
Input: Points in $\mathbf{X} = X_r \forall r = 1...n$
Output: Vector of $(y, z)$ pairs and associated counts
1 HashTable $H$;
2 map($k_1 = r$, $v_1 = X_r$) begin
3     foreach $z \in X_r$ do
4         collect($(y, z)$, 1);
5 end
6 reduce($k_2 = (y, z)$, $\mathbf{v_2}$) begin
7     $H(y, z) += \sum \mathbf{v_2}$;
8 end
```

Fig. 5. MapReduce pseudo-code for *categoryAttributeCounts*

E. PPDM Services

The PPDM Services (PPDM-WS) is responsible for both providing primitive DDM/PPDM operations (e.g. secure sum), but also for providing the send, receive and peer finding operations on which those operations are built. Developing this set of services for PPDM is efficient, because most popular SMC-based PPDM algorithms tend to utilize a small set of SMC operations. By providing a toolkit of frequently used operations, as suggested in [5], developers can easily implement numerous PPDM algorithms. Table II lists algorithms which utilize popular SMC operations.

TABLE II
EXAMPLES OF SUPPORTED OPERATIONS AT THE PPDM LEVEL.

Operation	Reference
Secure Sum [31]	[9], [32]
Secure Scalar Product [30]	[33], [34], [35], [36], [32]
Yao Circuits [3]	[33], [37], [35]
Oblivious Transfer [38]	[7], [6]

Figure 6 contains the algorithm for secure sum [31], implemented in the PPDM services layer to support the Naïve Bayes algorithm. To simplify the pseudo-code, it is assumed that the result should end up on party P^1. P^1 starts by creating a random number (line 2) adding it to its value v^1 and sending it along (line 3). Meanwhile, parties P^2 to P^K receive what their previous neighbor is sending, add it to their respective value, and send it along (lines 5–6). Finally, P^1 receives the sum from P^K, subtracts the random number, and obtains the final sum v.

V. Implementation

The APHID prototype is implemented in Java, using Apache Axis2 [39] to handle web services communication. The MEL, PPDM-WS and HPC-WS layers are implemented as Axis2 modules, contained within a Tomcat [40] application container. The cluster-based MapReduce functionality is provided by Hadoop [28].

VI. Conclusions

Through analyzing the shortcomings of available PPDM runtime architectures and development systems, we designed a system capable of overcoming those challenges. As the

```
Input: Parties $P^1$ through $P^K$, each with values $v^1$
    through $v^k$ respectively
Output: The sum $v = \sum_{k=1}^{K} v^k$
if $P == P^1$ then
    $R = rand() \, mod \, F$;
    send($P^2$, $(v^1 + R) \, mod \, F$);
    $v$ = receive($P^K$) $- R \, mod \, F$;
else
    send($P^{k \, mod \, (K+1)}$,
    receive($P^{k-1}$) $+ v^k \, mod \, n$);
```

Fig. 6. PPDM service for the secure sum [31].

field of PPDM continues to evolve, yielding new algorithms, SMC techniques, and support for high-performance computing, APHID will continue to evolve as well.

References

[1] J. Vaidya and C. Clifton, "Privacy-preserving data mining: Why, how, and when," *IEEE Security and Privacy*, vol. 2, no. 6, pp. 19–27, 2004.

[2] S. R. M. Oliveira and O. R. Zaane, "Toward standardization in privacy-preserving data mining," in *In Proc. of the 3nd Workshop on Data Mining Standards (DM-SSP 2004), in conjuction with KDD 2004*, 2004, pp. 7–17.

[3] A. Yao, "How to generate and exchange secrets," in *Proc. 27th Annual Symposium on Foundations of Computer Science*, 1986, pp. 162–167.

[4] O. Goldreich, S. Micali, and A. Wigderson, "How to play any mental game," in *STOC '87: Proceedings of the nineteenth annual ACM conference on Theory of computing*. New York, NY, USA: ACM Press, 1987, pp. 218–229.

[5] C. Clifton, M. Kantarcioglu, J. Vaidya, X. Lin, and M. Y. Zhu, "Tools for privacy preserving distributed data mining," *SIGKDD Explorations*, vol. 4, no. 2, pp. 28–34, 2003.

[6] B. Pinkas, "Cryptographic techniques for privacy-preserving data mining," *SIGKDD Explor. Newsl.*, vol. 4, no. 2, pp. 12–19, 2002.

[7] M. Kantarcoglu and J. Vaidya, "Privacy preserving naive bayes classifier for horizontally partitioned data," in *IEEE ICDM Workshop on Privacy Preserving Data Mining*, Melbourne, FL, November 2003, pp. 3–9.

[8] J. Vaidya and C. Clifton, "Privacy-Preserving K-Means Clustering over Vertically Partitioned Data," in *The Ninth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, Washington, DC, August 2003. [Online]. Available: http://www.cs.purdue.edu/homes/jsvaidya/pub-papers/vaidya-kmeans.pdf

[9] H. Yu, J. Vaidya, and X. Jiang, "Privacy-preserving svm classification on vertically partitioned data," in *Pan-Asia Conference on Knowledge Discover and Data Mining (PAKDD)*, Singapore, 2006, pp. 647–656.

[10] J. Zhan, L. Change, and S. Matwin, "Privacy preserving k-nearest neighbor classification," *International Journal of Network Security*, vol. 1, no. 1, 2005.

[11] J. Vaidya and C. Clifton, "Privacy preserving association rule mining in vertically partitioned data," in *In The Eighth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, July 2002.

[12] J. Dean and S. Ghemawat, "Mapreduce: Simplified data processing on large clusters," in *In Proceedings of OSDI'04: Sixth Symposium on Operating System Design and Implementation*, December 2004.

