Novel Concepts for Realizing Neural Networks as Services in the Sky

Altaf Ahmad Huqqani\textsuperscript{1}, Erwin Mann\textsuperscript{2}, and Erich Schikuta\textsuperscript{3}

\textsuperscript{1} University of Vienna, Austria aahuqqani@gmail.com
\textsuperscript{2} University of Vienna, Austria erich.schikuta@univie.ac.at
\textsuperscript{3} University of Vienna, Austria erwin.mann@gmail.com

Abstract

A Virtual Organization (VO) is logical orchestration of globally dispersed resources to achieve common goals fostering new computing paradigms as Utility Computing, Grid Computing, Autonomic Computing, Clusters Computing and Cloud computing. The Computational Intelligence community is striving hard to build an online community to share resources such as data, paradigms, human expertise, models and methods. We build up on the N2Sky system, a Virtual Organization architecture for the Computational Intelligence community, which provides sharing and exchange of neural network resources, neural network specific knowledge, neural network objects and paradigms. In this paper we present novel extensions to N2Sky: automatic exploitation of parallelization capabilities of modern hardware (Multi-core servers, GPGPU) and softwares (OpenMP, CUDA), SLA management for Quality-of-Service guarantees of Cloud resources, and NN2Query, a novel query interface facilitating the user to find solutions to problems by smart searching of neural network resources using semantic web technology.

Keywords: Neural Networks, Sky Computing, Virtual Organization, Computational Intelligence, GPGPU

1 Introduction

A Virtual Organization (VO) an informal enterprise enabled by the ICT technology where peoples and organizations with their own expertise bring their geographical dispersed resources together for a common goal. The participating entities produce a problem solving environment by sharing information and resources, problem solving methodologies, business process and data storages etc. In scientific literature many related terms such as collaborations \cite{2} \cite{29}, e-Science or e-Research \cite{2} \cite{7}, distributed workgroups or virtual teams \cite{2} \cite{14}, virtual environments and online communities \cite{15} \cite{2} have become popular in this context. VOs typically provide shared realtime access to centralized/decentralized or distributed resources, community tools, applications, experimental data, sensors and experimental operations.
The scope of VO can not be restricted as it serves to solve problems in different domains. VOs have played several advantageous roles such as that of the facilitator to access research (BIRN in biomedical, LEAD mesoscale metrology, NANOHub in nanotechnology), enabler of system level science (SCEC for earth engineering, caBig cancer bioinformation, Large Hadron Collider for high energy physics and ESGrid for climate research), enhancer of Problem Solving Processes (TeraGrid) and key to Competitiveness (GEON [13]). These VOs and others are addressing problems that are too large and complex for any individual or institution to tackle alone. As the matter of fact it is not even feasible to assemble at a single location all of the expertise required to design a modern accelerator, understands cancer, or predict the likelihood of future earthquakes. In daily life we come across VO as social networking, e.g. Facebook, Twitter or MySpace, and in near future it is supposed that every person on the earth will be a part of some VO serving its purpose in the organization.

In the past we [12] developed a reference architecture to build virtual organizations and developed N2Sky [20] on the basis of this architecture. Now, in this paper we present enhancements to N2Sky by incorporating Service-Level-Agreement (SLA) management, providing NN2Query and exploiting parallelization capabilities of new hardware and software technologies. Thus, we provide hardware and ANN based knowledge resources to user transparently to play around, test and solve problems without buying/worrying expensive hardware and system configuration. The system follows the sky computing [11] paradigm; a federated cloud computing concept.

The layout of this paper is as follows: Section 2 describes the building architecture of N2Sky while section 3 presents a specific deployment of the system. The section 4 describes the parallel approaches for neural network and section 5 presents smart querying of neural network resources by ontology alignment. To maintain a smooth functioning of federated clouds and QoS we have described SLA management modules in section 6 while section 7 presents some simulation results. Finally the paper concludes in section 8 and gives direction to our future work.

