Cloud Colonography: Distributed Medical Testbed over Cloud

Yuichi Motai, *Senior Member, IEEE*, Eric Henderson, Nahian Alam Siddique, *Student Member, IEEE*, and Hiroyuki Yoshida, *Member, IEEE*

Abstract—Cloud Colonography is proposed in this paper, using different types of cloud computing environments. Databases from the Computed Tomographic Colonography (CTC) screening tests among several hospitals are explored. These networked databases are going to be available in the near future via cloud computing technologies. Associated Multiple Databases (AMD) was developed in this study to handle multiple CTC databases. When AMD is used for assembling databases, it can achieve very high classification accuracy. The proposed AMD has the potential to play a role of the core classifier in the cloud computing framework. AMD for multiple institutions databases yields high detection performance of polyps using Kernel principal component analysis (KPCA). Two cases in the proposed cloud platform are private and public. We adapted a university cluster as a private platform, and Amazon Elastic Compute Cloud (EC2) as a public. The computation time, memory usage, and running costs were compared using three representative databases between private and public cloud environments. The proposed parallel processing modules improved the computation time, especially in the public cloud environments. The successful development of a cloud computing environment that handles large amounts of data will make Cloud Colonography feasible for a new health care service.

Index Terms—Cloud service, health care, computed tomographic colonography, distributed databases, kernel feature analysis, group learning, cloud computing

1 Introduction

The most prominent limiting factor pertaining to widespread usage of the latest screening technology as a replacement for traditional screening colonoscopy is the limited supply of internal CTC images available at a single medical institution. Such a limited supply of CTC training images significantly constrains the accuracy of an automated detection algorithm. Many medical institutions employ a limited number of CTC specialists, whose CTC image analysis skills vary widely [1], [2]. To overcome these difficulties while providing a high detection performance of polyps, computer-aided detection (CAD) schemes have been investigated which semi-automatically detect suspicious lesions in CTC images [4], [75].

CAD schemes have been developed for medical institutions (i.e., hospitals), where the number of training instances used during automated diagnosis is determined by resources and training requirements [5], [6], [8], [9], [10]. In most clinical settings, if more training data are collected after an initial polyp model is computed, retraining of the model becomes imperative to incorporate the newly-added data from the local institution, and to preserve the classification accuracy [5], [6], [12], [13]. This classification accuracy may be proportionately related to the amount of training data available, i.e., more training data may yield a higher degree of classification. With this in mind, we propose a new framework, called "Cloud Colonography", where the colonography database at each institution may be shared and/or uploaded to a single composite database on a cloud server. The CAD system at each institution can then be enhanced by incorporating new data from other institutions through the use of the distributed learning model proposed in this study.

The concept of collaborative cloud applications using distributed databases has been discussed in [59], [60], [61], but not yet in the context of CAD in Cloud Colonography; thus, the presented work is a first attempt at such a study. Therefore, the proposed Cloud Colonography framework shown in Fig. 1 requires a comprehensive study to determine whether the overall performance is improved by using multiple databases. However, it is worth noting that, these existing studies showed that the overall classification performance for larger multiple databases was improved in practical settings [54].

The primary contribution of this study is to develop Cloud Colonography using Associated Multiple Databases (AMD) to represent the statistical data characteristics in both private and public clouds. The proposed Cloud Colonography is capable of learning multiple databases provided by multiple institutions. We propose composite kernel feature analysis with multiple databases for this specific problem. Kernel feature analysis is widely used for data compression, clustering/classification [58]. The improvement in performance could be obtained by employing such efficient cloud computing approaches in terms of resources, infrastructure and operational costs. Comapred

- *Y. Motai, E. Henderson, and N.A. Siddique are with the Department of Electrical and Computer Engineering, Virginia Commonwealth University, Richmond, VA 23284. E-mail: {ymotai, hendersonec, siddiquena}@vcu.edu.*
- *H. Yoshida is with the Department of Radiology, Massachusetts General Hospital and Harvard Medical School, Boston, MA 02115. E-mail: Yoshida.Hiro@mgh.harvard.edu.*

Manuscript received 2 Dec. 2014; revised 12 May 2015; accepted 31 Aug. 2015. Date of publication 23 Sept. 2015; date of current version 9 June 2020. Recommended for acceptance by J. He, R. Kotagiri, and F.M. Sanchez. Digital Object Identifier no. 10.1109/TCC.2015.2481414

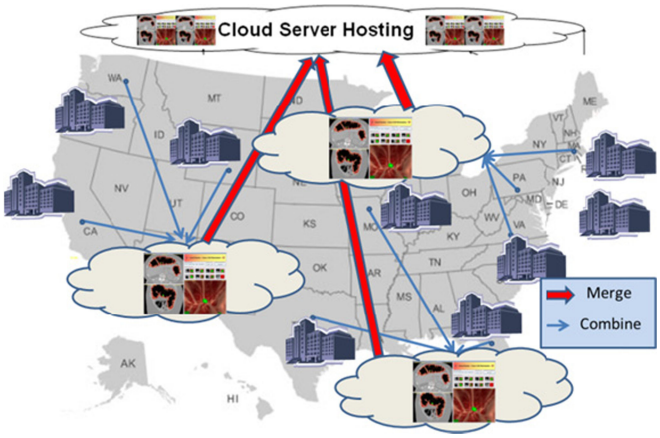

Fig. 1. Concept of cloud colonography with distributed image databases for colon cancer diagnosis. The cloud server hosting will collect distributed databases from different institutions and group them by assembling database analysis with attributes.

to other work [69], [70], [71], [72], the proposed AMD improved computation time. The database acquired over a long period of time can sometimes be highly diverse. The parallelization in the cloud environment for multiple databases efficiently utilizes the computational capability of public servers.

The rest of this paper is organized as follows. Section 2 provides an introduction to cloud environment and a brief review of the existing feature extraction methods. Section 3 describes the proposed AMD for Cloud Colonography among multiple databases, Section 4 evaluates the experimental results for classification performance, Section 5 accesses cloud computing performance, while conclusions are drawn in Section 6.

2 Cloud Environments

Cloud Colonography proposes new services using a platform as a service (PaaS) model [60], [61]. Cloud Colonography is a cloud computing platform for programming, databases, and web servers. The clinical users access Cloud Colonography via the proposed software on a cloud platform without the cost and complexity of buying and managing the underlying hardware and software layers. Using existing PaaS platforms like Microsoft Azure and Google App Engine, the underlying computer and storage resources scale automatically to match application demand so that the cloud user does not have to allocate resources manually.

Section 2.1 evaluates the existing cloud environment and effectiveness with respect to the proposed method. Section 2.2 introduces the relevant cloud framework, showing how the mathematical background on Kernel principal component analysis (KPCA) are used for AMD.

2.1 Server Specifications of Cloud Platforms

The cloud vendor specification is shown in Table 1. In addition to the private cloud environment using the intranet, the public cloud vendors using the internet are considered to implement the proposed Cloud Colonography.

The leading commercial cloud hosting service, Amazon, is chosen because of its superiority in performance as listed in Table 1. Microsoft and IBM do not support Matlab programming environment. Therefore, we decided to adopt Amazon as a public cloud platform for the implementation of Cloud Colonography. Private Cloud [67] is a department cluster used for a private cloud platform, and as a comparison, we added a desktop PC as well. The effectiveness of Cloud Colonography mainly comes from how many datasets are manageable and the computational capacity of those large datasets. The proposed KPCA with AMD is a solution for these main problems. In the next section, we will describe the mathematical background of KPCA in the cloud environment.

TABLE 1
Representative Cloud Vendors

Vendors	Monthly Uptime (%)	Credit level max (%)	RAM per CPU (GB)	APIs Supports
Amazon [62]	99.95	30	0.615	REST, Java, IDE, Command line
Microsoft [63]	99.95	25	1.75	REST, Java, IDE, Command line
IBM [64]	99.00	10	2.00	REST
Private Cloud [67]	NA	NA	64	–
Desktop PC	NA	NA	16	–

2.2 Cloud Framework of KPCA for AMD

For efficient feature analysis, extraction of the salient features of polyps is essential because of the huge data size from many institutions and the 3D nature of the polyp databases [4]. The desired method should allow for the compression of such big data. Moreover, the distribution of the image features of polyps is nonlinear [34]. To address these problems, we adopt the kernel approaches [58] in a cloud platform.

The key issue of kernel feature analysis is how to select a nonlinear, positive-definite kernel $K : R^d \times R^d \rightarrow R$ of an integral operator in the d-dimensional space. The kernel K, which is a Hermitian and positive semi-definite matrix, calculates the inner product between two finite sequences of inputs $\{x_i : i \in n\}$ and $\{x_j : j \in n\}$, defined as $K :== (K(x_i, x_j)) = (\Phi(x_i).\Phi(x_j) : i, j \in n)$, where x is a grey-level CTC image, n is the number of image database, and $\Phi : R^d \rightarrow H$ denotes a nonlinear embedding (induced by K) into a possibly infinite dimensional Hilbert space H as shown in Fig. 2. A more thorough discussion of kernels can be found in [28]. Our AMD module for CTC images is a dynamic extension of KPCA as described below.

