POLISH MARITIME RESEARCH Special Issue 2017 S2 (94) 2017 Vol. 24; pp. 39-44
10.1515/pomr-2017-0062

FUZZY METHOD AND NEURAL NETWORK MODEL PARALLEL IMPLEMENTATION OF MULTI-LAYER NEURAL NETWORK BASED ON CLOUD COMPUTING FOR REAL TIME DATA TRANSMISSION IN LARGE OFFSHORE PLATFORM

Zhang Hu, Ph. D.
Wei Qin, Ph. D.
School of Information Engineering, Wuhan University of Technology,Wuhan, 430074, China

ABSTRACT

With the rapid development of electronic technology, network technology and cloud computing technology, the current data is increasing in the way of mass, has entered the era of big data. Based on cloud computing clusters, this paper proposes a novel method of parallel implementation of multilayered neural networks based on Map-Reduce. Namely in order to meet the requirements of big data processing, this paper presents an efficient mapping scheme for a fully connected multi-layered neural network, which is trained by using error back propagation (BP) algorithm based on Map-Reduce on cloud computing clusters (MRBP). The batch-training (or epoch-training) regimes are used by effective segmentation of samples on the clusters, and are adopted in the separated training method, weight summary to achieve convergence by iterating. For a parallel BP algorithm on the clusters and a serial BP algorithm on uniprocessor, the required time for implementing the algorithms is derived. The performance parameters, such as speed-up, optimal number and minimum of data nodes are evaluated for the parallel BP algorithm on the clusters. Experiment results demonstrate that the proposed parallel BP algorithm in this paper has better speed-up, faster convergence rate, less iterations than that of the existed algorithms.

Keywords: Parallel implementation; Multi-layer neural network; Cloud computing

INTRODUCTION

With the rapid development of electronic technology, network technology and cloud computing technology, the current data is increasing in the way of mass, has entered the era of big data. Real world data, such as digital images, the gene expression patterns, face data set or web page text, usually have the characteristics of high dimension and large data volume. For traditional technologies of artificial intelligence and pattern recognition and so on, are all faced with the challenge of how to implement the data processing in the era of big data. For example, in the classification of a large scale of face data sets, a computer or workstation is very difficult to adapt to the actual requirements because of the lack of speed and storage capacity. Therefore, it is very necessary to study how to implement the technologies of artificial intelligence and pattern recognition based on multi-computer clusters in large data environment. When using the neural network in artificial intelligence to deal with the related data, if the number of size of training samples is not large, generalization ability and running time of single neural network are relatively ideal. However, with the increase of the identification number of categories, the structure of the neural network will also become more complex, lead to neural network training time become longer, convergence speed become slower, being easy to fall into local minimum and have the worse generalization ability and so on. In order to eliminate these problems, it can consider designing Hybrid Neural Networks (HNNs) composed of multi single neural network to replace the complex single neural network. In

addition, it proposed a novel semi-supervised learning algorithm – deep learning approach using Deep Belief Network embedded with Soft max regress (DBNESR) as a classifier.

With the advent of "big data" era, the traditional standalone serial-based training machine learning has been difficult to meet the needs of "big data" applications. To this end, this paper discusses and has realized neural network learning algorithm based on cloud computing and an affective computing research based on cloud computing clusters. Namely with the help of a cloud computing platform through network circulation and combination to provide computing power as super computer to realize parallel training and classification recognition application of RBF neural network and the relevant algorithm, so that to make neural network and the relevant algorithm can study and process mass, high-dimensional data by cross-platform.

OVERVIEW ON CLOUD COMPUTING

Cloud computing involves a large number of computers connected through a communication platform such as the Internet. It is similar to utility computing. Cloud computing is also defined as a large-scale distributed computing model. Here, the cloud providers and users can have their own private infrastructure, and several types of services can be provided to clients using virtual machines hosted by the providers. It in- eludes utilization techniques for improving the efficiency of the system. These include: Network Utility, Disk I/O utility, CPU utilization of a system as well as available memory for performing operations. Cloud Computing is a term that renames some common technologies and techniques that we know in IT. It can be understood to mean data center hosting [1-2]. The principal concept of computing goes back to the 1950s.

At this time, large-scale mainframe computers became available, accessible via thin clients or terminal computers. These were referred to as "static terminals" because they were used for communications but had no internal processing capacities. To make a more resourceful use of high cost mainframes, a technique evolved which allowed multiple users to share physical access to the computer from multiple terminals just as the CPU time. Cloud computing providers offer their services in several fundamental models. These models include: infrastructure as a service (IaaS), platform as a service (PaaS), and software as a service (SaaS). Here, IaaS is the simplest and highest model which extracts details from the lower models [3-5]. It shows the architecture of cloud computing layers. Additionally, there are three layers that are not provided as user services.

