A Generalized Scalable Software Architecture for Analyzing Temporally Structured Big Data in the Cloud

Magnus Westerlund, Ulf Hedlund, Göran Pulkkis, and Kaj-Mikael Björk

Arcada University of Applied Sciences, Jan-Magnus Janssons plats 1,
00550 Helsinki, Finland
{magnus.westerlund,ulf.hedlund,goran.pulkkis,
kaj-mikael.bjork}@arcada.fi

Abstract. Software architectures that allow researchers to explore advanced modeling by scaling horizontally in the cloud can lead to new insights and improved accuracy of modeling results. We propose a generalized highly scalable information system architecture that researchers can employ in predictive analytics research for working with both historical data and real-time temporally structured big data. The proposed architecture is fully automated and uses the same analytical software for both training and live predictions.

Keywords: predictive analytics, temporal data, system-level design, self-adaptive systems, runtime models.

1 Introduction

Development of big data analytics enabled solutions are being driven by unstructured data, surprisingly little is done in open sourced development of processing structured big data. Structured data is often considered, mischievously so, as data at rest perhaps due to its usual implementation as a state storage. In the world of big data analytics, it is not enough to simply analyze historical or at rest data, instead this must be done in a (near) real-time environment. The term real-time environment means something slightly different depending on the context, here we refer to an event being triggered when new data exists and how the system consequently handles that data automatically. We concur with the argument that a big data model's excellence can only be shown after it has been employed in an online environment [1][2]. In [3] it is claimed that "Advancing the cloud from a computation and data management infrastructure to a pervasive and scalable data analytics platform requires new models, tools, and technologies that support the implementation of dynamic data analysis algorithms." The process of integrating disparate data sources (external and/or internal origin), continuously preprocessing data for validity or other purposes, and transporting data to processing nodes is such a delicate and often limiting step in the workflow that a model's performance can only be known after this step has been completed. Quite few predictive models exist that have an ability for online learning; this adds a requirement for models employed in a live environment to be re-trained if/once their effectiveness diminishes. This self or auto tuning is an integral part of any real-time automated analytical system.

We propose a generalized scalable software architecture based on predictive analytics for working with structured (near) real-time and historical data. The structured data we refer to can originate from a multitude of sources like sensor data, machine or user interface, financial data or any other source of online data with a temporal structure, commonly referred to in programming terms as data tuples. The proposed architecture will be able to take advantage of resources from both a public cloud infrastructure and private cloud providers [4]. We set a requirement that data processing activities should be fully automated once the system user has initiated model training. The architecture should automatically handle scaling on available resources, perform feature extraction, carry out model training, initiate auto tuning, and do live prediction.

We focus on the information system architecture and make the assumption that global memory is not usually required for analytical models that consume temporally structured big data. The contribution we bring is an understanding of a generalized light-weight scalable architecture for designing actual analytical information systems that can be used in collaboration with modern decision support systems. Our architecture allows the researcher to take almost any off-the-self model that can be encapsulated as an executable binary and deploy it as a scalable predictive analytics information system. By making extensive use of cloud computing in order to be able to scale sufficiently, we can employ an ensemble of ensembles of different machine learning models to achieve a good prediction rate [5]. We primarily narrow the type of forecasting models we use to those using supervised learning, e.g. Support Vector Machines [6] or Neural Networks [7]. This does not however exclude the use of unsupervised or reinforcement based learning methods provided a model and training error calculation can be automatized.

2 Related Research

Analytical Information Systems have recently received a great deal of attention from academics, open source community, as well as industry [8][9][10][11][12]. Also standardization steps in big data analytics have been taken [13]. The driving force behind related software design has been the Service Oriented Architecture (SOA) [14] approach, which has become an industry de-facto standard for building cloud based, loosely-coupled "X as a Service" enabled data sharing software modules.

The open source community tools for processing and storing big data [15], e.g. technologies such as Hadoop [16] and PIG [17], are mainly focusing on unstructured data and on extending the MapReduce programming model [18]. The MapReduce programming model assists the developer in segmenting data into smaller pieces and process them on the node were data resides, instead of moving data to a second node for processing. This consequently reduces processing time for large data sets.

In the case of handling structured big data, as temporal data often is, more traditional methods are still often applied, such as queuing methods and clustered relational databases, with data then moved to separate nodes for processing. The field of structured data analytics is dominated by proprietary software companies such as IBM [19] and SAP (e.g. SAP HANA [20]). From the perspective of start-ups, small and medium enterprises and academics the cost structure for such solutions is a significant barrier. Recommendations in [3] for future cloud-based data analytics research include development of “scalable higher-level models and tools” and the use of the Web and cloud services for “worldwide integration of multiple data analytics frameworks”.

3 Defining a Scalable Analytics Architecture

Scalable architectures usually refers to an ability to add and employ either more processing power in a parallel fashion or adding more storage capacity. In this paper we mainly refer to adding more processing power by adding more computing instances, i.e. scaling horizontally. Scalable architectures have been used by academics (in particular) in natural sciences for a long time, predominantly using data at rest as input data. For example weather forecasting is often based on a time interval update frequency instead of being data driven, hence considered being a batch driven architecture requiring manual operation or a scheduler [21].

