2014 International Joint Conference on Neural Networks (IJCNN)
July 6-11, 2014, Beijing, China

Parallelized Neural Networks as a Service

Altaf Ahmad Huqqani
Workflow Systems and Technology
Faculty of Computer Science
University Of Vienna
Austria,
Email:aahuqqani@gmail.com

Erich Schikuta
Workflow Systems and Technology
Faculty of Computer Science
University Of Vienna
Austria,
Email:erich.schikuta@univie.ac.at

Erwin Mann
Workflow Systems and Technology
Faculty of Computer Science
University Of Vienna
Austria,
Email:erwin.mann@gmail.com

***Abstract*—We present a novel neural network simulation framework, which provides the parallelized execution of artificial neural network by exploiting modern hardware and software environments adhering to the service oriented paradigm within the N2Sky system. The goal of the N2Sky system is to share and exchange neural network resources, neural network specific knowledge, neural network objects and paradigms and, in turn, to deliver a transparent environment to novice and experienced users to do neural network research. The parallelization techniques are deployed transparently for the user reducing significantly the time consuming training of neural networks by selecting appropriate service implementation according to complexity of the problem. N2Sky follows the sky computing paradigm fostering ample resources by using federated clouds.**

I. Introduction

Neural networks are efficient to solve problems where mathematical modeling of the problem is difficult. They are used to overcome problems including predictions, classifications, feature extraction, image matching and noise reduction. There are two things that must taken care to make neural networks feasible. One is to present the neural network resources, objects and paradigms to the users and researchers transparently and accessible over the internet. Second is to reduce the training time of the neural networks as the time required to train increases exponentially with the size of the data.

The cloud computing paradigm provides access to large amounts of computing power by aggregating resources both hardware and softwares and offering them as a single system view. It hides the details of implementation and management of softwares and hardware from the end user. Could computing has been evolved from technologies like cluster computing and grid computing and has given rise to sky computing [17]; an architectural concept that denotes federated cloud computing. Sky computing is an emerging computing model where resources from multiple cloud are leveraged to create a large scale distributed infrastructure. The cloud computing was originally intended for business oriented approach. High performance computing (HPC), which was confined to super computers (strictly preserved system software) or dedicated grids, is making in-roads in cloud computing paradigm facilitating scientists to build their own virtual machines and configure them to suit their needs and preferences.

N2Sky is an artificial neural network simulation environment facilitating the users to create, train, evaluate neural network providing different types of resources from clouds of different affinity, e.g. computational power, disk space, networks etc, on one hand and exploits the parallelization techniques to reduce the time consuming training phase on the other hand. The aim of the project is to facilitate the ever growing community of users that want to play around, test, and solve their problems without buying expansive hardware and worrying about the installation and configuration of the required software.

Typically large amount of data is required to train and evaluate neural networks. Many efforts are made to reduce the training time of the neural network by selecting initial values [19], controlling the learning parameters of a neural network [16] and determining weight adjustments [38]. Recent technology advancement in hardware and software enabled us to use capabilities of modern hardware and softwares interfaces, specifically by exploiting the parallelization capabilities of multicore/multithreaded CPUs or of graphic processing units (GPUs), which earned a strong research focus today. A modern GPU provides hundred of streaming cores and handles thousands of threads, which makes it specifically suitable for computation-intensive applications like neural network simulation.

In N2Sky we have explored two software environments, CUDA and OpenMP, which have established themselves as quasi-standard for GPU and multicore systems respectively. The Compute Unified Device Architecture (CUDA) is a parallel computing platform and programming model invented by NVIDIA [4]. It provides set of extensions to standard programming languages, like C, that enables implementation of parallel algorithms. Since the launch of the CUDA architecture many applications were developed and more features were added to new GPUs. The OpenMP Application Programming Interface is a powerful and compact programming model for shared memory parallelism in C/C++ and FORTRAN in multicore servers [20]. It provides directives, library routines, and environment variables to manage parallel program across the different processors in a server. OpenMP exits in industry since the 90's and became a de-facto standard to exploit the capabilities of available multi-core processors with virtualization and hyper-threading with shared memory.

In this paper we extend our prior work of CUDA on GPU and OpenMP on multicore CPUs for the parallelized simulation of a neural network based face recognition application and make these features available to end users by incorporating them into the N2Sky system. We provide these parallelization schemes transparently to the users allowing for

978-1-4799-1484-5/14/$31.00 ©2014 IEEE

automatic fostering of available parallel resources aiming for high performance execution of network tasks.

The paper is structured as follows: Section II presents the state of the art and elaborates the recent research trends. Section III describes the details of the transparent parallelization approach. In section IV we present the deployment scheme on federations of clouds followed by the description of the N2Sky architecture and the scientific workflow in section V. A practical use case in section VI discusses the results obtained and derive our recommendations. Finally, section VII gives a summary and future directions of our work.