KPCA uses a Mercer kernel [37] to perform a linear principal component analysis of the transformed image. Without loss of generality, we assume that the image of the data has been centered so that its scatter matrix in S is given by $S = \sum_{i=1}^{n} \Phi(x_i)(x_i)\Phi(x_i)^T$. The eigenvalues λ_j and eigenvectors e_j are obtained by solving the following equation, $\lambda_j e_j = S e_j = \sum_{i=1}^{n} \Phi(x_i)\Phi(x_i)^T e_j = \sum_{i=1}^{n} < e_j, \Phi(x_i) > \Phi(x_i)$. If K is an $n \times n$ Gram matrix, with the elements $k_{ij} = \langle \Phi(x_i), \Phi(x_j) \rangle$ and $a_j = [a_{j1}, a_{j2}, \ldots, a_{jn}]$ as the eigenvectors associated with eigenvalues λ_j, the dual eigenvalue problem equivalent to the problem can be expressed as follows: $\lambda_j a_j = K a_j$.

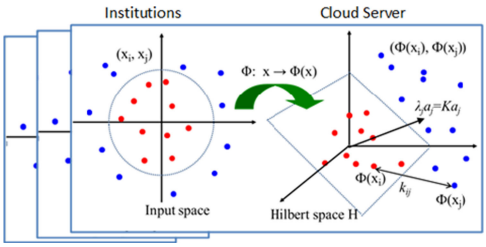

Fig. 2. Illustration of the mathematical background of KPCA. KPCA calculates the eigenvectors and eigenvalues by analyzing the kernel feature space of multiple institutions so that a cloud server can handle larger datasets.

The traditional KPCA can be extended into the cloud framework as follows:

1. Compute the Gram matrix that contains the inner products between pairs of image vectors.
2. Configure AMD by grouping datasets from multiple institutions.
3. Solve: $\lambda_j a_j = K a_j$ to obtain the coefficient vectors a_j for $j = 1, 2, \ldots, n$.
4. The projection of a test point $x \in R^d$ along the j-th eigenvector is $< e_j, \Phi(x) > = \sum_{i=1}^{n} a_{ji} < \Phi(x_i), \Phi(x) > = \sum_{i=1}^{n} a_{ji} k(x, x_i)$.

The above implicitly contains an eigenvalue problem of rank n; therefore, the computational complexity of KPCA is $O(n^3)$. Each resulting eigenvector can be expressed as a linear combination of n terms. The total computational complexity is given by $O(ln^2)$ where l stands for the number of features to be extracted and n stands for the rank of the Gram matrix K [28], [33]. Once we have our Gram matrix ready, we can apply these algorithms to our database to obtain a higher dimensional feature space. This cloud-based kernel framework will be discussed further in the following sections.

3 AMD for Cloud Colonography

This section proposes the AMD method. Section 3.1 shows the overall concept of AMD for cloud environment, Section 3.2 explains more details on the design of AMD, Section 3.3 shows the two implementation cases for the selected public cloud environments, and Section 3.4 shows the proposed parallelization.

3.1 AMD Concept

As in Fig. 2, KPCA mentioned is executed for Cloud Colonography by analyzing the images of polyps with nonlinear big feature space. We apply KPCA to both an individual dataset and groups of datasets. KPCA can be used in conjunction with AMD to synthesize individual databases into larger composite databases as shown in Fig. 3.

The concept of AMD is to format distinct distributed databases into a uniform larger database for the proposed Cloud Colonography platform. AMD are expected to solve the issues of nonlinearity and excessive data size so that CAD classification can achieve optimal performance. Specifically, kernels handle nonlinear to linear projection, and PCA handles data size reduction. The next section will describe how our proposed kernel framework will be organized for AMD.

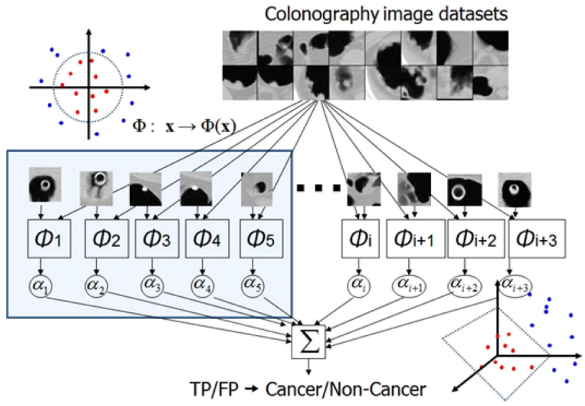

Fig. 3. A concept of AMD. The proposed kernel framework combines the Cloud Colonography datasets by analyzing the images of polyps with nonlinear big feature space.

3.2 Data Configulation of AMD

We adapt Community Cloud [62] for Cloud Colonography to share infrastructure between several hospitals with common CTC domains (data compliance and security, HIPPA agreement, etc.). The costs are spread over fewer users than a public cloud (but more than a private cloud); therefore, only some cost savings are expected.

Fig. 4 illustrates the AMD construction by analyzing the data from other individual institutional databases through the four representative steps. Step 1 splits each database, Step 2 combines several databases, Step 3 sorts the combined databases, and Step 4 merges the sorted databases. We will explain individual steps in the following:

Step 1: Split

We split each database by maximizing the Fisher scalar for kernel optimization [57]. The Fisher scalar is used to measure the class separability J of the training data in the mapped feature space. It is formulated as $J = trace(\sum_l S_{bl})/trace(\sum_l S_{wl})$, where S_{bl} represents "between-class scatter matrices", and S_{wl} represents "within-class scatter matrices." According to [35], the class separability by Fisher scalar can be expressed by using the basic kernel matrix P^l

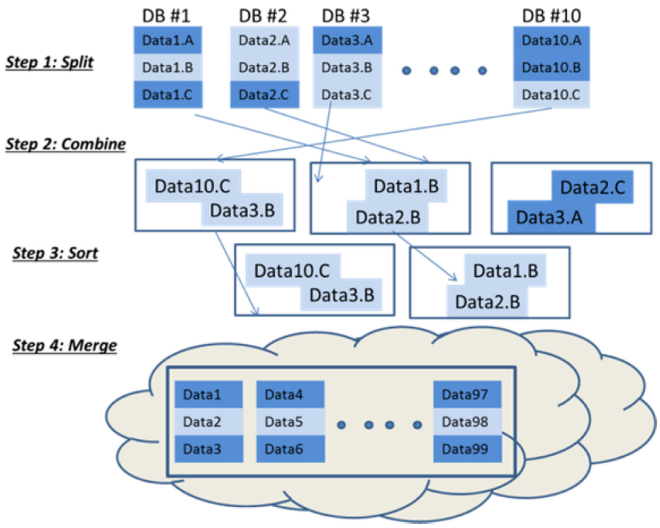

Fig. 4. Four representative steps of AMD. The proposed AMD consists of the four main criteria to manage databases by 1) Split, 2) Combine, 3) Sort, and 4) Merge.

(submatrices of $P_{11}^l, P_{12}^l, P_{21}^l$, and P_{22}^l) as $J_l(a_l) = a_l^T M_l a_l / a_l^T N_l a_l, where\ M_l = K_l^T B_l K_l, N_l = K_l^T W_l K_l$, and

$$W_1 = diag(P_{11}^1, P_{22}^1) - \begin{pmatrix} P_{11}^l/n_1 & 0 \\ 0 & P_{22}^l/n_2 \end{pmatrix}$$

$$B_l = \begin{pmatrix} P_{11}^l/n_1 & 0 \\ 0 & P_{22}^l/n_2 \end{pmatrix} - P_l/n.$$

To maximize $J_l(a_l)$, the standard gradient approach is followed. If the matrix N_{0i} is nonsingular, the optimal a_l which maximizes the $J_l(a_l)$ is the eigenvector that corresponds to the maximum eigenvalue of $M_l a_l = \lambda_l N_l a_l$. The criterion to select the best kernel function is to find the kernel which produces the largest eigenvalue

$$\lambda_{l*} = \arg\max_{\lambda_l}(N_l^{-1} M_l).$$

Choosing the eigenvector that corresponds to the maximum eigenvalue can maximize the $J_l(a_l)$ to achieve the optimum solution. Once we determine the eigenvalues, the eigenvectors associated with the selected eigenvalues represent each split dataset.

Step 2: Combine

Alignment is used to measure the adaptability of a kernel to the target data. Alignment provides a practical objective for kernel optimization whether datasets are similar or not. The alignment measure is defined as a normalized Frobenius inner product between the kernel matrix and the target label matrix introduced by Cristianini et al. [45]. The empirical alignment between the two kernels with respect to the training set S is given as:

$$Frob(K_1, K_2) = \langle K_1, K_2 \rangle_F / \|K_1\|_F \|K_2\|_F,$$

where K_i is the kernel matrix for the training set S. $\langle K_i, K_j \rangle_F$ is the Frobenius inner product between K_i and K_j. It has been shown that if datasets chosen are well aligned with the other datasets, these datasets are considered as combined [45].