This includes things like the processing power resources on each node and bandwidth resources on the links that the embedding must fulfill. For example, to run an experiment, a researcher may need 1 GHz of CPU for each virtual node and 10 Mbps for each virtual link. Added to these requirements are the constraints on location and any link propagation delay problems. In another example, a gaming service needs virtual nodes in many cities, as well as virtual links with very small propagation delays. These combinations of node and link constraints would make the embedding problem computationally difficult to approach and solve [6-7]; figure 1 show a simple virtual network request and how it was mapped on a substrate network.

PARALLEL IMPLEMENTATION MODEL AND BP ALGORITHM BASED ON CLOUD COMPUTING

Parallel implementation is a worldwide cooperation of developers and cloud computing technologists producing the universal open source cloud computing platform for public and private clouds. The Parallel implementation project aims to bring solutions for all types of clouds. The goal here is to make it simple to implement, scalable, and rich in features. Open- Stack technology contains a series of unified projects working together to deliver various components for cloud infrastructure solutions. Over 200 companies joined the Parallel implementation project. Some of these companies included leading IT Companies like Arista Networks, Brocade Communications Systems, AT&T, AMD, Canonical, EMC, Ericsson, FS Networks, Hewlett-Packard, Go Daddy, Cisco, Dell, Groupe Bull, IBMNEC, NetApp, Nexenta, Rackspace Hosting, Red Hat, Inktank, Intel, SUSE Linux, Oracle, VMware and Yahoo. This technology consists of a series of interrelated projects which control pools of processing, storage, and networking resources. It is able to be managed and provisioned through an interface dashboard. In the last few years, Parallel implementation has evolved from just a joint venture between NASA and Rackspace to build cloud infrastructure on product hardware, to ad hoc development projects for the factory industry.

In 2012, the parallel implementation community was deeply involved in core technology development, as well as deploying and managing parallel implementation projects [8]. Currently, parallel implementation consists of seven core components: Compute, Object Storage, Block Storage, Network, Dashboard, Image Service and Identity. Few studies or research has conducted to test the usage and the benefits of implementing virtual network embedding strategies within the Parallel implementation cloud operating system architecture. Currently, there is much need for this type of research since Open-Stack is a new and emerging Internet technology, and as such, is facing a resources al- location problem. Parallel implementation is the leading cloud computing technology and it has now received attention from members of the scientific community.

In this section, I provided a brief introduction to Parallel implementation, including its components, development, and research status. I also provided a brief introduction into the current research and study implications for connecting Parallel implementation and virtual networking. In this paper, I will discuss the virtual network embedding problem

and the existing strategies for solving this problem. Parallel implementation is a cloud operating platform which controls resources of computing, networking and storage throughout a datacenter, in which all are managed through a dashboard that gives administrators control and authorizing their users to use resources through a web interface (dashboard). The goals of the Parallel implementation originality are to support cloud services and allow businesses to build cloud services in their own data centers. Parallel implementation is available under the Apache 2.0 license freely, referred knows as "the Linux of the Cloud" and also comparing to Eucalyptus and the Apache Cloud Stack projects. Parallel implementation has a modular architecture that currently has three main components: compute, storage and image service [9-10].

Parallel implementation Compute (Nova) – a controller for cloud computing and managing large networks of virtual machines (VMs) is important. Parallel implementation Object Storage – a storage system that provides support for both block storage and object storage is also important. Image Service – Service which provides discovery as well as registration for visual disk images. Among many Parallel implementation services and projects (the list is growing with every re-lease), only Compute is considered within this paper. Compute or "Nova" is the service responsible for providing a compute provisioning function to clouds. It can be considered as a management layer which operates on top of a free option of supported hypervisors, exposing a REST API for the purpose of management and provisioning. It consists of a set of service binaries that work together to accomplish one common goal. They all interact directly through messaging and through a shared state which is stored in a central database. This is shown in figure 2. To interact with other services, we can directly target the REST API or use the python language provided by in the python-Nova client library. This also includes a command-line client. Other interfaces, such as the web-based Dashboard, use this as client libraries for interacting with the different Parallel implementation services as well. Pro-visioning requests, which enter the API and then pass the initial authorization and verification, will step before being sent out to the Nova- scheduler to decide which one of the available compute nodes should be handling the request. Our main focus of this section is the customization of the Nova Parallel implementation main component. The actual code for the Nova services are in. /nova and the corresponding unit tests are in the related directory under ./nova/tests. The following represents a short explanation of the Nova source directory structure.