II. Related Work

The computational intelligence community developed many neural network simulators. NeuroWeb [26] exploits internet-based networks as transparent layer to exchange neural network information, NeuroAccess [3] integrates neural networks into relational database, Emergent Neural Network Simulation System [5] focuses on easiness for user to create, train and test the neural network by providing drag and drop facility of components and XNBC [39] a tool for neurobiologists to simulate and analyze the behavior of simulated biological neurons and neural networks, to name a few. Most of these simulators are stand alone application, developed to serve a particular purpose and need to be configured before using them. We have developed N2Cloud [12] which is based on the Service Oriented Architecture (SOA) and is an Evolution of N2Grid [25]. We further extend the idea of N2Cloud to N2Sky [28] environment using sky computing paradigm with enhanced features of best execution environment of the service.

There are many efforts in the research community to bridge the gap between HPC and cloud computing. In [18] traditional HPC, grid and cloud in terms of their motivation, strengths and weaknesses are discussed. It combines the important characteristics of each proposing a model for how these paradigms can work together and how scientific workloads can be managed terming it as *hybrid computing environment*. The authors in [1] presented a gap analysis between the HPC and cloud computing resources in terms of price; pay-as-you-go model. They describe the difficulties and performance of both computing paradigms. In [11] the authors presented an approach that makes use of InfiniBand clusters for HPC cloud computing. They also proposed a performance-driven design of HPC IaaS layer for InfiniBand, which provides throughput and latency-aware virtualization of nodes, networks, and network topologies, as well as an approach to an HPC-aware, multi-tenant cloud management system for elastic virtualized HPC compute clusters. While in [21] author introduced a service composition Framework for Market-Oriented High Performance Computing Cloud by an ontology that describes dependencies and relationships among HPC software and resources. The authors in [10] focused on evaluating the technical capability of current public cloud computing platforms (Amazon EC2, IBM, GoGrid), and their suitability for running scientific HPC applications comparing these public clouds by running standard benchmark and NASA climate prediction application.

In the course of exploiting cloud resources for HPC we deliver the N2Sky environment which offers neural network resources as a service which dynamically uses the available computing environment to reduce the execution time. Summing up, the N2Sky environment provides:

- Sharing of neural network paradigms, objects and related information between the researchers and end user world wide.
- Reduction of training time of neural network by automatically selecting appropriate parallel implementations of the neural network services exploiting suitable cloud resources.
- Transparent access to High-end neural network resources stored in cloud environment.
- Uniform Look and feel for location independence of computational, storage and Network resources.

III. Parallelization of Neural Network Task Execution

Neural networks proved extremely well for solving problems, which are hard to catch in mathematical models. However, the usage and employment of neural networks in such application domains is often dependent on the tractability of the processing costs. The problem domains for the employment of neural networks are increasing and also the problems themselves are getting larger and more complex. This leads to larger networks consisting of huge numbers of nodes and interconnection links, which results in exceeding costs for the network specific operations, as evaluation and training. Especially the cost intensive training phase of a neural network inherits a major drawback, due to the situation that large numbers of patterns (input and/or target values) are fed into the network iteratively. Thus, a key aspect of an artificial neural network simulation infrastructure to be accepted by the community is the efficient execution of the calculation intensive training phase. High performance computing applications typically apply specialized hardware and software infrastructure to speed up performance by parallelization of task execution. However, parallelization of applications is a difficult and error-prone task, very often leading to hand-crafting of code by experts.

Many research on parallelization of neural network simulation are done in the past. Also the authors of this paper did ample research on this issue. We focused on different neural network paradigms, as Backpropagation[23], Kohonen [22], and cellular neural networks [34], and various parallel infrastructures, as hyper-cube [24], cluster [32], GPU and multicore systems [13]. As result of our work we developed set of rules [34], [35] for the simplified development of parallel execution scenarios, which are even applicable for the arbitrary user. To make parallelization feasible for arbitrary users we believe that it has to be done automatically and transparently by the simulation system.

Therefore, our vision is a smart system, which administrates parallelized execution patterns for neural network simulation in a repository and selects applicable schemes based on given information on networks, problem and available infrastructure using expert know-how stored in a rule-based knowledge base.

In a first attempt[13] we concentrate on two parallelization approaches motivated by our problem domain and the available hardware resources: structural and topological data parallelism.

- **Structural Data Parallelization**: The structural data parallelization techniques is used for cluster and multicore systems. The training images are divided into disjoint sets. Identical copy of neural networks are created for each thread. Each thread is trained on its own data. After certain number of epochs the weights of all the threads are collected, updated and broadcasted to the threads again. This process is continued unless the error is less than the defined threshold value.
- **Topological Data Parallelization:** The topological node parallelization is deployed on GPU using CUDA. In this approach only one copy of the neural network is instantiated, which resides on GPU. Each thread on the GPU behaves like a neuron and executes independently. To speed up the implementation the training weights and input data are stored in an one dimensional array aligned with the host and the device memory for looping in GPU.