Step 3: Sort

We use class separability as a measure to identify whether the combined data is configured correctly in the right order. The separability within the combined datasets is required to check how well the data is sorted. It can be expressed as the ratio of separabilities:

$\xi = J'_*(\alpha'_r)/J_*(\alpha_r)$, where $J'_*(\alpha'_r)$ denotes the class separability yielded by the most dominant kernel for the composite data (i.e., new incoming data and the previous offline data). $J_*(\alpha_r)$ is the class separability yielded by the most dominant kernel for another dataset. Thus, relative separability can be rewritten as: $\xi = \lambda'_*/\lambda_*$, where λ_* corresponds to the most dominant eigenvalue of composite data to be tested, and λ_* is the most dominant eigenvalue of the combined/entire data. Based on the comparison of relative separabilities, the relationships among the combined datasets are finalized in the correct configuration. In this sorting step, we reduce the data size by ignoring the data with nondominant eigenvalues.

Step 4: Merge

Finally, we consider merging the combined databases from Step 3.

Among the kernels $k_i, i = 1, 2, ..p$, we will tune ρ to maximize $k(\rho) = \sum_{i=1}^{p} \rho_i k_i$ for the empirical alignment as follows:

$$\hat{\rho} = \arg\max_{\rho}(Frob(\rho, k_i, k_j)) = \arg\max_{\rho} \left(\frac{\left\langle \sum_i \rho_i K_i, K_j \right\rangle}{\sqrt{\left\langle \sum_i \rho_i K_i \right\rangle, \left\langle \sum_j \rho_j K_j \right\rangle}} \right) = \arg\max_{\rho} \left(\frac{\rho^T V_{ij} \rho}{\rho^T U_{ij} \rho} \right),$$

where

$$u_i = \sqrt{\langle K_i, yy^T \rangle}, v_{ij} = \sqrt{\langle K_i, K_j \rangle},$$
$$U_{ij} = u_i u_j, V_{ij} = v_i v_j, \rho = (\sqrt{\rho_1}, \sqrt{\rho_2}, .., \sqrt{\rho_p}).$$

Let us introduce the generalized Raleigh coefficient, which is given by:

$$J(\rho) = \rho^T V \rho / \rho^T U \rho.$$

We obtain $\hat{\rho}$ by solving the generalized eigenvalue $V\rho = \delta U \rho$ where δ denotes the eigenvalues. Once we find this optimum composite coefficient $\hat{\rho}$, which will be the eigenvector corresponding to maximum eigenvalue δ, we can compute the composite data-dependent kernel matrix. Becuase we have associated all cloud data, which makes use of the data-dependent composite kernel starting from Steps 1-3, we can proceed to recombine the data from all the institutions into small sets of merged databases as shown in Fig. 4.

AMD Algorithm

Step 1: Split each database by maximizing the Fisher scalar.
Step 2: Combine other datasets if the alignment measure of datasets is high.
Step 3: Sort those combined datasets by using class separability as a measure to identify whether the combined data is correctly configured in the right order.
Step 4: Merge the sorted cloud datasets from all the institutions by computing the maximum eigenvalue δ for the composite data-dependent kernel matrix, which represents the associated datasets for KPCA.

The major advantage of the proposed AMD algorithm is to compress multiple datasets into merged databases with the bounded data size. These compressed databases can be handled easier than the original databases, indicating a reduction in computational time, memory, and running cost.

3.3 Implementation of AMD for Two Cloud Cases

The four steps in AMD are implemented into the two platforms, private and public clouds, which are commercially available. We will demonstrate how the proposed Cloud Colonography is specifically adapted to these widly available environments.

Case 1: Private Cloud

The first specification of Cloud Colonography is listed in Fig. 5. This figure shows the representative layered implementation of the private cloud framework and its architectural components. A single institution/hospital handles the patient CTC datasets by a clinical IT staff or a third-party organization, and hosts them either internally or externally.

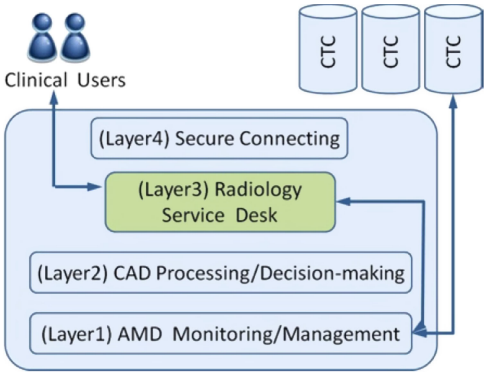

Fig. 5. Cloud Colonography architecture in private cloud environment. The proposed Cloud Colonography consists of the four representative layers from CTC based on CAD analysis to the service desk reporting for clinical users.

Self-run data centers/servers are generally capital intensive, requiring allocations of space, hardware, and environmental controls.

Case 1 has a "layered architecture", with each layer adding further value and complimentary functionality to the data input from the layer below, and providing the relevant output to the layer above. The solution architecture has the following four representative layers:

(*Layer1) AMD Monitoring/Management Layer*: This layer consists of various modules for CTC AMD, which monitor the characteristics of CTC databases as described in Section 3.2. These modules, as described in the AMD algorithm, generate a salient feature space, compare existing datasets to new ones, and prepare shared datasets for the next layer.

(Layer2) CAD Processing/Decision-making Layer: All the CAD decision making from data collection of Layer1 is processed in this layer. This CAD processing includes the detection of cancer, such as patient and cancer identification for the diagnosis.

(Layer3) Radiology Service Desk: This layer further enables to summarize the outcomes to increase radiological efficiency, such as visualization and annotation. These outcomes of the Service Desk provide views, reporting, administration, and operational support for practical and clinical uses.

(Layer4) Secure Connecting Layer: Qualified client-server applications are adapted for clinical users to assess Cloud Colonography. This layer is designed to prevent eavesdropping and tampering. Upon the ID request, this layer of the server switches the connection to Transport Layer Security (TLS) using a protocol-specific mechanism.

The private cloud specification is shown in Table 2 for the comparison of three representative platforms.

The computational performance is shown in Section 5, to analyze the speed, memory, and running cost. The private cloud server [67] has relatively large memory per CPU; however, the other specs of the personal desktop are superior to the private cloud. The personal desktop has the latest specification manufactured in 2014, and the private cloud server is two years older. The key specification is CPU clock speed; 3.6 GHz for the personal desktop, and 2.6 GHz for the private cloud. Other key specifications of these servers for Cloud Colonography are the hard disk size of data storage and the RAM memory size.

TABLE 2
Desktop and Private Cloud Server Hardware Specification

Platform	Desktop Computer	Private Cloud Server
Name	Dell Optiplex 9020	Dogwood
Processor	Intel i7 4790	AMD Opteron 6282 SE
Clock Speed	3.6GHz	2.6GHz
# processors	1	4
# cores	4	16
L1 Cache	4x32KB Data 4x32KB Instruction	8x64KB Data 16x16KB Instruction
L2 Cache	4x256KB	8x2MB
L3 Cache	8MB	2x8MB
Processor Bus Speed	5GT/s DMI	3200 MHz 6.4GT/s HTL
RAM	16GB DDR3	64GB per CPU
Memory Bus Speed	1600MHz	1333 MHz
Storage Technology	7.2k RPM SATA 3Gbps	10k RPM SAS 6Gbps

Case 2: Public cloud

The Cloud computing is also extended into a distributed set of servers that are running at different locations, while still connected to a single network or a hub service as shown Fig. 6. Examples of this include distributed computing platforms such as BOINC [65], which is a voluntarily shared resource that implements cloud computing in the provisions model. A public cloud scenario is used for cloud computing when the services are rendered over a network that is open for multiple institutions/hospitals. Rather than a single database being used, the proposed AMD-based KPCA approach uses more than a single database to improve the cancer classification performance as shown in the experimental results.

Technically, there is little or no difference between the public and private cloud architecture. However, security consideration may be substantially different for services (applications, storage, and other resources) that are made available by a service provider for multiple institutions/hospitals. Generally, public cloud services provide a firewall, like an extended Layer4 of private cloud scenario, to

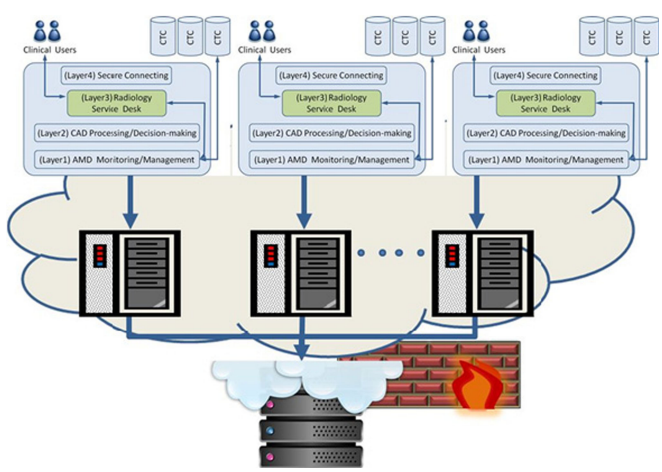

Fig. 6. An extension of AMD framework in public cloud scenario.