The basic equation of key algorithm is shown as the equation (1) [11-12]:

$$(N, sk) \leftarrow Key(1^k) \tag{1}$$

This formula is used to generate file checksum parameter which is denoted by:

$$\begin{aligned} &r \leftarrow \{0,1\}^k; sk \leftarrow \{e, d, r\}; \\ &Output\{N, sk\}; \end{aligned} \tag{2}$$

The Euler function is:

$$\phi(N) = (p-1)(q-1) \tag{3}$$

Then choose an integer e to satisfy the following equation 4:

$$\begin{cases} 1<e<\phi(N) \\ \gcd(e, \phi(N))=1 \end{cases} \tag{4}$$

Then finally export (N, sk) in Tag algorithm, we can get the optimization equation (5):

$$(T_0, T_2, ...T_{n-1}) \leftarrow Tag(pk, sk, m) \tag{5}$$

The formula generates labels for each file block.

$$for(j = 0; j \le n-1; j++); \tag{6}$$

$$\begin{aligned} &\{W_j = r*(j+1); T_i \\ &= [h(W_j)*m_j]^c \bmod N\}; \end{aligned} \tag{7}$$

$$Output(T_0, T_2, ...T_{n-1}); \tag{8}$$

And local fractional integral of $f(x)$ defined by Eq.9.

$$\begin{aligned} {}_aI_b^{(\alpha)} f(t) &= \frac{1}{\Gamma(1+\alpha)} \int_a^b f(t)(dt)^{\alpha} \\ &= \frac{1}{\Gamma(1+\alpha)} \lim_{\Delta t \to 0} \sum_{j=0}^{j=N-1} f(t_j)(\Delta t_j)^{\alpha} \end{aligned} \tag{9}$$

In which, $T(\nabla) = \begin{Vmatrix} T_{ik}(\nabla) & t_i(\nabla) \\ t_k^T(\nabla) & -\tau(\nabla) \end{Vmatrix}$, $\mathsf{J} = \begin{Vmatrix} \delta_{ik} & 0 \\ 0 & 0 \end{Vmatrix}$,

$$f(x, \omega) = \begin{Vmatrix} u_k(x, \omega) \\ \varphi(x, \omega) \end{Vmatrix} \tag{10}$$

$T_{ik}(\nabla) = \partial_j C_{ijkl} \partial_l$, $t_i(\nabla) = \partial_j e_{ijk} \partial_k$, $\tau(\nabla) = \partial_i \eta_{ik} \partial_k$

Consider an infinite situation; we have the equation (5) in the following:

$$\mathbb{L}^0 = \begin{Vmatrix} C_{ijkl}^0 & e_{kij}^0 \\ e_{ikl}^{0T} & -\eta_{ik}^0 \end{Vmatrix} \tag{11}$$

Consider the propagation, instead the equation (11) with the following form:

$$\eta(\mathrm{x}) = \eta^0 + \eta^1(\mathrm{x}) \ , \rho(\mathrm{x}) = \rho_0 + \rho_1(\mathrm{x}) \tag{12}$$

Then we have equation (13) to (14):

$$C^1 = C - C^0, e^1 = e - e^0 \tag{13}$$

$$\eta^1 = \eta - \eta^0, \rho_1 = \rho - \rho_0 \tag{14}$$

For such kind of material, general form of equation (10) is expressed as following equation (15-17):

$$G_{ik}(\overline{k},\omega) = \frac{1}{\rho_0\omega^2}[\frac{\beta^2}{\overline{k}^2-\beta^2}\theta_{ik} + \overline{k}_i\overline{k}_k\left(\frac{1}{\overline{k}^2-\alpha^2}-\frac{1}{\overline{k}^2-\beta^2}\right)+m_i m_k\frac{\beta_\perp^2}{\overline{k}^2-\beta_\perp^2}] \tag{15}$$

$$\mathcal{G}_{ik}(\overline{k},\omega) = -\frac{1}{\eta_{11}^0}\frac{1}{\overline{k}^2}+\frac{1}{\rho_0\omega^2}\left(\frac{e_{15}^0}{\eta_{11}^0}\right)^2\frac{\beta_\perp^2}{\overline{k}^2-\beta_\perp^2} \tag{16}$$

$$\gamma_i(\overline{k}_i,\omega) = \frac{1}{\rho_0\omega^2}\left(\frac{e_{15}^0}{\eta_{11}^0}\right)^2\frac{\beta_\perp^2}{\overline{k}^2-\beta_\perp^2}m_i \tag{17}$$

Parallel implementation consists of a modular architecture along with various codes for the components. It also has several shared services which extend the three main components (compute, storage and networking). This makes it much easier to implement and operate on your own cloud. These services integrate the Parallel implementation components with each other along with external systems to deliver an integrated experience for users.