The complexity of neural network increases as the number of neurons in hidden layer increases. So we defined the problem size as the product of neurons of the input layer and the hidden layer. Choosing a particular hardware and software configuration takes basically into considerations neural network paradigm and the problem size. Figure 1 presents the dependencies of absolute execution time of the neural network training as we changed number of neurons in input and hidden layers for openMP (optimal number of threads) and GPU implementations. This figure basically represents a decision map illustrating which parallelization scheme is to be applied for what problem characteristics.

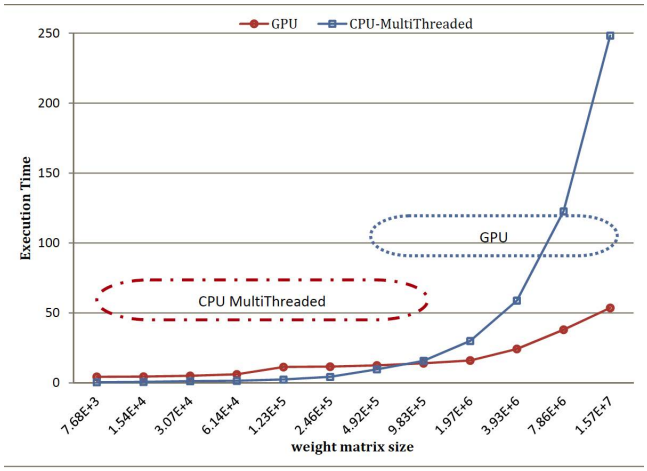

Fig. 1. Selection of openMP or GPU

The core of our envisioned system is a knowledge base which consists of set of rules, which selects parallelization schemes for given neural network paradigms, available parallel hardware and software infrastructure and problem characteristics.

The realization of this knowledge base is done by semantic web tools. The W3C has published the XML based standards Ontological Web Language (OWL) and the Resource Description Framework (RDF) for defining ontologies. As these technologies are platform independent, exchangeable, comprehensive and widely-accepted we use these technologies to build our architecture. For the management of the knowledge base we apply SPARQL [31] and Jena [15]. Summing up, RDF and OWL are used for ontologies, whereas SPARQL is used to query these ontologies, and Jena is used to execute rules from the knowledge base.

IV. N2Sky Cloud Deployment

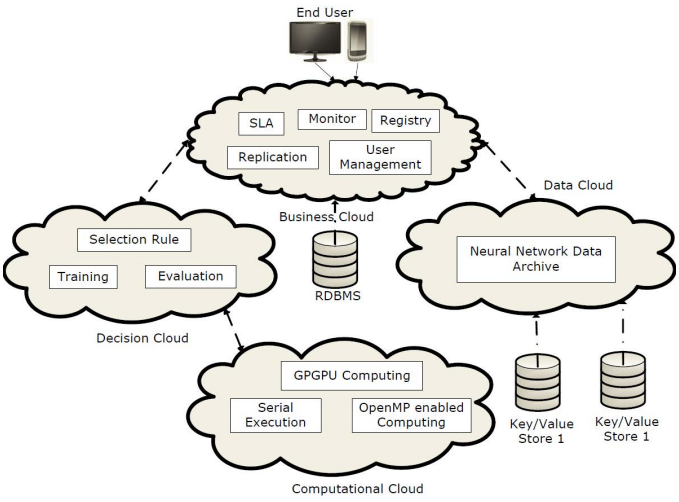

Fig. 2. N2Sky Cloud Deployment

We deployed N2Sky on Eucalyptus [6]. Eucalyptus is an open sources software framework (like Amazon Elastic Compute Cloud) for cloud computing by implementing Infrastructure as a Service (IaaS); that provides users the ability to run and control instances deployed across a variety of physical resources. The N2Sky design approach allows easy portability to other cloud computing platforms.

The N2Sky system is Java-based environment providing simulation of neural networks by using Apache Axis Library and Apache Tomcat Web container as hosting environment for the web services. To access these services Java Servlets/JSPs have been deployed as the web frontend. N2Sky can be deployed as a federated clouds model by fostering the specific affinities (capabilities) [14] of different cloud providers like data/storage clouds and compute clouds etc. A possible specific deployment is shown in Figure 2. Four different clouds Business Cloud, Data Cloud, Decision Cloud and Compute Cloud are depicted providing unique functionalities. Data Cloud provides ample storage resources by accessing relational or NoSQL database systems. The Business Cloud administrates the user management, SLA management, business logic and act as central access point to N2Sky system. The Decision Cloud stores in the parallelization knowledge base the description about various neural network parallelization schemes as a set of rule (classical Horn clauses). This knowledge base can be extended by knowledge from other web services as well. The Decision Cloud acts like an agent using service selection rules for training and evaluation for particular service implementation to execute tasks on the Computational Cloud, which provides the specific hardware resources, as Multicore

Servers, GPUs etc. and software environments, as OpenMP etc.

V. N2Sky Architecture

The architecture and the systems components are depicted in Figure 3.