TABLE 3
Amazon Representative Servers Specification

Platform	c3.xlarge	c3.8xlarge	r3.xlarge	r3.8xlarge
Processor name	Intel Xeon E5-2670 v2	Intel Xeon E5-2670 v2	Intel Xeon E5-2670 v2	Intel Xeon E5-2670 v2
Clock Speed	2.5 GHz	2.5 GHz	2.5 GHz	2.5 GHz
# vCPU	4	4	4	32
# ECU	14	108	13	104
RAM	7.5 GB	60 GB	30.5 GB	244 GB
Hard Disk Storage	80 GB SSD	640 GB SSD	80 GB SSD	640 GB SSD
Linux Instance Price	$0.21/hr	$1.68/hr	$0.35/hr	$2.80/hr
Processor Bus Speed	8GT/s DMI	8 GT/s DMI	8 GT/s DMI	8 GT/s DMI
Memory Bus Speed	1,866 MHz	1,866 MHz	1,866 MHz	1,866 MHz

enhance a security barrier design, and to prevent unauthorized or unwanted communications between computer networks or hosts. We adopt known cloud infrastructure service companies, such as Amazon AWS [62], Microsoft [63] and IBM [64] to operate the infrastructure at their data center for Cloud Colonography. The hardware specification of Amazon is shown in Table 3.

The computational performance is shown in Section 5, for analyzing the speed, memory, and running cost for the Amazon R3 instance servers. Amazon has relatively large RAM and reasonable pricing as shown in Table 3. We have chosen Amazon because of the MATLAB compatibility.

MODULE 1. PSEUDO ALGORITHM

Data Parallelization
Input: new data, X_n
Parameter: Worker Node Structure, {DX,S}
Output: Update Distributed Dataset, DX
Begin
1. Vectorize multi-dimensional input: $V \leftarrow X_n$
2. Sort vector dataset according to dimension index
3. Slice V as S: $V_s \leftarrow$ Reshape(V,S)
4. Append $V_s\{i\}$ at the end of $DX\{i\}$

End

3.4 Parallelization of AMD

The parallelized programs for a public could server are proposed to optimize KPCA using the MATLAB parallel toolbox. The parallel modules consist of 1) data parallelization of AMD, and 2) process parallelization of KPCA. MATLAB distributed computing servers use MATLAB job schedulers to distribute data and computational load to the cloud nodes. Fig. 7 shows, in [Module1], that large-scaled CTC images are transformed into high-dimensional image-based feature space with the form of distributed array. In [Module2], those arrays are assigned and independently processed by multiple cores of each node. These two modules are fused by the head node in Fig. 7, which optimistically arranges the number of cores to the size of data array. To optimize the overall performance, we need two criteria; 1) minimizing inter-node data transmission for computational speed, and 2) minimizing data storage for memory access requirement. The proposed criteria are designed to optimize the computational

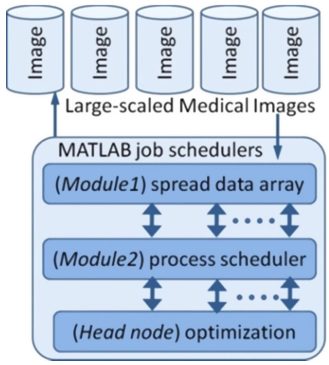

Fig. 7. Proposed parallelized framework for cloud computing.

independency between nodes. The proposed method allows us to maximize the computational resources of elastic computing, which will reduce the computational time and required memory for the processing of pattern recognition algorithms.

The function of the proposed modules 1 and 2 is shown in the following pseudo codes. These steps are designed to reduce the computational time for the processing of pattern recognition algorithms.

MODULE 2. PSEUDOALGORITHM

Training Process Parallelization
Input: Distributed Dataset
Parameter: Kernel parameter, Kernel Identifier
Output: Composite kernel Gram Matrix, Kernel Projector Model, K_c, Classifier Model
Begin
1. Locate and Load Node Data, $DX\{i\}$. Keep data private to the node.
2. Assign Node worker and Corresponding Data by data index.
3. Initiate cloud controller
4. Compute intermediate gram matrix according to kernel parameter
5. Terminate Cloud Controller
6. Return intermediate gram matrix to Head Node
7. Allow Head Node to use kernel parameter to combine intermediate gram matrix from individual workers and construct final gram matrix
8. Associate Gram Matrix with Class label, y
9. Find Eigen Components of the Gram Matrix and use Fisher Analysis to represent Eigen Vectors
10. Identify and Construct Kernel Projector Model, K_c
11. Initiate cloud Controller
12. Compute Kernel Projection using PCA
13. Terminate Cloud Controller
14. Compute Classifier Model

End

The proposed head node is designed to integrate the two parallelization algorithms. The following pseudo code shows testing the data samples using the modules 1 and 2.

4 Classification Evaluation of Cloud Colonography

We evaluated the proposed cloud colonography in Section 4.1 Databases with Classification Criteria, and in Section 4.2 Classification Results.

TABLE 4
Databases

Databases	# Patients		Total # Patients	# Database		Total # Database
	# TP patients	# FP patients		# TP	# FP	
1	5	30	35	12	155	167
2	3	29	32	5	217	222
3	3	10	13	7	213	220
4	3	27	30	5	206	211
5	7	35	42	12	196	208
6	3	25	28	6	198	204
7	3	24	27	6	208	214
8	1	28	29	4	200	204
9	3	17	20	8	190	198
10	3	22	25	7	198	205
11	4	23	27	8	181	189
12	3	29	32	4	191	195
13	2	11	13	8	208	216
14	3	27	30	5	188	193
15	3	15	18	7	147	154
16	3	5	8	8	221	229
17	3	12	15	7	169	176
18	2	25	27	12	169	181
19	2	11	13	5	183	188
Average	3.1	21.3	24.4	7.15	190.1	197.3

TABLE 5
AMD with KPCA for Assembled Database

Kernel type	Database assembling	First kernel	Second kernel
Sigmoid and Gauss	Database9 Database13 Database11 Database16	*Sigmoid* $d=6.67*10^{-4}$ *Offset=0.* $\rho_1=0.49$	*Gauss* $\sigma=0.4$ $\rho_2=0.51$
	Database19 Database6 Database12 Database1	*Sigmoid* $d=8.89*10^{-4}$ *Offset=0.* $\rho_1=1.00$	*Gauss* $\sigma=0.2$ $\rho_2=0.12$
	Database8 Database18 Database15 Database5	*Sigmoid* $d=7.78*10^{-4}$ *Offset=0.* $\rho_1=0.95$	*Gauss* $\sigma=0.3$ $\rho_2=0.05$
Sigmoid and Poly	Database2 Database3 Database14 Database17	*Sigmoid* $d=3.34*10^{-4}$ *Offset=0.* $\rho_1=0.71$	*Linear* *d =3, Offset=0.1* $\rho_2=0.20$
Sigmoid and Linear	Database4 Database7 Database10	*Sigmoid* $d=3.34*10^{-4}$ *Offset=0.* $\rho_1=1.00$	*Linear* *d =13* $\rho_2=0.12$

HEAD Node Testing Algorithm

Testing Process Parallelization
Input: Test Sample(s)
Parameter: Kernel Model, Kernel Projector Model
Output: Test decision Class
Begin
1. Use Data Parallelization Algorithm to distribute test samples to the workers
2. Compute Kernel Projection of the test samples
3. Employ Classifier model to test class of the test sample
End

4.1 Databases with Classification Criteria

We used several cloud settings, with a database consisting of 464 CTC cases that were obtained from 19 institutions/hospitals in the United States and Europe. Our previously developed CAD scheme [55], [56], [57], [73], [74] was applied to these CTC volumes, which yielded a total of 3,774 detections (polyp candidates), consisting of 136 true positive (TP) detections and 3,638 false positive (FP) detections. Note that the supine and prone CTC volumes of a patient were treated as independent in the detection process. A volume of interest (VOI) was placed at each CTC candidate, and the collection of the VOIs for all of the candidates consisted of the databases used for the performance evaluation as shown in Table 4.

We applied up to 40 percent-fold cross-variation for evaluation. Note that the training and testing data were separated from the distinguished form. All the simulations were executed in a MATLAB environment optimized for the Parallel Computing Toolbox [66]. The Windows desktop computing system featured an i7 processor with 3.6 GHz clock speed, and 16 GB of DDR3 memory.

Table 4 lists each database used for the classification results shown in Section 4.2 below. Using AMD, we connected these databases into assembled databases via cloud settings described in Section 3. The CAD classification results mainly come from the choice of databases, which means AMD chooses the databases used for assembling.

Table 5 shows the results of KPCA shown in Section 2 and AMD shown in Section 3. To assemble the databases, we applied KPCA to represent the databases. Using four steps of AMD, for example, Sigmoid and Gauss kernel functions represented the 12 databases (out of 19 databases) into three assembling databases (9-13-11-16, 19-6-121, 8-18-15-5) according to the order of the eigenvalue. The second set of assembling databases was represented by Sigmoid and Poly for databases (2-3-14-17). The last assembling databases were Sigmoid and Linear group (4-7-10). We used Table 5 for the classification results in Section 4.2.