2 N2Sky Cloud Development

Grid Computing [3] is considered ideal for building a VO due to its layered architecture. Existing grid environments are data grids or computational grids but for a VO to solve problems both data and computational resources are essentials. However due to scalability, lack of business models and strict trust model it is difficult for the user and participating entities to enter or leave a grid environment: so making grid computing hard to build a VO. However cloud computing can provide such basis, where every resource is virtualized, ranging from hardware/software, infrastructure, platform, applications and even human beings can act as a service implementing the XaaS (Everything-as-a-Service) vision. Our research endeavor has focused on developing a Reference Architecture for Virtual Organizations (RAVO) [12], which allowed us to develop N2Sky [20].

Building up on these achievements, we aim at novel extensions of N2Sky. Figure 1 presents an extended architecture catering QoS by SLA and HPC by enabling OpenMP on multicore servers and CUDA on GPGPU. Highlighted components are mandatory while the others are optional. Each of the five layers depicted in the figure provide functionality in SOA manner; so we can say that N2Sky realizes the XaaS.

The Cloud computing paradigm was taken into consideration while developing and deploying the N2Sky system because this technology enables ubiquitous, convenient, on-demand network access to shared pool of configurable computing resources like networks, servers, storage, application and services that can rapidly be provisioned and supplied to end user with minimum
management effort or service provider interaction [13]. We have taken into consideration the five Cloud characteristics [13], which are:

- **Shared Pool of Resources**: N2Sky shares hardware resources as well as knowledge resources to allow the creation of a shared pool of neural net paradigms, neural net objects and other data and information between researchers, developers and end users worldwide.

- **On-demand self-service**: N2Sky provides access to consumer whenever he needs independent of his location from a desktop or a smart phone transparently to “high-end” resources (computing and knowledge).

- **Broad network access**: N2Sky fosters location independence of computational, storage and network resources to end users from anywhere they need it as long as they are connected to the network.

- **Rapid elasticity**: N2Sky delivers to the users a resource infrastructure by virtualization technology which scales up/down according to the problem. This leads to the situation that always the necessary resources are available for any neural network problem.

- **Measured service**: N2Sky supports the creation of neural network business models, access to neural network resources, as novel paradigms or trained neural networks for specific problem solutions, can be free or following the certain business regulations, e.g. a certain fee for usage or access only for specific users/groups.

3 **N2Sky Cloud Deployment**

We deployed the N2Sky prototype on Eucalyptus (http://www.eucalyptus.com). 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 physical resources. The N2Sky design approach allows easy portability to other cloud computing platforms.

The N2Sky is Java-based simulation of neural networks by using Apache Axis Library and Apache Tomcat Web container as hosting environment for the web services with Java Servlets/JSPs based web fronted to access these services. It can be deployed as a federated clouds model by fostering the specific affinities (capabilities) [9] 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 implementations to execute tasks on the Computational Cloud, which provides the specific hardware resources, as Multicore Servers, GPUs etc. and software environments, as OpenMP etc.

4 Parallelization Approaches for ANN Task Execution

We envision a smart system, which administrates parallelized execution patterns for neural network simulation 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. We focused on different neural network paradigms, as Backpropagation[17], Kohonen[16], and cellular neural networks [26], and various parallel infrastructures, as hyper-cube [18], cluster [24], GPU and multicore systems [8] and developed set of rules [26, 24] for the simplified development of parallel execution scenarios, which are even applicable for the arbitrary user also, to be done automatically and transparently by the simulation system. As a proof-of-concept figure 3 presents the 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.

In a first attempt [8] we concentrate on two parallelization approaches motivated by our problem domain and the available hardware resources: structural data parallelization techniques for multicore systems using OpenMP and topological data parallelism on GPU using CUDA. The core of our envisioned system is a knowledge-base which consists of set of rules to select 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 search and management of the knowledge-base we apply SPARQL [23] and Jena [10]. 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.
Figure 3: Selection of openMP or GPU

5 NN2Query: a “Neural Network Google” Engine

The vision of N2Sky foresees a huge number of neural network objects stored all over the Internet which deliver problem solution capabilities at will of the members of the N2Sky virtual organisation. Neural network objects are on the one hand generic neural network paradigms which can be instantiated by learning mechanisms solving a specific problem on a given training data set and on the other hand already trained neural networks for given application problems. The number of these neural network objects is expected to be very large and continuously growing. Another characteristic is that these neural network objects are distributed on a world wide scale on the Internet administratively under the umbrella of the N2Sky virtual organization on participating resource nodes.