N2Sky Simulation Service: This service provides three basic functionalities, as ***Train***, ***Retrain*** and ***Evaluate***. In the **Train** phase the user provides input and output data sets to the neural network and performs the training. In the **Retrain** phase an already trained neural network is provided with additional input data sets so that its accuracy can be increased. In the **Evaluation** phase the trained neural network calculates the solutions to the given problem. This service also uses the Paradigm Replication Service which provides information and starts paradigm specific cloud instances. The Simulation Selection Service caters the selection of the hardware environment on which the simulation has to be executed according to the rules of the parallelization in knowledge base.

N2Sky Simulation Selection Service: Different implementations of a service running on different hardware environments can have different execution times as shown in Figure 1. The N2Sky Simulation Service chooses the most appropriate implementation of the service doing the job in minimal time transparently to the end user. In a self-adaptive fashion it dynamically updates its knowledge base learning from the executed simulations.

N2Sky Paradigm Execution Service: This service resides on the physical resources, sequential and parallel ones. The N2Sky Paradigm Replication service executes the required implementation of the service in co-ordination of N2Sky simulation Service by using this service. It also provides status during execution of the service as well as results at the end of the execution of the simulation.

N2Sky Data Archive: This service provides key information about already trained neural network, available paradigms, neural networks object, evaluation data sources, input data sets and output data sets for training and retraining of the neural network by publishing. It provides two methods to store and retrieve data from the data archive. The **put** method stores a neural network object and it corresponding input and output results, total execution time as well as the environment on which the simulation was carried out. The **get** method fetches all information regarding the a particular neural network from the data source when needed.

N2Sky Database Service: Already trained neural network object and paradigm are available to end users. This functionality provides the N2Sky Database Service as it dynamically updates itself, by storing input/output data sets, paradigm used, weight and error matrix, simulation environment, by storing information as a particular training/retraining/evaltion simulation ends. These results are then published by the N2Sky Data Archive service so that these are available to users.

N2Sky Service Monitor: This is the entry point of the whole system to end user. This service keeps track of available neural network services and publishes these services. This module enables the N2sky user to select either already published paradigms like Back Propagation, Quick Propagation, Jordan etc. or submit its own modified paradigm by defining or selecting input / output data sets or parameters. Hardware virtualization play important role in presenting the resources and services transparently to end user to interact with this system.

N2Sky Paradigm / Replication Service: It contains the business logic of the neural network implementation and uses the Paradigm Execution Services to install / start new instance in the cloud environment.

N2Sky Registry: This services plays a key role in the whole system. This service publishes the available paradigm, neural network objects, input / output data sets to end user. It also provides the information / status of a simulation during the execution and at finish stage of the simulation. It co-ordinates between Simulation Selection Service, Paradigm Replication Service and the N2Sky Web Portal.

N2Sky Java Application / Applet: The aim of N2sky system is to provide the end user (experienced or novice) predefined neural network objects and paradigms. The system presents an application/applet which is a Java based Graphical User Interface (GUI) to interact with it. This interface allows the users to solve their problems by using either predefined neural network objects or by customising paradigms and defining their own parameters.

N2Sky (Mobile) Web Portal: The N2Sky (Mobile) Web Portal provides the access point to the N2Sky system by a web browser interface which can be used on PCs, tablets or even a smart phones.

N2Sky User Management Service: This service provides session ID and check credentials.

Figure 3 shows the layout of the system services and the execution workflow is described (the numbers refer to the labels in the figure).

1) During the Publishing process all the components update themselves by fetching information from the N2Sky Data Archive Service and Database Services and register the available resources and services to the N2Sky Registry and the N2Sky Service monitor.
2) The Paradigm Replication Services activates and maps available paradigms to running instances in the Paradigm Execution Service.
3) Via web browsers users login into the N2Sky system. The N2Sky Services Monitor sends users login request to User Management and Access Control Services.
4) User Management and Access Control Services check the Users Credentials.
5) User Management and Access Control Services send a new Session ID or access rejection.
6) The user is provided with either a new Session ID or the access is denied.
7) On successful login the user is able to query the available neural network Paradigms.
8) The user is provided with the query results. He can select a Paradigm or can modify it by defining own parameters.
9) The user requests Creating / Training / Retraining / Evaluation of the neural network.