We demonstrated the advantage of Cloud Colonography in terms of the CAD classification performance in a more quantitative manner by introducing numerical criteria. We evaluated the classification accuracy of CAD using receiver operating characteristic (ROC) of the sensitivity and specificity criteria as statistical measures of the performance. The true positive rate (TPR) defines a diagnostic test performance for classifying positive cases correctly among all positive instances available during the test. The false positive rate (FPR) determines how many incorrect positive results occur among all negative samples available during the test as follows:

$$\mathrm{TPR} = \frac{\mathrm{True\,Positives\,(TP)}}{\mathrm{True\,Positives\,(TP)} + \mathrm{False\,Negatives\,(FN)}},$$

$$\mathrm{FPR} = \frac{\mathrm{False\,Positives\,(FP)}}{\mathrm{False\,Positives\,(FP)} + \mathrm{True\,Negatives\,(TN)}}.$$

We also evaluated alternative numerical values, Area Under the Curves (AUC) by

TABLE 6
Classification Accuracy for Assembled Database

Kernel type	Database assembling	Dataset size (MB)	AUC	**Assembled AUC**
Sigmoid and Gauss	Database9	*916*	*0.94*	***0.99***
	Database13	*988*	*0.95*	
	Database11	*868*	*0.97*	
	Database16	*1064*	*0.97*	
	Database19	*780*	*0.38*	***0.99***
	Database6	*940*	*0.82*	
	Database12	*900*	*0.93*	
	Database1	*780*	*0.97*	
	Database8	*932*	*0.50*	***0.98***
	Database18	*756*	*0.99*	
	Database15	*696*	*0.79*	
	Database5	*960*	*0.85*	
Sigmoid and Poly	Database2	*1028*	*0.32*	***0.85***
	Database3	*1036*	*0.12*	
	Database14	*876*	*0.87*	
	Database17	*820*	*0.99*	
Sigmoid and Linear	Database4	*976*	*0.40*	***0.97***
	Database7	*251*	*0.17*	
	Database10	*956*	*0.57*	

$$AUC = \sum_{i=1}^{z} (FPR_i - FPR_{i-1}) TPR_i,$$

where Z is the number of discrete FPR_i.

The proposed statistical analysis by use of AMD was applied to the multiple databases in Table 1, which showed that the CTC data were highly biased toward FPs (the average ratio between TP and FP is 1: 26.6) due to the limited number of true polyps found in the patients comprising the CTC database.

4.2 Classification Results

We used the K-NN method for Table 4 for classification with the parameter k, and yielded the performance of TPR and specificity with respect to the variable k in ROC analysis. We compared assembling databases performance, with a single database in Table 6. AMD for assembling database outperformed the KPCA for a single database by achieving a higher TPR and a lower FPR.

The ROC analysis results show the calculated AUC (vertical axis) for the proposed method as the total network data size (horizontal axis) increases (mostly FT data). These figures show that Cloud Colonography with AMD exhibited AUC performance for classification in handling the AMD in all five different cloud environments shown in Section 5. The ability to track changes using growing database size was also verified by the results shown in the following Section.

5 Cloud Computing Performance

We evaluated private and public cloud scenarios in four aspects of the proposed design in Cloud Colonography. These are examined in Section 5.1 cloud computing setting with Cancer Databases, Section 5.2 Computational time, Section 5.3 Memory usage, Section 5.4 Running cost, and Section 5.5 Parallelization.

TABLE 7
Databases for Cloud Computing Analysis

Databases	Training Datasets				Testing Datasets			
	#TP	#FP	#Total	Size (MB)	#TP	#FP	#Total	Size (MB)
1	122	1,289	1,411	9,524	14	157	171	1,154
2	120	853	973	6,568	16	110	126	851
3	121	1,071	1,192	8,046	15	133	148	999
Total	363	3,213	3,576	2,4138	45	400	445	3,004

5.1 Cloud Computing Setting with Cancer Databases

Table 7 shows how databases are formed into AMD. Three databases were generated from Table 4 for the analysis of both private and public cloud environments. These three datasets are a synthesis of the existing 19 databasets by merging them into the cloud server; thus, the classification performance was equivalent to Section 4. These three generated databases are mainly used for the evaluation of cloud computing.

Fig. 8 shows the compressed data ratio (vertical axis) of the Cloud Colonography network as the total network data size (horizontal axis) increases. The three databases for cloud environments are used to analyze how the proposed method handles the enlarged data sizes. The data compression ratio was defined as the size of the AMD feature space versus the size of the raw data. As the data size increased, the compressed data ratio was reduced. Compared to the raw data, the three generated databases worked well to maintain the size for the analysis over the entire data size using AMD. The compression ratio reflects the corresponding decrease in heterogeneous to homogenous data. In other words, a constant ratio with increasing data size indicates an equal growth in both heterogeneous and homogenous data, such as in the case of raw data. The evaluation criteria for the experimental datasets start from computational speed with varying database sizes. These results have been fully-outlined in the remaining sections below.

5.2 Computation Time

In this section, we analyzed the time required for processing of Cloud Colonography with AMD. All the experiments were implemented in MATLAB R2010a using the Statistical

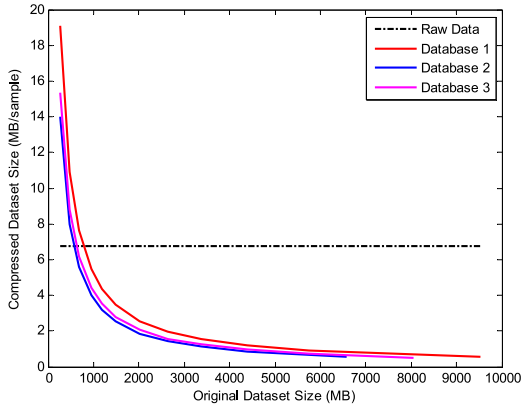

Fig. 8 Data compression ratio for data size. The horizontal axis denotes the size of the data, and vertical axis denotes the compressed data size.

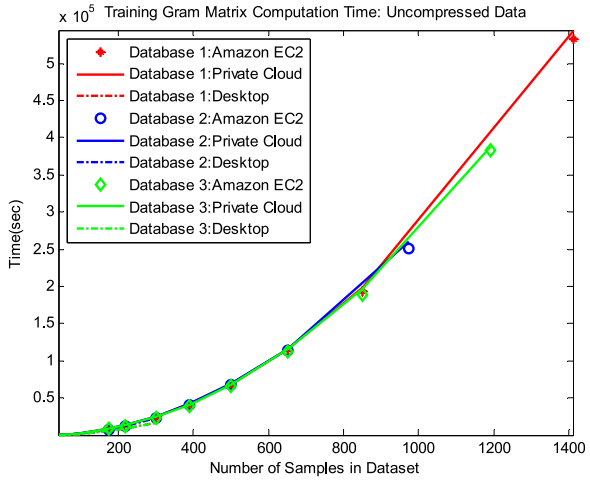

Fig. 9. Total training time required for Cloud Colonography with AMD.

Pattern Recognition Toolbox for the Gram matrix calculation and kernel projection. For processing the large volume of data, private cloud environments were used. Run time was determined using the cputime command, representing how significant computational savings were for each environment.

Fig. 9 shows the total computational time required for growing network database sizes in the Private Cloud and Desktop environment. The total training time was measured for three uncompressed databases listed in Table 7. The total training time was increased as the number of samples increased. The difference between the Private Cloud and Desktop was relatively small.

Fig. 10 shows the mean execution time for compressed datasets. Compared to Fig. 9, the computation time was much improved when the datasets were compressed. The Private Cloud required more computation time than the Desktop for all three data cases. The time difference increased as the number of datasets increased. The difference was calculated by averaging three databases shown in Table 8.

Table 8 shows the average of the total training time and mean execution time shown in Figs. 9 and 10. These values were calculated by averaging three databases for each cloud environment.

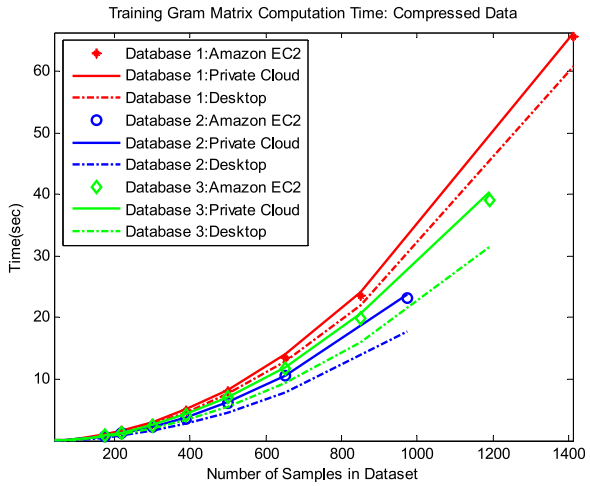

Fig. 10. Mean execution time for Cloud Colonography with AMD.

TABLE 8
Averaged Training Computation Time for Three Databases

Cloud	Uncompressed data (sec)	Compressed data (sec)
Desktop	1.4×10^5	12.3
Private Cloud	1.4×10^5	14.6
Public Cloud	1.4×10^5	13.9

Table 8 demonstrates that the computation time for the private cloud was 18 percent larger than the desktop, indicatitng that the desktop was 18 percent , on average, faster than the private cloud. Based on the hardware specification in Table 2, the CPU speed of the desktop was 38 percent faster than the private cloud. The difference of the computational time between uncompressed and compressed datasets was over 10^4. The big reduction of computational time was achieved by the AMD due to the data compression. The increased ratio of the computation time in Figs. 9 and 10 shows that the proposed method was computationally efficient as the overall network database size increased. Therefore, Cloud Colonography was better-suited to handle long-term sequences in a practical manner. Our computational speed for larger CTC databases is promising even if much larger datasets are used for the screenings.

Fig. 11 selected the main module for the training time to specifically calculate Gram Matrix in Cloud Colonography with AMD. The calculation of Gram Matrix is the key module to computer KPCA with comparison between the Desktop and private cloud. Fig. 11 also shows that Desktop was faster than private cloud for all three databases for the module of Gram matrix. The Private Cloud required more computation time than the Desktop for testing phase as shown in Table 9.