[13] C.-T. Chu, S. K. Kim, Y.-A. Lin, Y. Yu, G. Bradski, A. Y. Ng, and K. Olukotun, "Map-reduce for machine learning on multicore," in *In the Proceedings of NIPS 19*, 2006.

[14] S. Parthasarathy and R. Subramonian, "Facilitating data mining on a network of workstations," in *Advances in Distributed and Parallel Knowledge Discovery*. AAAI Press, 2000.

[15] M. Cannataro, A. Congiusta, A. Pugliese, D. Talia, and P. Trunfio, "Distributed data mining on grids: services, tools, and applications," *IEEE Transactions on Systems, Man and Cybernetics, Part B*, vol. 34, no. 6, pp. 2451–2465, 2004.

[16] M. Cannataro, A. Congiusta, D. Talia, and P. Trunfio, "A data mining toolset for distributed high-performance platforms," in *Proceedings of Data Mining 2002*, W. I. Press, Ed., Bologna, Italy, 2002.

[17] I. Foster and C. Kesselman, "Globus: A metacomputing infrastructure toolkit," *International Journal of Supercomputer Applications*, vol. 11, no. 2, pp. 115–128.

[18] V. Stankovski, M. Swain, V. Kravtsov, T. Niessen, D. Wegener, J. Kindermann, and W. Dubitzky, "Grid-enabling data mining applications with datamininggrid: An architectural perspective," *Future Generation Computer Systems*, 2007.

[19] J. M. P. na and E. Menasalvas, "Towards flexibility in a distributed data mining framework," in *Proc. DMKD Workshop*, 2001.

[20] M. Z. Ashrafi, D. Taniar, and K. Smith, "A data mining architecture for distributed environments," in *Second International Workshop on Innovative Internet Computing Systems*, June 2002, pp. 27–38.

[21] A. Kumar, M. Kantardzic, P. Ramaswamy, and P. Sadeghian, "An extensible service oriented distributed data mining framework," in *Proc. 2004 International Conference on Machine Learning and Applications*, December, 2004, pp. 256–263.

[22] W. Ahmad and A. Khokhar, "Triumf: A trusted middleware for fault-tolerant secure collaborative computing," University of Illinois at Chicago Tech Report, Tech. Rep. TR-MSL0786, August 2006.

[23] "Jxta technology," http://www.sun.com/software/jxta/, 2008.

[24] J. Wang, C. Xu, H. Shen, , and Y. Pan, "Hierarchical infrastructure for large-scale distributed privacy-preserving data mining," in *In Proc. of the 5th International Conference on Computer Science*, May 2005, pp. 1020–1023.

[25] W. Ahmad and A. Khokhar, "Towards secure and privacy preserving data mining over computational grids," in *In Proceedings of the NSF International Workshop on Frontiers of Information Technology*, December 2003.

[26] D. Malkhi, N. Nisan, B. Pinkas, and Y. Sella, "Fairplay - a secure two-party computation system," in *Proc. of the USENIX Security Symposium*, 2004, pp. 287–302.

[27] S. Bailey, R. Grossman, H. Sivakumar, and A. Turinsky, "Papyrus: A system for data mining over local and wide area clusters and super-clusters," in *1999 ACM/IEEE conference on Supercomputing*. Portland, OR: ACM Press, 1999.

[28] "Welcome to hadoop!" http://lucene.apache.org/hadoop/, 2007.

[29] "Mpi forum," http://www.mpi-forum.org/, 2008.

[30] B. Goethals, S. Laur, H. Lipmaa, and T. Mielikäinen, "On private scalar product computation for privacy-preserving data mining." in *Information Security and Cryptology - ICISC 2004, 7th International Conference, Seoul, Korea, December 2-3, 2004, Revised Selected Papers*, ser. Lecture Notes in Computer Science, C. Park and S. Chee, Eds., vol. 3506. Springer, 2004, pp. 104–120.

[31] B. Schneier, *Applied Cryptography*, 2nd ed. John Wiley & Sons, 1995.

[32] J. Secretan, M. Georgiopoulos, and J. Castro, "A privacy preserving probabilistic neural network for horizontally partitioned databases," Aug. 2007.

[33] G. Jagannathan, K. Pillaipakkamnatt, and R. Wright, "A new privacy-preserving distributed k-clustering algorithm," in *Proceedings of the 2006 SIAM International Conference on Data Mining (SDM)*, 2006.

[34] Z. Yang and R. N. Wright, "Improved privacy-preserving bayesian network parameter learning on vertically partitioned data," in *ICDEW '05: Proceedings of the 21st International Conference on Data Engineering Workshops*. Washington, DC, USA: IEEE Computer Society, 2005, p. 1196.

[35] G. Jagannathan and R. N. Wright, "Privacy-preserving distributed k-means clustering over arbitrarily partitioned data," in *KDD '05: Proceeding of the eleventh ACM SIGKDD international conference on Knowledge discovery in data mining*. New York, NY, USA: ACM Press, 2005, pp. 593–599.

[36] H. Yu, X. Jiang, and J. Vaidya, "Privacy-preserving svm using non-linear kernels on horizontally partitioned data," in *Selected Areas in Cryptography*, Dijon, France, 2006.

[37] Y. Lindell and B. Pinkas, "Privacy preserving data mining," *Journal of Cryptology*, vol. 15, no. 3, 2002.

[38] M. Naor and B. Pinkas, "Oblivious transfer and polynomial evaluation," in *STOC '99: Proceedings of the thirty-first annual ACM symposium on Theory of computing*. New York, NY, USA: ACM Press, 1999, pp. 245–254.

[39] "Apache axis2," http://ws.apache.org/axis2/, 2008.

[40] "Apache tomcat," http://tomcat.apache.org/, 2009.