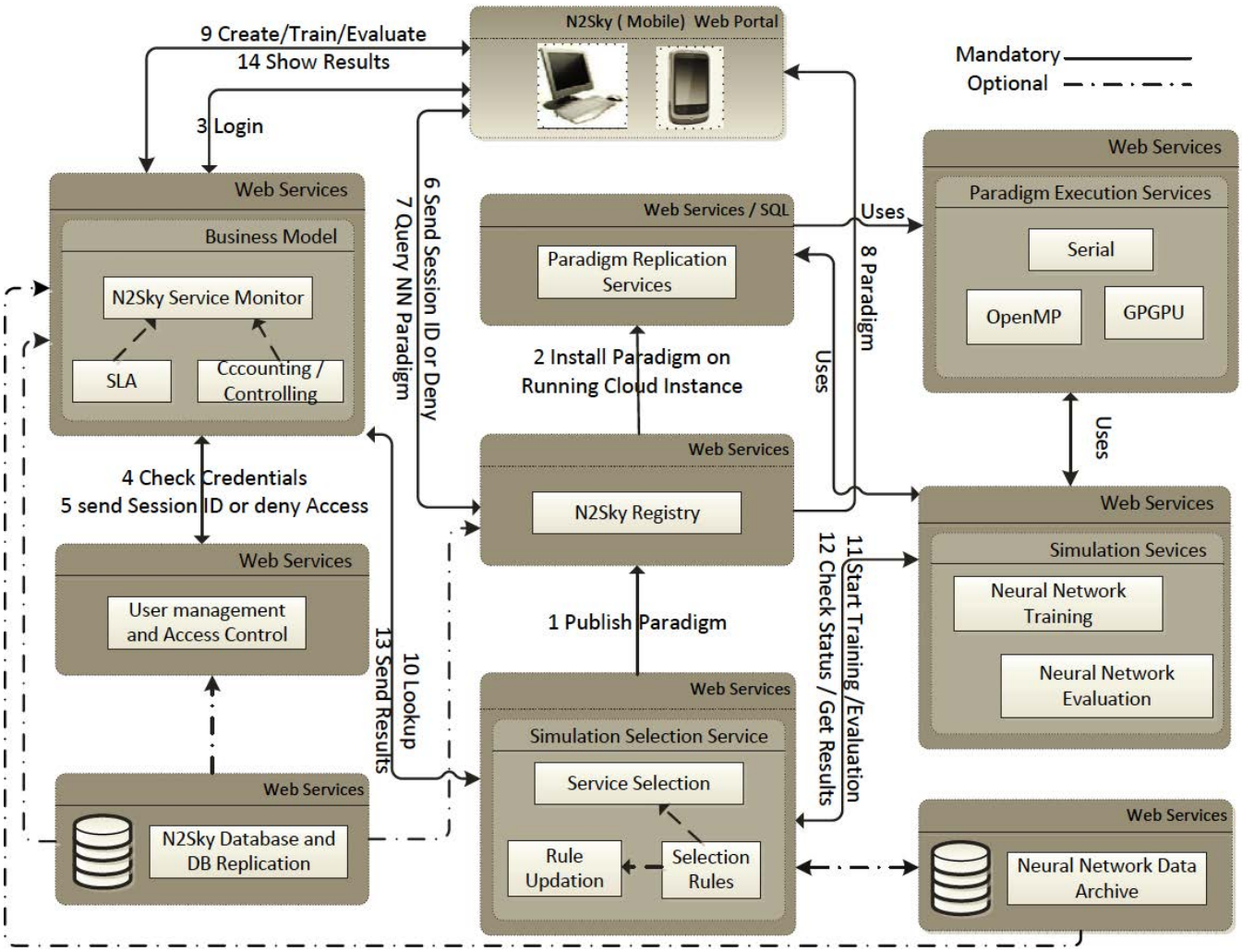

Fig. 3. Parallelized N2Sky Architecture and components

10) The system receives the simulation request and lookups the Simulation Selection Service rules to select a service implementation on sequential or parallel resource as to minimize the execution time of the simulation.
11) The Simulation Service starts Training / Evaluation of the selected neural network implementation by using the Paradigm Execution Service.
12) The Simulation Service sends the status / results to the Simulation Selection Services which updates its selection rules if new knowledge is available.
13) The results are sent to N2sky Service Monitor.
14) The simulation results are presented to the user.

VI. N2Sky Use case

N2Sky user interface is driven by guiding principles of acceptance, simplicity, originality, homogeneity and system extensibility. Using HTML5, CSS 3.0 and JQuery we are able to provide the access to user through web browser (Safari, Chrome, Mozilla Firefox or the Internet Explorer) as well as from tablets, Macs and smart phones. Figure 4 shows the login screens for N2Sky on IPhone and Mac. For a detailed interface walk-through see [28].

Fig. 4. N2Sky Login Screens iPhone(left) and Mac (right)

As use case we target the face recognition problem by a Backpropagation neural network (BPNN) trained by a supervised learning mode. The face images we used are from [29]

and are available in a pgm-p2 format. These images contain faces of 20 people, in various head poses (left, right, straight, up), various expressions (neutral, happy, sad, angry) and different eye status (open, closed). The images are available in different resolutions: $32x30$, $64x60$ and $128x120$ pixels. In our evaluation we used images in the resolutions $32x30$ and $128x120$. For each resolution 70 images were used as the TrainSet and 32 images were used as the TestSet – for the evaluation phase of a neural network – to verify the quality of the training.