5.3 Memory Usage

In this section, we examine the degree of memory usage in Cloud Colonography with AMD as the data size changes. Fig. 12 illustrates the effect of continuously increasing data size on the observed level of memory usage in each of the three databases.

Fig. 12 shows that the proposed framework required more memory usage as the training sample increased in

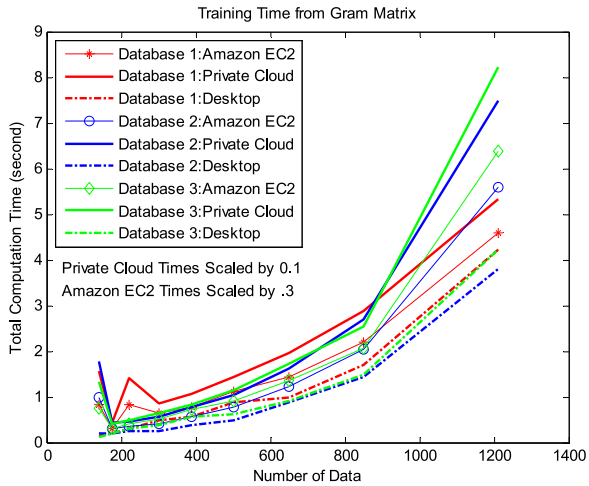

Fig. 11. Computational time for Gram matrix in Cloud Colonography with AMD.

TABLE 9
Average Total Testing Time per Sample

Database	Desktop Environment (ms)	Private Cloud (ms)	Public Cloud (ms)
1	361	2,851.4	983.1
2	263	2,037.1	695.2
3	289.7	2,232.9	775.3

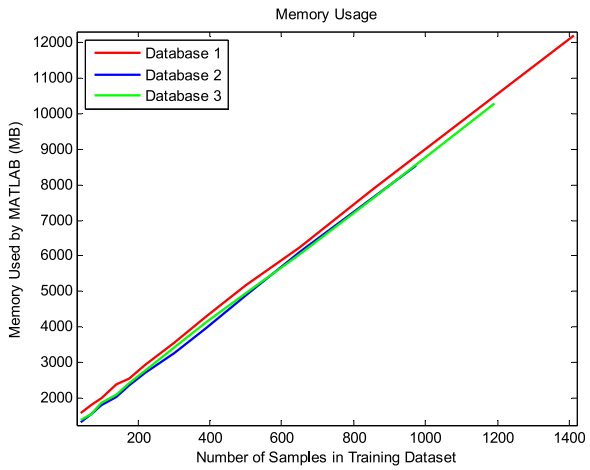

Fig. 12. Memory usage for Cloud Colonography with AMD.

TABLE 10
Cost Components for Private Cloud Service

Cost Component	Database 1	Database 2	Database 3
Storage (GB)	12.3	8.9	10.7
Training Memory (MB)	12.2×10^3	10.28×10^3	8.523×10^3
Testing Memory (MB)	2.12×10^3	1.61×10^3	1.81×10^3
Training Processing (trillion FLOPs)	907	436	648
Testing Processing (trillion FLOPs)	12.8	6.9	8.9

size. There are no differences of memory usage among cloud environments examined in this study. The degree of memory usage was proportionally increased as the training samples increased. We need to develop more efficient method of memory usage, as the database size increases.

5.4 Running Cost

Cloud service providers typically charge based on usage of memory, computation amount, storage, data transfer speed, and subscription type of the service. The private cloud service, as well as Desktop, was offered free of charge for this research purpose, but this was not the case for the public cloud. Thus we estimated the cost based on resource consumption for the three databases used.

Tables 10 and 11 illustrate the total cost component of the total storage data, memory, and processing in the cloud for three different databases used in this study. The result in previous sections, such as Fig. 12, showed that the total cost for cloud data storage increased in proportion to the total data stored on the cloud. As the size of the data stored was increased, the training time was also increased, which increased the total cost.

TABLE 11
Cost Components for Public Cloud Amazon EC2 (c3.8xlarge)

Cost Component	Database 1	Database 2	Database 3
Storage (GB)	12.3	8.9	10.7
Maximum Training Memory (MB)	14.5×10^3	11.0×10^3	12.9×10^3
Maximum Testing Memory (MB)	2.16×10^3	1.64×10^3	1.85×10^3
Approximated Marginal Cost	\$9.25	\$4.40	\$6.60

TABLE 12
Time and Memory Using Parallelization for Private Cloud

Module Environment	Worker	Computation Time Mean (ms/sample)	Maximum Memory
Parallelization	16	0.519	~59 GB
	32	0.271	~63 GB
Without parallelization	1	78.24	~35 GB

TABLE 13
Time and Cost Using Parallelization for Public Cloud Amazon EC2 (c3.8xlarge)

Module Environment	Node	Worker	Computation Time Mean (ms/sample)	Approximate Cost
With parallelization	1	16	0.504	\$2.75
	2	32	0.268	\$2.05
	4	64	0.145	\$1.91
Without parallelization	1	1	76.82	\$6.60

5.5 Parallelization

The proposed parallelization method described in Section 3.4 was tested on both private and public cloud environment. Computation time for large uncompressed unified Database (~25 GB) experienced ~100 fold improvement. The summary of computation time for training from uncompressed data was summarized in Table 12 for private cloud, and in Table 13 for public cloud.

As shown in Tables 12 and 13, the computational time was dramatically improved when the proposed parallelization module was implemented. In the case of public cloud, the cost was reduced by more than half. These results demonstrate that our proposed parallelization is effective for Cloud Colonography.

5.6 Comparison to Other Studies

The results were compared with existing most related work [69], [70], [71], [72]. The criteria were the same as those of Tables 12 and 13 in Section 5.5. The node was defined as physically separated hardwares, while the worker was virtually separated program running in the same physical environment. The computation time reduction was calculated as a measure of improvement because of the parallelization. The higher time reduction indicated better performance.

TABLE 14
Comparisons of other Methods

Method	Node (Worker)	Computation Time Reduction (%)
AMD (Public Cloud)	2(64)	99.81
AMD (Private Server)	1(32)	99.65
AMD (Private Server)	1(16)	99.34
MapReduce SVM [69]	1(4)	76.14
Distributed SVM Ensemble [70]	1(4)	91.67
Method 3 [71]	1(2)	81
Parallel Dynamic ANN [72]	1(2)	48.72

As shown in Table 14, the computation time was much reduced for the proposed AMD, compared to other studies [69], [70], [71], [72]. These comparision results validated the efficient computational performance of AMD.

6 Conclusion

We proposed a new framework of Cloud Colonography, using different types of cloud computing environments. The databases from CTC screening tests at several hospitals are going to be networked in the near future via cloud computing technologies. The proposed method of AMD was developed in this study for efficient handling of multiple CTC databases. When AMD is used for assembling databases, it can achieve almost 100 percent classification accuracy. The proposed AMD has the potential to be a core classifier tool in the cloud computing framework for a model-based CAD scheme, which will yield high detection performance of polyps using KPCA for databases at multiple institutions. Two cases in the proposed cloud platform are private and public. The public cloud performed better than the private cloud in computation time, but the memory usage was equivalent. The parallelization was successfully developed to improve the speed and cost. CTC based on CAD in the cloud computing environment is expected to advance the clinical implementation of cancer screening and promote the early diagnosis of colon cancer.

Acknowledgments

This study was supported partly by CTSA UL1TR000058 from the National Center for Advancing Translational Sciences, Center for Clinical and Translational Research Endowment Fund of Virginia Commonwealth University (VCU), Dean's Undergraduate Research Initiative (DURI) at VCU, Institutional Research Grant IRG-73-001-31 from the American Cancer Society through the Massey Cancer Center, Presidential Research Incentive Program at VCU, National Science Foundation (NSF) CAREER Award 1054333. Part of this study was supported by the National Institutes of Health grant of CA095279 (PI: H.Y.) and CA166816 (PI: H.Y.)

References

[1] S. Winawer, R. Fletcher, D. Rex, J. Bond, R. Burt, J. Ferrucci, T. Ganiats, T. Levin, S. Woolf, D. Johnson, L. Kirk, S. Litin, and C. Simmang, "Colorectal cancer screening and surveillance: Clinical guidelines and rationale-Update based on new evidence," *Gastroenterology*, vol. 124, pp. 544–60, Feb 2003.

[2] J. G. Fletcher, F. Booya, C. D. Johnson, and D. Ahlquist, "CT colonography: Unraveling the twists and turns," *Curr. Opin. Gastroenterol.*, vol. 21, pp. 90–98, Jan. 2005.

[3] Z. B. Xu, M. W. Dai, and D. Y. Meng, "Fast and efficient strategies for model selection of Gaussian support vector machine," *IEEE Trans. Syst., Man, Cybern. B*, vol. 39, no. 5, pp. 1292–1207, Oct. 2009.

[4] H. Yoshida and J. J. Näppi, "CAD in CT colonography without and with fecal tagging: Progress and challenges," *J. Comput. Med. Imag. Graph.*, vol. 30, pp. 267–284, 2007.