1. Parallel Implementation Computer nova is a cloud computing controller (considered the IaaS system's main component). It uses Python language as well as many external 1ibraries like Event let, Kombu (for AMQP communication), and SQL Alchemy (for database access). Nova's architecture is designed to be scaled horizontally on substrate hardware with no additional hardware or software requirements. It simultaneously pro- vides the ability to integrate with current legacy systems and third party technologies. Nova is designed to manage pools of computer resources and it can work with almost all available virtualization technologies. It can also work with high-performance computing (HPC).
2. Parallel Implementation Object Storage (Swift) is a scalable storage system in which objects and files are written to multiple disks in the data center servers. Parallel implementation will ensure data replication and integrity across the cluster. Storage clusters scale simply by adding new servers. If a server or a hard driver fails, Parallel implementation replicates its content to new locations in the cluster. Due to the fact Parallel implementation uses software logic to ensure data replication and distribution across different devices, inexpensive servers and hard drivers can be used.
3. Parallel Implementation Block Storage (Cinder) provides the software to create and man- age a service that provisions storage in the format of block devices known as Cinder volumes. Cinder provides persistent storage to guest virtual machines (instances) which are managed via Parallel implementation Compute. It can also be used independently with other Parallel implementation services.
4. Parallel Implementation Network (Neutron) provides virtual networking service for the compute module. The networking module can manage IP addresses which allows for dedicated DHCP or IPs. Network module suits the requirements of applications or user groups.
5. Parallel Implementation Dashboard (Horizon) gives users and administrators an interface to access, automate and provide cloud resources. The design makes it easy to plug in as well use third party products and services. This includes such things as monitoring, billing, and management tools. The dashboard can be customized for service providers and vendors who desire to use it. Dashboard is also a way to interact with Parallel implementation software resources.
6. Parallel Implementation Image Service (Glance) provides location and delivery for services for disks and server images. It has the ability to snapshot and copies a server image and stores it away. This is a something very useful about the Parallel implementation cloud operating system. The stored snapshots and images can be used to get servers mining faster and more consistently. It also can be used to catalog and store many backups.
7. Parallel Implementation Identity (Keystone) controls the central directory of users mapped to the Parallel implementation services and can access it. It works as an authentication system across every part of the cloud operating system and can be integrated with already existing backed directory services such as LDAP. The Keystone module supports many types of authentication, including username and passwords credentials, as well as AWS-style and token based systems.

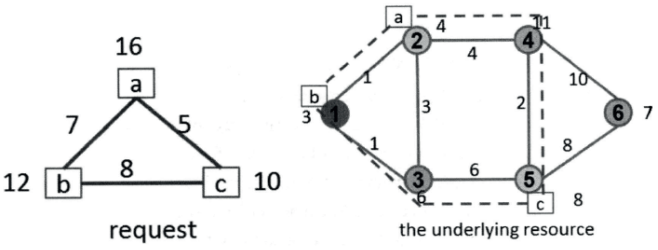

Fig. 1. Virtual Network Embedding

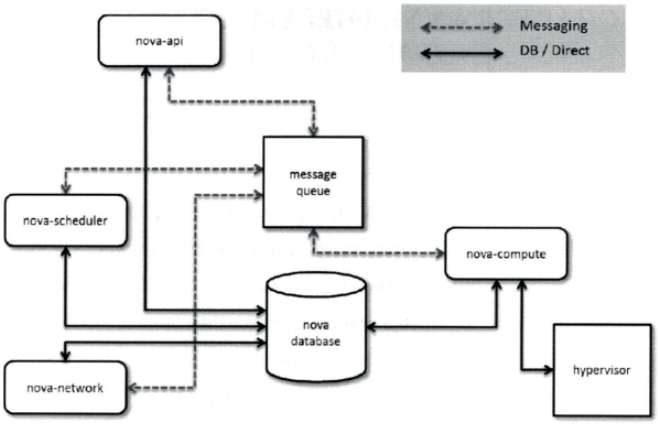

Fig. 2. Parallel implementation Compute architecture overview

EXPERIMENT AND DISCUSSION

The Hardware Layer and the Virtualization Layer are operated by the cloud services provider, while the Client Layer is provided by the end users. Basically, clouds can be defined in three ways:

1) Private Clouds: where data and processes are managed in the organization without security exposures and legal require- menu;
2) Public Clouds: where a set of computers and computer network resources, in which a service provider provide resources (like storage and applications) is available to the general public over the Internet;
3) Hybrid cloud: a combination of two or more clouds (private, public and community) that remain different entities but bound together. These provide the benefits of multiple cloud models. Therefore, hybrid cloud means the ability to connect, manage and dedicate services with cloud resources. A hybrid cloud service crosses isolation boundaries so it cannot simply be categorized as a private, public or community cloud. Virtual network embedding (VNE) has been a major challenge for future Internet (FI). The problem of embedding virtual networks within a substrate network is the main resource allocation in network virtualization.

Parallel implementation Compute scheduler is also known as the nova-scheduler service. It is responsible for mapping instance requests onto the physical hosts named compute nodes. Compute nodes will execute the nova-compute service on top of a hypervisor. When the scheduler service is launched, it will load a scheduler driver which holds the actual scheduling logic and policies. A scheduler driver is derived from a base driver class and implements interface. A number of simple scheduler drivers are included in Parallel implementation Nova. Advanced filters can be written as long as they able to implement the required interface of the base driver. In addition to defining the interface requirements, it also holds some of the basic requirements which are needed by every scheduler; this includes easy access to the global system state and some utility methods used by most schedulers. The figure 3 shows us the existing network and connected virtual bridges. When entering the command to view virtual bridge and virtual network port status we bet the figure 4.

```
root@IPL213:/home/openstack# ovs-vsctl list-br
br-eth1
br-int
root@IPL213:/home/openstack# ovs-vsctl list-ports br-eth1
eth1
phy-br-eth1
root@IPL213:/home/openstack# ovs-vsctl list-ports br-int
int-br-eth1
tap215779b8-aa
tap24ebecf2-1a
tap556ebc58-c8
tap9f9a565a-e1
tapba36604e-e9
```

Fig. 3. Obtain VN information

```
root@IPL213:/home/openstack# ovs-ofctl show br-int
OFPT_FEATURES_REPLY (xid=0x1): ver:0x1, dpid:00007e2e3d4a0940
n_tables:255, n_buffers:256
features: capabilities:0xc7, actions:0xfff
 1(tap556ebc58-c8): addr:fe:16:3e:af:ff:54
     config:     0
     state:      0
     current:    10MB-FD COPPER
 2(tap215779b8-aa): addr:fe:16:3e:6e:4a:81
     config:     0
     state:      0
     current:    10MB-FD COPPER
 4(tap24ebecf2-1a): addr:fe:16:3e:39:86:fd
     config:     0
     state:      0
     current:    10MB-FD COPPER
 5(tap9f9a565a-e1): addr:fe:16:3e:ba:d2:bc
     config:     0
     state:      0
     current:    10MB-FD COPPER
 6(tapba36604e-e9): addr:fe:16:3e:97:52:f5
     config:     0
     state:      0
     current:    10MB-FD COPPER
 7(int-br-eth1): addr:8a:bc:e5:63:0b:cb
     config:     0
     state:      0
     current:    10GB-FD COPPER
 LOCAL(br-int): addr:7e:2e:3d:4a:09:40
     config:     PORT_DOWN
     state:      LINK_DOWN
OFPT_GET_CONFIG_REPLY (xid=0x3): frags=normal miss_send_len=0
```

Fig. 4. Virtual bridge status

The target of the VNE problem is the allocation of virtual resources in nodes and links. Therefore, it is divided in two sub problems: first, Virtual Node Mapping (VNoM), where virtual nodes are mapped in physical nodes; and second, Virtual Link Mapping (VLiM), where the virtual links linking virtual nodes have to be mapped on paths connecting these nodes in the substrate network. Future Internet architectures have been evolving to become based on the Infrastructure as a Service (IaaS). This is a model that divides the role of current Internet Service Providers (ISPs) into two new main roles: first, the Infrastructure Provider on P which deploys and keeps the network equipment operating; and second, Service Provider (SP). Service provider (SP) is responsible for deploying various network protocols and providing end to end services. For example, Voice over IP (known as VoIP) can run on a virtual network which provides anticipated performance.