Output values range from 0.0 to 1.0, where a high value (above 0.9) indicate that the image matches an assumed person. A low value (below 0.1) indicates that the image does not belong to the assigned person. After a feed-forward operation, the output value is compared with the target value and classified to rather high (above 0.5) or rather low (below 0.5), to check whether the BPNN has correctly classified the face image. Both multithreaded and GPU implementations make use of the existing timer function `clock_gettime()` to record the system time and calculate the overall execution time of a simulation run of the algorithm. For the two multithreaded versions of the BPNN face recognition algorithm we used the following hardware and software environment: the multithreaded CPU program was compiled by GCC 4.3.3 and runs on a dual Xeon X5570 machine (2x 2.93GHz quad-cores with hyper-threading, each 6GB memory at 1333MHz). Thus totally 16 logical cores can be used. The multithreaded GPU program was compiled by CUDA NVCC 3.0 and runs on a Tesla C1060 graphics card (240x 1.296GHz streaming cores, 4GB memory at 800MHz).

For our simulation we used the same BPNN configuration (learning rate, momentum, number of neurons, initialized weights and number of epochs) for CPU (serial execution), OpenMP and GPU programs. Therefore we can directly compare the execution times of the different runs. For all runs we set the learning rate (0.3) and the momentum (0.3) and vary only the number of epochs (100 epochs for 960 inputs, 20 epochs for $15,360$ inputs) and the number of hidden neurons (from 8 to $1,024$). For 960 input neurons in the input layer the execution time for the serial execution increases dramatically as the numbers of hidden neurons increases. However the multi-threaded OpenMP program outperforms the GPU version having a better runtime for each variation of the hidden neurons. The Figure 5(a) and 6(b) which shows the reduction in execution time as compared to serial execution for 960 input neurons and 15360 input neurons respectively. As the complexity of problem increases GPU environment performs better by utilizing the streaming cores than the multi-threaded OpenMP version.

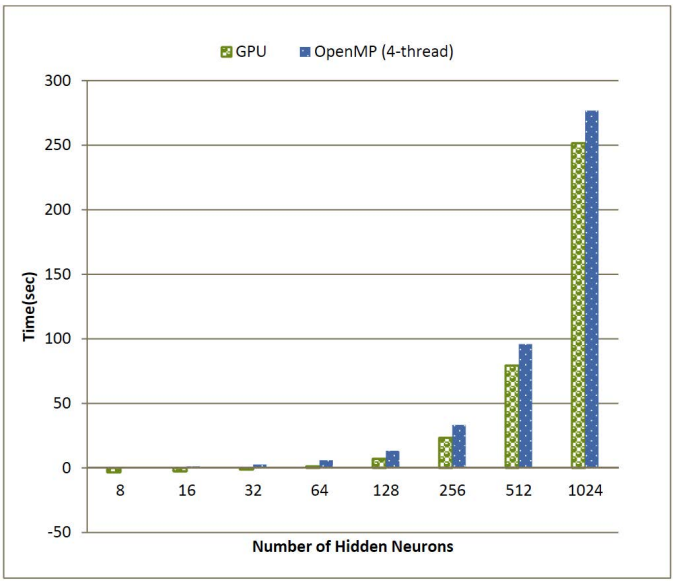

Fig. 5. BPNN Execution Time Reduction Analysis: Time reduction for (32x30)-100 Epochs)

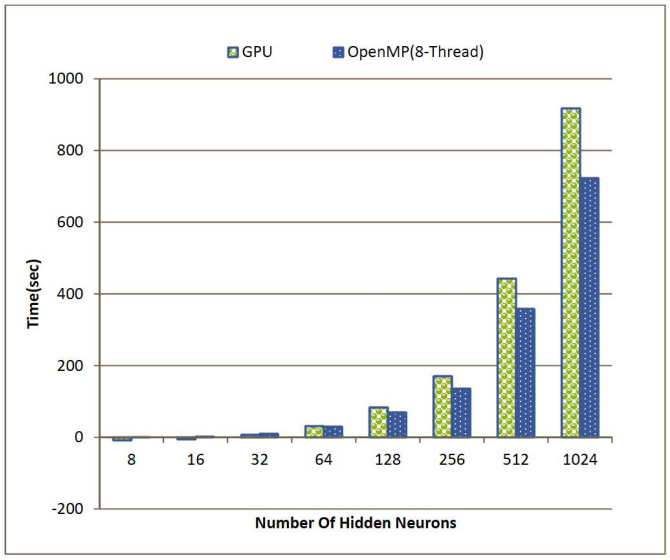

Fig. 6. BPNN Execution Time Reduction Analysis: Time reduction for (128x120)-20 Epochs)

VII. Conclusion & Future Work

We presented N2Sky, realizing neural networks as a service paradigm, which fosters cloud resources delivering a framework for the Computational Intelligence community to share and exchange neural network resources.

Parallelization of neural network training is key for increasing the overall performance. The specific focus of this paper lies on the parallelization mechanism which transparently delivers available high performance computing resources, as clusters, multi core architectures and GPGPUs, utilizing parallel processing schemes. Based on our research on neural network parallelization we envision an automatically definition and usage of parallelization patterns for specific paradigms.