[5] G. Kiss, J. Van Cleynenbreugel, M. Thomeer, P. Suetens, and G. Marchal, "Computer-aided diagnosis in virtual colonography via combination of surface normal and sphere fitting methods," *Eur. Radiol.*, vol. 12, pp. 77–81, Jan. 2002.

[6] D. S. Paik, C. F. Beaulieu, G. D. Rubin, B. Acar, R. B. Jeffrey, Jr., J. Yee, J. Dey, and S. Napel, "Surface normal overlap: A computer-aided detection algorithm with application to colonic polyps and lung nodules in helical CT," *IEEE Trans. Med. Imag.*, vol. 23, no. 6, pp. 661–675, Jun. 2004.

[7] A. K. Jerebko, R. M. Summers, J. D. Malley, M. Franaszek, and C. D. Johnson, "Computer-assisted detection of colonic polyps with CT colonography using neural networks and binary classification trees," *Med. Phys.*, vol. 30, pp. 52–60, Jan. 2003.

[8] J. Näppi and H. Yoshida, "Adaptive correction of the pseudo-enhancement of CT attenuation for fecal-tagging CT colonography," *Med. Image Anal.*, vol. 12, pp. 413–426, 2009.

[9] A. K. Jerebko, J. D. Malley, M. Franaszek, and R. M. Summers, "Multiple neural network classification scheme for detection of colonic polyps in CT colonography Databases," *Acad. Radiol.*, vol. 10, pp. 154–160, Feb. 2003.

[10] A. K. Jerebko, J. D. Malley, M. Franaszek, and R. M. Summers, "Support vector machines committee classification method for computer-aided polyp detection in CT colonography," *Acad. Radiol.*, vol. 12, pp. 479–486, 2005.

[11] M. Doumpos, C. Zopounidis, and V. Golfinopoulou, "Additive support vector machines for pattern classification," *IEEE Trans. Syst., Man, Cybern. B.*, vol. 37, no. 3, pp. 540–550, Jun. 2007.

[12] R. M. Summers, C. F. Beaulieu, L. M. Pusanik, J. D. Malley, R. B. Jeffrey, Jr., D. I. Glazer, and S. Napel, "Automated polyp detector for CT colonography: feasibility study," *Radiology*, vol. 216, pp. 284–290, 2000.

[13] R. M. Summers, M. Franaszek, M. T. Miller, P. J. Pickhardt, J. R. Choi, and W. R. Schindler, "Computer-aided detection of polyps on oral contrast-enhanced CT colonography," *AJR Am. J. Roentgenol.*, vol. 184, pp. 105–108, 2005.

[14] C. F. Juang, S. H. Chiu, and S. J. Shiu, "Fuzzy system learned through fuzzy clustering and support vector machine for human skin color segmentation," *IEEE Trans. Syst., Man, Cybern. A*, vol. 37, no. 6, pp. 1077–1087, Jan. 2007.

[15] X. Zhang and X. Ren, "Two dimensional principal component analysis based independent component analysis for face recognition," *Int. Conf. Multi. Tech. (ICMT)*, pp. 934–936, 2011.

[16] M. Xu, P. M. Thompson, and A. W. Toga, "Adaptive reproducing kernel particle method for extraction of the cortical surface," *IEEE Trans. Med. Imag.*, vol. 25, no. 6, pp. 755–762, Jun. 2006.

[17] Y. Shao and C. -H. Chang, "Bayesian separation with sparsity promotion in perceptual wavelet domain for speech enhancement and hybrid speech recognition," *IEEE Trans. Syst., Man, Cybern. A*, vol. 41, no. 2, pp. 284–293, 2011.

[18] W. Zheng, C. Zou, and L. Zhao, "An improved algorithm for kernel principal component analysis," *Neural Process. Lett.*, vol. 22, pp. 49–56, 2005.

[19] J. Kivinen, A. J. Smola, and R. C. Williamson, "Online learning with kernels," *IEEE Trans. Signal Process.*, vol. 52, no. 8, pp. 2165–2176, Aug. 2004.

[20] S. Ozawa, S. Pang, and N. Kasabov, "Incremental learning of chunk data for online pattern classification systems," *IEEE Trans. Neural Networks*, vol. 19, no. 6, pp. 1061–1074, Jun. 2008.

[21] H. T. Zhao, P. C. Yuen, and J. T. Kwok, "A novel incremental principal component analysis and its application for face recognition," *IEEE Trans. Syst., Man, Cybern. B*, vol. 36, no. 4, pp. 873–886, Aug. 2006.

[22] Y. M. Li, "On incremental and robust subspace learning," *Pattern Recog.*, vol. 37, no. 7, pp. 1509–1518, 2004.

[23] Y. Kim, "Incremental principal component analysis for image processing," *Opt. Lett.*, vol. 32, no. 1, pp. 32–34, 2007.

[24] B. J. Kim and I. K. Kim, "Incremental nonlinear PCA for classification," in *Proc. 8th Eur. Conf. Principles Practice Knowl. Discovery Databases*, 2004, vol. 3202, pp. 291–300.