This is done by provisioning dedicated resources and employing routing protocols which can ensure fast recovery from any equipment failures that might occur. On the other

hand, online banking runs on a virtual network that provides security guarantees. This is done through addresses and secure routing protocols. Making efficient usage of the substrate resources demands effective techniques for visual network embedding (a new virtual network mapping). The VNE problem is extremely difficult for two main reasons, the first of which is node and link constraints. Each VN request has resource limitations.

CONCLUSION

For traditional technologies of artificial intelligence and pattern recognition and so on, are all faced with the challenge of how to implement the data processing in the era of big data. For example, in the classification of a large scale of face data sets, a computer or workstation is very difficult to adapt to the actual requirements because of the lack of speed and storage capacity. Therefore, it is very necessary to study how to implement the technologies of artificial intelligence and pattern recognition based on multi-computer clusters in large data environment. When using the neural network in artificial intelligence to deal with the related data, if the number of size of training samples is not large, generalization ability and running time of single neural network are relatively ideal. Based on cloud computing clusters, this paper proposes a novel method of parallel implementation of multilayered neural networks based on Map-Reduce. Namely in order to meet the requirements of big data processing, this paper presents an efficient mapping scheme for a fully connected multi-layered neural network, which is trained by using error back propagation (BP) algorithm based on Map-Reduce on cloud computing clusters(MRBP). Experiment results demonstrate that the proposed parallel BP algorithm in this paper has better speed-up, faster convergence rate, less iterations than that of the existed algorithms.

BIBLIOGRAPHY

1. Bhandarkar S M, Wang X: *Efficient parallel implementation of the multi-layer perceptron on an SIMD mesh architecture.* Neural Parallel & Scientific Computations, Vol. 4, no. 1, pp. 69-82, 1996.

2. Li X J, Li L: *IP Core Based Hardware Implementation of Multi-Layer Perceptrons on FPGAs: A Parallel Approach.* Advanced Materials Research, Vol. 433, pp.:5647-5653, 2012.

3. Kalaitzakis K, Stavrakakis G S, Anagnostakis E M: *Short-term load forecasting based on artificial neural networks parallel implementation.* Electric Power Systems Research, Vol. 63, no. 3, pp.185-196, 2012.

4. Kim Y C, Shanblatt M A: *Architecture and statistical model of a pulse-mode digital multilayer neural network.* IEEE Transactions on Neural Networks, Vol. 6, no. 5, pp.1109-1118, 1995.

5. Hikawa H: *Frequency-based multilayer neural network with on-chip learning and enhanced neuron characteristics.* IEEE Transactions on Neural Networks, Vol. 10, no. 3, pp.:545-53, 1995.

6. Serpen G, Gao Z: *Complexity Analysis of Multilayer Perceptron Neural Network Embedded into a Wireless Sensor Network.* Procedia Computer Science, Vol. 36, pp.192-197, 2014.

7. Kumar A, Joshi H, P. S: *Neural Network Approach for Automatic Landuse Classification of Satellite Images: One-Against-Rest and Multi-Class Classifiers.* International Journal of Computer Applications, pp.134, 2016.

8. Raza M Q, Khosravi A: *A review on artificial intelligence based load demand forecasting techniques for smart grid and buildings.* Renewable & Sustainable Energy Reviews, Vol. 50, pp.1352-1372, 2015.

9. Ahmedalazzawi N: *Automatic Recognition System of Infant Cry based on F-Transform.* International Journal of Computer Applications, Vol. 102, no. 12, pp.28-32, 2014.

10. Druitt C M, Alici G: *Intelligent Control of Electroactive Polymer Actuators Based on Fuzzy and Neurofuzzy Methodologies.* Mechatronics IEEE/ASME Transactions on, Vol. 19, no. 6, pp.1951-1962, 2014.

11. Francesquini E, Castro M, Penna P H: *On the energy efficiency and performance of irregular application executions on multicore, NUMA and manycore platforms.* Journal of Parallel & Distributed Computing, Vol. 76, pp.32-48, 2015.

12. Li Y, Tang X, Cai W: *Play Request Dispatching for Efficient Virtual Machine Usage in Cloud Gaming.* IEEE Transactions on Circuits & Systems for Video Technology, pp. 1-11, 2015.

CONTACT WITH AUTHOR

Zhang Hu, Ph. D.,
e-mail: 61525963@qq.com
tel.: 13343408090

School of Information Engineering
Wuhan University of Technology
Wuhan Hubei, 430074
China