N2Sky is a prototype system with quite some room for further enhancement. Ongoing research is done in the following areas:

- We are working on an enhancement of the neural network paradigm description language ViNNSL [2] to allow for automatic sharing of resources based on the formalized description.
- Key for fostering of cloud resources are service level agreements (SLAs) which give guarantees on quality

of the delivered services. We are working on the embedment of our research findings on SLAs [8], [9], [7] into N2Sky to allow for novel business models [37], [27], [36], [33] on the selection and usage of neural network resources based on quality of service attributes [30].

- A further important issue is to find neural network solvers for given problems, similar to a "Neural Network Google". In the course of this research we are using ontology alignment by mapping problem ontology onto solution ontology.

REFERENCES

[1] R. Aversa, B. Di Martino, M. Rak, S. Venticinque, and U. Villano, "Performance prediction for hpc on clouds," *Cloud Computing: Principles and Paradigms, R. Buyya, J. Broberg, and AM Goscinski, Ed., John Wiley &Sons*, pp. 437–454, 2011.

[2] P. P. Beran, E. Vinek, E. Schikuta, and T. Weishäupl, "ViNNSL - the Vienna Neural Network Specification Language," in *Proceedings of the International Joint Conference on Neural Networks, IJCNN 2008, part of the IEEE World Congress on Computational Intelligence, WCCI 2008*. IEEE, June 2008, pp. 1872–1879.

[3] C. Brunner and C. Schulte, "Neuroaccess: The neural network data base system," *Masters Thasis, University of Vienna, Vienna, Austria*, 1998.

[4] CUDA, "CUDA Specifications and Documentation," http://docs.nvidia.com/cuda/index.html.

[5] Emergent, "Emergent Neural Network Simulation System," http://grey.colorado.edu/emergent/index.php/Main_Page.

[6] Eucalyptus, "Eucalyptus Website," http://www.eucalyptus.com.

[7] I. U. Haq, R. Alnemr, A. Paschke, E. Schikuta, H. Boley, and C. Meinel, "Distributed trust management for validating sla choreographies," in *Grids and Service-Oriented Architectures for Service Level Agreements*, P. Wieder, R. Yahyapour, and W. Ziegler, Eds. Springer US, 2010, pp. 45–55.

[8] I. U. Haq, I. Brandic, and E. Schikuta, "SLA validation in layered cloud infrastructures," in *Economics of Grids, Clouds, Systems, and Services, 7th International Workshop, GECON'10*, ser. Lecture Notes in Computer Science, vol. 6296. Ischia, Italy: Springer Berlin / Heidelberg, 2010, p. 153–164.

[9] I. U. Haq and E. Schikuta, "Aggregation patterns of service level agreements," in *Frontiers of Information Technology (FIT'10)*. Islamabad, Pakistan: ACM, 2010.

[10] Q. He, S. Zhou, B. Kobler, D. Duffy, and T. McGlynn, "Case study for running hpc applications in public clouds," in *Proceedings of the 19th ACM International Symposium on High Performance Distributed Computing*. ACM, 2010, pp. 395–401.

[11] M. Hillenbrand, V. Mauch, J. Stoess, K. Miller, and F. Bellosa, "Virtual infiniband clusters for hpc clouds," in *Proceedings of the 2nd International Workshop on Cloud Computing Platforms*. ACM, 2012, p. 9.

[12] A. A. Huqqani, L. Xin, P. P. Beran, and E. Schikuta, "N2Cloud: Cloud based Neural Network Simulation Application," in *Neural Networks (IJCNN), The 2010 International Joint Conference on*, July 2010, pp. 1–5.

[13] A. A. Huqqani, E. Schikuta, S. Ye, and P. Chen, "Multicore and gpu parallelization of neural networks for face recognition," *Procedia Computer Science*, vol. 18, pp. 349–358, 2013.

[14] H. Jang, A. Park, and K. Jung, "Neural Network Implementation Using CUDA and OpenMP," *Digital Image Computing: Techniques and Applications*, vol. 0, pp. 155–161, 2008.

[15] A. Jena, "Reasoners and rule engines: Jena inference support," http://jena.apache.org/documentation/inference/.

[16] H. Kanan and M. Khanian, "Reduction of neural network training time using an adaptive fuzzy approach in real time applications," *International Journal of Information and Electronics Engineering*, vol. 2, no. 3, 2012.

[17] K. Keahey, M. Tsugawa, A. Matsunaga, and J. Fortes, "Sky computing," *Internet Computing, IEEE*, vol. 13, no. 5, pp. 43–51, 2009.

[18] G. Mateescu, W. Gentzsch, and C. J. Ribbens, "Hybrid computingwhere hpc meets grid and cloud computing," *Future Generation Computer Systems*, vol. 27, no. 5, pp. 440–453, 2011.

[19] D. Nguyen and B. Widrow, "Improving the learning speed of 2-layer neural networks by choosing initial values of the adaptive weights," in *Proceedings of the international joint conference on neural networks*, vol. 3. Washington, 1990, pp. 21–26.