[25] B. J. Kim, J. Y. Shim, C. H. Hwang, I. K. Kim, and J. H. Song. "Incremental feature extraction based on empirical kernel map," *Found. Intell. Syst.*, vol. 2871, pp. 440–444, 2003.
[26] L. Hoegaerts, L. De Lathauwer, I. Goethals, J. A. K. Suykens, J. Vandewalle, and B. De Moor "Efficiently updating and tracking the dominant kernel principal components," *Neural Netw.*, vol. 20, no. 2, pp. 220–229, 2007.
[27] T. J. Chin and D. Suter. "Incremental kernel principal component analysis," *IEEE Trans. Image Process.*, vol. 16, no. 6, pp. 1662–1674, Jun. 2007.
[28] X. Jiang, R. Snapp, Y. Motai, and X. Zhu, "Accelerated kernel feature analysis," in *Proc. IEEE Comput. Soc. Conf. Comput. Vis. Pattern Recog.*, 2006, pp. 109–116.
[29] V. N. Vapnik, *The Nature of Statistical Learning Theory*, 2nd ed. New York, NY, USA: Springer, 2000.
[30] T. Damoulas and M. A. Girolami, "Probabilistic multi-class multi—kernel learning: On protein fold recognition and remote homology detection," *Bioinformatics*, vol. 24, no. 10, pp. 1264–1270, 2008.
[31] S. Amari and S. Wu, "Improving support vector machine classifiers by modifying kernel functions," *Neural Netw.*, vol. 6, pp. 783–789, 1999.
[32] B. Souza and A. De Carvalho, "Gene selection based on multiclass support vector machines and genetic algorithms," *Molecular Res.*, vol. 4, no. 3, pp. 599–607, 2005.
[33] L. Jayawardhana, Y. Motai, and A. Docef, "Computer-aided detection of polyps in CT colonography: On-line versus off-line accelerated kernel feature analysis," *Signal Process.*, Special issue on processing and analysis of high-dimensional masses of image and signal data, vol. 90, no. 8, pp. 2456–2467, 2010.
[34] M. Awad, Y. Motai, J. Näppi, and H. Yoshida, "A clinical decision support framework for incremental polyps classification in virtual colonoscopy," *Algorithms*, vol. 3, pp. 1–20, 2010.
[35] H. Xiong, M. N. S. Swamy, and M. O. Ahmad, "Optimizing the kernel in the empirical feature space," *IEEE Trans. Neural Netw.*, vol. 16, no. 2, pp. 460–474, Mar. 2005.
[36] H. Xiong, Y. Zhang, and X. W. Chen, "Data-dependent kernel machines for microarray data classification," *IEEE/ACM Trans. Comput. Biol. Bioinformat.*, vol. 4, no. 4, pp. 583–595, Oct.–Dec. 2007.
[37] B. Schökopf and A. J. Smola, "Learning with kernels: support vector machines, regularization, optimization, and beyond," in *Adaptive Computation and Machine Learning*. Cambridge, MA, USA: MIT Press, 2002.
[38] H. Fröhlich, O. Chapelle, and B. Scholkopf, "Feature selection for support vector machines by means of genetic algorithm," in *Proc. 15th IEEE Int. Conf. Tools Artif. Intell.*, 2003, pp. 142–148.
[39] X. W. Chen, "Gene selection for cancer classification using bootstrapped genetic algorithms and support vector machines," in *Proc. IEEE Int. Conf. Comput. Syst., Bioinformat. Conf.*, 2003, pp. 504–505.
[40] C. Park and S. B. Cho, "Genetic search for optimal ensemble of feature-classifier pairs in DNA gene expression profiles," in *Proc. Int. Joint Conf. Neural Netw.*, 2003, vol. 3, pp. 1702–1707.
[41] F. A. Sadjadi, "Polarimetric radar target classification using support vector machines," *Opt. Eng.*, vol. 47, no. 4, pp. 046201:1–046201:8, 2008.
[42] C. Hu, Y. Chang, R. Feris, and M. Turk, "Manifold based analysis of facial expression," in *Proc. Comput. Vis. Pattern Recog. Workshop*, 2004, vol. 27, pp. 81–85.
[43] T. T. Chang, J. Feng, H. W. Liu, and H. Ip, "Clustered microcalcification detection based on a multiple kernel support vector machine with grouped features," in *Proc 19th Int. Conf. Pattern Recog.*, 2008, vol. 8, pp. 1–4.
[44] T. Briggs and T. Oates, "Discovering domain specific composite kernels," in *Proc 20th Nat. Conf. Artif. Intell.*, 2005, pp. 732–738.
[45] N. Cristianini, J. Kandola, A. Elisseeff, and J. Shawe-Taylor, "On kernel target alignment," in *Proc. Neural Inf. Process. Syst.*, 2001, pp. 367–373.
[46] G. R. G. Lanckriet, N. Cristianini, P. Bartlett, L. Ghaoui, and M. I. Jordan, "Learning the kernel matrix with semidefinite programming," *J. Mach. Learn. Res.*, vol. 5, pp. 27–72, 2004.
[47] Y. Tan and J. Wang, "A support vector machine with a hybrid kernel and minimal Vapnik-Chervonenkis dimension," *IEEE Trans. Knowl. Data Eng.*, vol. 16, no. 4, pp. 385–395, Apr. 2004.
[48] A. K. Jerebko, R. M. Summers, J. D. Malley, M. Franaszek, and C. D. Johnson, "Computer-assisted detection of colonic polyps with CT colonography using neural networks and binary classification trees," *Med. Phys.*, vol. 30, pp. 52–60, 2003.
[49] J. Näppi, H. Frimmel, A. H. Dachman, and H. Yoshida, "A new high-performance CAD scheme for the detection of polyps in CT colonography," in *Proc. Med Imag. 2004: Image Process.*, 2004, pp. 839–848.
[50] A. K. Jerebko, J. D. Malley, M. Franaszek, and R. M. Summers, "Multiple neural network classification scheme for detection of colonic polyps in CT colonography databases," *Acad. Radiol.*, vol. 10, pp. 154–160, 2003.
[51] A. K. Jerebko, J. D. Malley, M. Franaszek, and R. M. Summers, "Support vector machines committee classification method for computer-aided polyp detection in CT colonography," *Acad. Radiol.*, vol. 12, pp. 479–86, 2005.
[52] D. P. Zhu, R. W. Conners, D. L. Schmoldt, and P. A. Araman, "A prototype vision system for analyzing CT imagery of hardwood logs," *IEEE Trans. Syst., Man, Cybern. B*, vol. 26, no. 4, pp. 522–532, Aug. 1996.
[53] Y. Shao and C. H. Chang, "Bayesian separation with sparsity promotion in perceptual wavelet domain for speech enhancement and hybrid speech recognition,"*IEEE Trans. Syst., Man, Cybern. A*, vol. 41, no. 2, pp. 284–293, Mar. 2011.
[54] J. H. Yao, M. Miller, M. Franaszek, and R.M. Summers, "Colonic polyp segmentation in CT colonography-based on fuzzy clustering and deformable models," *IEEE Trans. Med. Imag.*, vol. 23, no. 11, pp. 1344–1356, Nov. 2004.
[55] H. Yoshida and J. Nappi, "Three-dimensional computer-aided diagnosis scheme for detection of colonic polyps," *IEEE Trans. Med. Imag.*, vol. 20, no. 12, pp. 1261–1274, Dec. 2001.
[56] J. Näppi and H. Yoshida, "Fully automated three-dimensional detection of polyps in fecal-tagging CT colonography," *Acad. Radiol.*, vol. 14, pp. 287–300, Mar. 2007.
[57] H. Yoshida, J. Näppi, and W. Cai, "Computer-aided detection of polyps in CT colonography: Performance evaluation in comparison with human readers based on large multicenter clinical trial cases," in *Proc 6th IEEE Int. Symp. Biomed. Imag.*, 2009, pp. 919–922.
[58] Y. Motai and H. Yoshida, "Principal composite kernel feature Analysis: Data-Dependent kernel approach," *IEEE Trans. Knowl. Data Eng.*, vol. 25, no. 8, pp. 1863–1875, Aug. 2013.
[59] S. Di, C. Wang, and F. Cappello, "Adaptive algorithm for minimizing cloud task length with prediction errors," *IEEE Trans. Cloud Comput.*, vol. 2, no. 2, pp. 194–207, Apr.–Jun. 2014.
[60] C. Tsai, W. Huang, M. Chiang, M. Chiang, and C. Yang, "A hyper-heuristic scheduling algorithm for cloud," *IEEE Trans. Cloud Comput.*, vol. 2, no. 2, pp. 236–250, Apr.–Jun. 2014.
[61] F. Larumbe and B. Sanso, "A tabu search algorithm for the location of data centers and software components in green cloud computing networks," *IEEE Trans. Cloud Comput.*, vol. 1, no. 1, pp. 22–35, Jan.–Jun. 2013.
[62] (2015). Amazon Elastic Compute Cloud [Online]. Available: http://aws.amazon.com/ec2/
[63] (2015). Microsoft Azure [Online]. Available: https://azure.microsoft.com
[64] (2015). IBM cloud services [Online]. Available: http://www.ibm.com/cloud-computing/us/en/
[65] (2015). Berkeley Open Infrastructure for Network Computing [Online]. Available: http://boinc.berkeley.edu/
[66] (2015). MathWorks [Online]. Available: http://www.mathworks.com/products/parallel-computing/
[67] (2015). Department cluster Dogwood [Online]. Available: http://computerscience.egr.vcu.edu/undergraduate/resourcescurrent/computing-facilities/
[68] M. Kesavan, I. Ahmad, O. Krieger, R. Soundararajan, A. Gavrilovska, and K. Schwan, "Practical compute capacity management for virtualized datacenters," *IEEE Trans. Cloud Comput.*, vol. 1, no. 1, p. 1, Jan.–Jun. 2013.
[69] G. Caruana, L. Maozhen, and M. Qi, "A MapReduce based parallel SVM for large scale spam filtering," in *Proc. 8th Int. Conf. Fuzzy Syst. Knowl. Discovery*, 2011, vol. 4, pp. 2659–2662.
[70] N. K. Alham, L. Maozhen, L. Yang, M. Ponraj, and M. Qi, "A distributed SVM ensemble for image classification and annotation," in *Proc. 9th Int. Conf. Fuzzy Syst. Knowl. Discovery*, 2012, pp. 1581–1584.
[71] C. Silva, U. Lotric, B. Ribeiro, and A. Dobnikar, "Distributed text classification with an ensemble Kernel-based learning approach," *IEEE Trans. Syst., Man, Cybern. C, Appl. Rev.*, vol. 40, no. 3, pp. 287–297, May 2010.
[72] L. Prechelt, "Exploiting domain-specific properties: Compiling parallel dynamic neural network algorithms into efficient code," *IEEE Trans. Parallel Distrib. Syst.*, vol. 10, no.11, pp. 1105–1117, Nov. 1999.

[73] H. Yoshida, Y. Masutani, P. MacEneaney, D. Rubin, A. H. Dachman, "Computerized detection of colonic polyps at CT colonography on the basis of volumetric features: pilot study," *Radiology*, vol. 222, pp. 327–336, 2002.
[74] H. Yoshida, J. J. Näppi, P. MacEneaney, and D. Rubin, A. H. Dachman, "Computer-aided diagnosis scheme for the detection of polyps in CT colonography," *RadioGraphics*, vol. 22, pp. 963–979 2002.
[75] H. Yoshida and A. H. Dachman, "CAD techniques, challenges and controversies in computed tomographic colonography," *Abdom. Imag.*, vol. 30, pp. 26–41, 2005.

Yuichi Motai (M'01-SM'13) received the BEng degree in instrumentation engineering from Keio University, Tokyo, Japan, in 1991, the MEng degree in applied systems science from Kyoto University, Kyoto, Japan, in 1993, and the PhD degree in electrical and computer engineering from Purdue University, West Lafayette, IN, in 2002. He is currently an associate professor of electrical and computer engineering at VCU, Richmond, VA. His research interests include the broad area of sensory intelligence; particularly in medical imaging, pattern recognition, computer vision, and robotics. He is a senior member of the IEEE.

Eric Henderson received the BS degree in computer engineering from VCU located in Richmond, Virginia, in May 2015. He joined the Sensory Intelligence Lab as an undergraduate research assistant for the US National Science Foundation (NSF) Research Experiences for Undergraduate (REU) program. He was also an undergraduate research fellow of DURI.

Nahian Alam Siddique (S'12) received the BS degree in electrical and electronic engineering from Bangladesh University of Engineering and Technology, Dhaka, Bangladesh, in 2011. He is currently working toward the graduate studies at the Department of Electrical and Computer Engineering, VCU, Richmond, VA. He was a graduate research mentor of DURI. His research interests include specific areas of sensory intelligence—particularly in medical imaging, pattern recognition, and computer vision. He is a student member of the IEEE.

Hiroyuki Yoshida (M'96) received the BS and MS degrees in physics and the PhD degree in information science from the University of Tokyo, Japan. He previously held an assistant professorship in the Department of Radiology, University of Chicago. He was a tenured associate professor when he left the university and joined the Massachusetts General Hospital (MGH) and Harvard Medical School (HMS) in 2005, where he is currently the director of 3D Imaging Research in the Department of Radiology, MGH and an associate professor of radiology at HMS. His research interest is in computer-aided diagnosis, in particular the detection of polyps in CTC, for which he received several awards at the Annual Meetings of Radiological Society of North America and the International Society for Optical Engineering. He is a member of the IEEE.