The goal of this query interface, NN2Query, is to allow the user to specify his/her query in form of a natural language description of the problem statement and delivering a list of ranked N2Sky resource-URIs, which provide solutions for these problems.

A typical approach followed by query engines, as Google etc., is to scan all contributing nodes by crawlers. However, this approach to search our neural network objects is due to the computational effort out of scope of N2Sky. Therefore we are following a centralized registry approach collecting all semantic knowledge of neural network objects by semantic web technologies as ontologies.

5.1 Ontology Alignment

Network objects administered by N2Sky are described by ViNNSL, the Vienna Neural Network Specification Language [1]. ViNNSL is an XML based language for describing, training and running neural network objects in a distributed infrastructure, as grids and clouds. ViNNSL allows dynamic resource-to-resource communication regarding the semantics of neural network resources, which supports both structural information and semantic information, as describing the usage of a network objects for given problems by a natural language approach.
In the effort to find an existing neural network paradigm or already trained object we applied the technique of ontology alignment. An ontology defines a set of representational primitives with which to model a domain of knowledge. The representational primitives are typically classes, attributes or relationships. These contain information about their meaning and constraints on their logically consistent application. So, an ontology represents the knowledge of a specific domain. To combine knowledge of different domains leads to ontology alignment, where one ontology is mapped to another. Hereby three ontology combination paradigms can be distinguished, ontology linking, ontology mapping, and ontology importing.

5.2 Neural Network Resources Knowledge Base

For our problem we apply ontology linking, where individuals from distinct ontologies are coupled with links.

The concept is as follows: We administer basically two ontologies, a problem ontology and a solution ontology:

- The problem ontology consists of a hierarchical organisation of typical neural network application problems, as classification, optimization, approximation, storage, pattern restoration, cluster analysis, feature extraction etc. In the ontology hierarchy these main domains are finer distinguished till the single problem specifications in the leave nodes.

- The solution ontology stores all known N2Sky neural network objects organized according to their paradigm, as perceptron, multi-layer backpropagation, self-organizing maps (Kohonen cards), recurrent networks (Elman, Jordan, etc.), cellular neural networks, etc. The idea is that paradigm families are appropriate for specific solution mechanisms. Here the ontology delivers a fine grained structure finally giving the neural network objects (trained neural networks for a specific problem) as leaves.

Now, we define a mapping of problem ontology nodes, describing a specific problem, to solution ontology nodes, denoting network objects which deliver a solution for this problem. Links can be defined not only between leaves of the hierarchies but also between internal nodes. So a node specifying a more general problem in the problem ontology can link to a subtree in the solution hierarchy identifying a set of similar neural network objects delivering solutions for the given problem. Figure 4 sketches this situation for finding a neural network object for the well known “travelling salesman problem”. Following the link from the problem specification in the problem ontology delivers the URI of an available Kohonen neural network object for execution in the solution ontology [16]. If the pattern “travelling salesman” would have not showed up in the problem ontology at least the higher level link from “optimization ⇒ minimum” would result the set of all Kohonen networks as possible solution candidates.

This mapping between problems to solutions can be done on the one hand by N2Sky administrators manually, and on the other hand by an automatic mapping during insertion of a new network objects based on the ViNNSL semantic information of this object.

5.3 Query Mechanism

The search algorithm is as follows: Based on the natural language keywords of the user query a scan over the problem ontology is performed. Hits, patterns matching the scan, resembled by nodes in the hierarchy, are collected and the links to the solution ontology are followed. There, a scan of the network objects, representing solutions to the problems, is done and fitting results
are reported to the user. The sequence of the results can be guided by a fitting rank of problem to solution matches.

By this approach the effort for delivering matching neural network resources is centralized in the management service. The number of network resources to be checked is pruned dramatically by only checking (solution) resources which are (obviously) targeting the problem domain.

By this approach we aim for delivering a domain specific query interface to the user resembling a “Neural Network Google”.

6 SLA management for N2Sky

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 [5, 6, 4] into N2Sky to allow for novel business models [28, 19, 27, 25] on the selection and usage of neural network resources based on quality of service attributes [22].