[20] OpenMP, "OpenMP Specifications," http://www.openmp.org/mp-documents/OpenMP3.1.pdf.

[21] T. V. Pham, H. Jamjoom, K. Jordan, and Z.-Y. Shae, "A service composition framework for market-oriented high performance computing cloud," in *Proceedings of the 19th ACM International Symposium on High Performance Distributed Computing*. ACM, 2010, pp. 284–287.

[22] H. Schabauer, E. Schikuta, and T. Weishäupl, "Solving very large traveling salesman problems by som parallelization on cluster architectures," in *Sixth International Conference on Parallel and Distributed Computing, Applications and Technologies, PDCAT 2005*. IEEE, 2005, pp. 954–958.

[23] E. Schikuta, H. Wanek, and T. Fuerle, "Structural data parallel simulation of neural networks," *Journal of Systems Research and Information Science*, vol. 9, no. 1, pp. 149–172, 2000.

[24] E. Schikuta and C. Weidmann, "Data parallel simulation of self-organizing maps on hypercube architectures," *Proceedings of WSOM*, vol. 97, pp. 4–6, 1997.

[25] E. Schikuta and T. Weishäupl, "N2Grid: Neural Networks in the Grid," in *Neural Networks, 2004. Proceedings. 2004 IEEE International Joint Conference on*, vol. 2, July 2004, pp. 1409–1414.

[26] E. Schikuta, "Neuroweb: an internet-based neural network simulator," in *Tools with Artificial Intelligence, 2002.(ICTAI 2002). Proceedings. 14th IEEE International Conference on*. IEEE, 2002, pp. 407–412.

[27] E. Schikuta, F. Donno, H. Stockinger, H. Wanek, T. Weishäupl, E. Vinek, and C. Witzany, "Business in the grid: Project results," in *1st Austrian Grid Symposium*. Hagenberg, Austria: OCG, 2005.

[28] E. Schikuta and E. Mann, "N2sky - neural networks as services in the clouds," in *International Joint Conference on Neural Networks*. USA: IEEE, 2013. [Online]. Available: http://eprints.cs.univie.ac.at/3709/

[29] *A Neural Network Face Recognition Assignment*, Shufelt, Jeff, 1994. [Online]. Available: http://www.cs.cmu.edu/afs/cs.cmu.edu/user/avrim/www/ML94/face_homework.html

[30] E. Vinek, P. P. Beran, and E. Schikuta, "Classification and composition of qos attributes in distributed, heterogeneous systems," in *11th IEEE/ACM International Symposium on Cluster, Cloud, and Grid Computing (CCGrid 2011)*. Newport Beach, CA, USA: IEEE Computer Society Press, May 2011.

[31] W3C, "Sparql query language for rdf [online]," http://www.w3.org/TR/rdf-sparql-query/, 2008.

[32] T. Weishäupl and E. Schikuta, "Parallelization of cellular neural networks for image processing on cluster architectures," in *Parallel Processing Workshops, 2003. Proceedings. 2003 International Conference on*. IEEE, 2003, pp. 191–196.

[33] T. Weishäupl, F. Donno, E. Schikuta, H. Stockinger, and H. Wanek, "Business in the grid: BIG project," in *Grid Economics & Business Models (GECON 2005) of Global Grid Forum*, vol. 13. Seoul, Korea: GGF, 2005.

[34] T. Weishäupl and E. Schikuta, "Cellular neural network parallelization rules," in *IEEE CNNA 2004*. IEEE, July 2004. [Online]. Available: http://eprints.cs.univie.ac.at/868/

[35] T. Weishaupl and E. Schikuta, "How to parallelize cellular neural networks on cluster architectures," in *Parallel Architectures, Algorithms and Networks, 2004. Proceedings. 7th International Symposium on*. IEEE, 2004, pp. 439–444. [Online]. Available: http://www.computer.org/portal/web/csdl/doi/10.1109/ISPAN.2004.1300519

[36] T. Weishäupl and E. Schikuta, "Towards the merger of grid and economy," in *International Workshop on Agents and Autonomic Computing and Grid Enabled Virtual Organizations (AAC-GEVO04) at the 3rd International Conference on Grid and Cooperative Computing (GCC04)*, ser. Lecture Notes in Computer Science, vol. 3252/2004,

Springer. Wuhan, China: Springer Berlin / Heidelberg, 2004, pp. 563–570.

[37] T. Weishäupl, C. Witzany, and E. Schikuta, "gSET: trust management and secure accounting for business in the grid," in *6th IEEE International Symposium on Cluster Computing and the Grid (CCGrid'06)*. Singapore: IEEE Computer Society, 2006, p. 349–356.

[38] P. Wu, S.-C. Fang, and H. Nuttle, "Curved search algorithm for neural network learning," in *Neural Networks, 1999. IJCNN '99. International Joint Conference on*, vol. 3, 1999, pp. 1733 –1736 vol.3.

[39] XNBC:, "A software package to simulate biological neural networks for research and education ," http://www.b3e.jussieu.fr/xnbc/.