Service level agreements SLA are the key component in enabling the QoS and smooth running of the VO, as it imposes certain obligation both on service provider and the end user. The Figure 5 shows the main building blocks of our SLA management system. The “Business Process Administrator” holds the business logics and controls “SLA Management”.

![SLA management building blocks and their relationship](image_url)

Figure 5: SLA management building blocks and their relationship
The main responsibilities of the Business Process Administrator include publishing and
discovery of the NN services using WSDL and the maintenance of WS-agreement based SLA
templates before the establishment of the binding SLAs. It also interprets the user requirements
to discover the required NN services and offers them to the user through the dynamically
generated SLA templates composed in the light of business rules and organizational policies.

The SLA Management can initiated and manage SLA negotiation based on the SLA tem-
plates generated by the “Business Process Administration”. It has two sub-modules namely
Computational Management module and Resource Management module. The role of the com-
putational management module is to manage computational infrastructure for NN service pro-
vision. It composes and schedules service according to committed SLA, instantiate the service
and co-ordinates with Resource Management Module to deploy on the physical resources and
keeps track of running service instances.

The Resource Management is a key component whose basic role is to maintain resources
in accordance with the promised level of service. In this regard it replicates/starts a new
instance of the service requested, provides used and idle resource information, and keeps track
of execution of submitted tasks. The Simulation Management services provide status/results
of the simulations/task executed.

7 N2Sky Simulation Setup & Results

Test Data: 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 [21]
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, 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.

Hardware and Software: For the two multithreaded versions of the BPNN face recog-
nition 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).
The table 1 shows the execution time for 20 epochs for different no of neurons as input while
varying the hidden neurons.

8 Conclusion & Future Work

We have outlined the architecture of the novel extended N2Sky system, which ensures QoS by
SLAs and provided resources to end user transparently over the network. In addition to this, the
proposed system provides a Virtual Organization bridging the gap between high performance
Neurons at input layer 15360

<table>
<thead>
<tr>
<th>Hidden Neuron</th>
<th>8</th>
<th>16</th>
<th>32</th>
<th>64</th>
<th>128</th>
<th>256</th>
<th>512</th>
<th>1024</th>
</tr>
</thead>
<tbody>
<tr>
<td>CPU (Sec)</td>
<td>2.92</td>
<td>6.04</td>
<td>19.65</td>
<td>45.46</td>
<td>99.44</td>
<td>194.64</td>
<td>480.91</td>
<td>971.06</td>
</tr>
<tr>
<td>OpenMP (Sec)</td>
<td>2.40</td>
<td>4.28</td>
<td>9.68</td>
<td>15.74</td>
<td>29.95</td>
<td>58.82</td>
<td>122.60</td>
<td>248.28</td>
</tr>
<tr>
<td>GPU (Sec)</td>
<td>11.34</td>
<td>11.59</td>
<td>12.47</td>
<td>13.98</td>
<td>16.00</td>
<td>24.23</td>
<td>38.00</td>
<td>53.60</td>
</tr>
</tbody>
</table>

Neurons at input layer 960 inputs

<p>| | | | | | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>CPU (Sec)</td>
<td>0.93</td>
<td>1.84</td>
<td>3.74</td>
<td>7.39</td>
<td>15.29</td>
<td>36.07</td>
<td>101.23</td>
<td>289.62</td>
</tr>
<tr>
<td>OpenMP (Sec)</td>
<td>0.42</td>
<td>0.63</td>
<td>1.20</td>
<td>1.44</td>
<td>2.03</td>
<td>2.81</td>
<td>5.36</td>
<td>12.79</td>
</tr>
<tr>
<td>GPU (Sec)</td>
<td>4.30</td>
<td>4.47</td>
<td>5.01</td>
<td>6.07</td>
<td>8.15</td>
<td>12.65</td>
<td>21.88</td>
<td>38.04</td>
</tr>
</tbody>
</table>

Table 1: Execution time (sec) for CPU, OpenMP and GPU with varying hidden neurons

computing by utilization of available parallel hardware. Finally the software interface provides a new query environment to find neural network solutions for given problem by mapping problem ontologies to solution ontologies.

We are working on integrating these new functionalities into the production N2Sky system, which is deployed, besides our development system on the private Eucalyptus cloud, on the Amazon EC2 as public demonstrator.

References