Scalable architectures exist for both grid computing and for cloud computing [22][23], here we focus on cloud computing. A cloud computing architecture is built upon several layers. On the top there is an application layer containing the user defined software, and at the bottom there is a fabric layer providing access to compute resources, storage resources, and network resources. Under the application layer there is a unified resource layer containing “resources that have been abstracted/encapsulated (usually by virtualization) so that they can be exposed to upper layer and end users as integrated resources” and a platform layer adding a collection of specialized tools, middleware and services on top of the virtual resources that provide a development and/or deployment platform [24]. Our architecture integrates with Amazons platform layer often referred to as Amazons Platform as a Service (PaaS) [25].

From an information system point of view the architectures required for handling event driven systems and batch driven systems are quite dissimilar. When a system is event driven it is assumed that computations are only and always initiated when new events, e.g. data, exist. However, when developing analytical systems the definition is not always so clear cut. Consider e.g. predicting a high volume financial instrument, here the model input update frequency can be measured in nanoseconds. Thus, preformatting data into more manageable update cycles is often a requirement as model throughput (i.e. forecasts) is usually measured in times order of magnitudes greater. There are different techniques for adapting the update cycle; often the simplest form is that data points are combined for a certain time resolution, e.g. 0.1 seconds and only updated values are sent forward. This case requires the real-time processing part of the system to be a data listener and allows for an event based programming model. However, the model training part is still driven by a type of command collection, due to that most forecasting models are not trained in an online manner. The training process itself is pre-determined by the system user when defining the model setup. As the requirement on a typical analytical system is to be able to handle both data at rest as well as real-time data, we defined two separate work flow processes: model training workflow and live prediction workflow.

3.1 Model Training Workflow Using Data at Rest

In the following sub-chapter we discuss a workflow for how machine learning models learn a behavior without manual assistance, based on historical data. Our training workflow includes four main phases, namely model creation, data feature extraction, model learning and model verification. In Figure 1 we show the model training process for using historical data. Data as represented in Figure 1 (accessed from the Data Gateway) has been pre-processed for validity and other preformatting techniques such as time boxing and adjusted for required granularity, i.e. time resolution. The data feature extraction block refers to techniques such as component analysis, technical analysis or normalization as required by the system user. While data is being prepared, an asynchronous flow initializes contact to the worker nodes in the public or private cloud (hereafter referred to as cloud when no differentiation is needed). These worker instances will be started manually with a pre-prepared image containing required software (e.g. correct software versions, security and our base client service that starts automatically with the instance), in order for the system user to have full control over all simultaneously running instances.

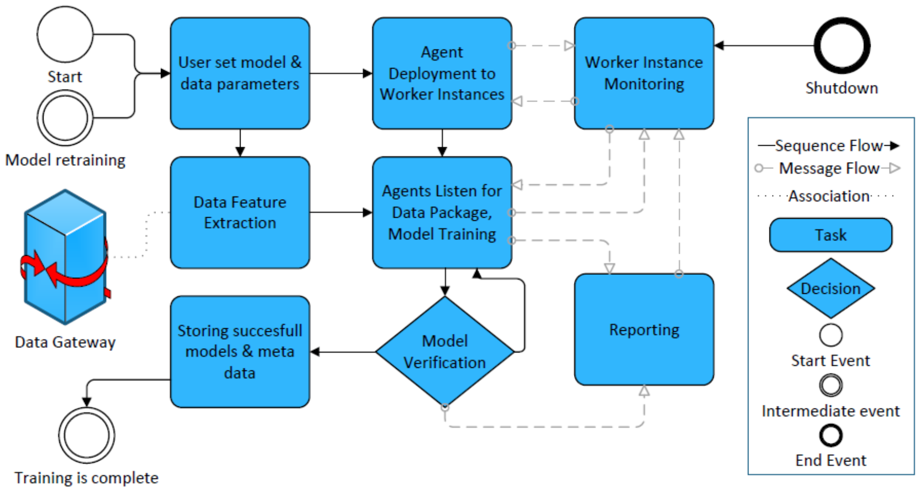

Fig. 1. Model training workflow using data at rest

We have defined two identification means for initiating contact to worker instances depending on if they are in the cloud subnet of the model or not. In the former case the new instances are detected when they come into existence and consequently initiate handshaking. Once the initial handshake is performed necessary binaries defined by the developer and used by worker instances to process data are transferred to the worker nodes. We refer to these binaries henceforth as agents. In the latter case, when worker instances are outside the cloud subnet of the model, they contact the model through a predetermined address. The agents automatically subscribe to the worker instance monitoring service in order to receive jobs.

Here we define one type of agent, but this construct allows us the possibility to use different types of agents. Assume that the underlying worker instances are different, one instance having a powerful Graphical Processing Unit (GPU) and other don't, then we can assign different binaries to each, optimized for the specific hardware. Once an agent receives its data, the process of training a model commences. The primary reason to differentiate between model, agent and worker instance is that the worker instance represents the virtualized cloud instance (including our base client service), the model is the serialized network representation, while the agent is the generic binary with ability to report status, listen to data events and execute models as defined in the data event. A data event refers to a new or updated input variable, however, the actual data packages sent to the worker instances vary depending on type of job. In the case of a training job the data package includes all the input data a model needs in a fully preprocessed format as well as model metadata. The input data for training can be of considerable size and is therefore transferred in a compressed format.

Given that the training process is fully automated we make use of both a verification data set and an evaluation out-of-sample training data set for calculating model error rate. The verification data set is usually a pre-training data sample. We continue training till we can confirm that a certain training iteration delivered the lowest in- and out-of-sample training error, in order to minimize the risk of overfitting. We perform a comparison of the verification and training error to determine the convergence point, when they combined reached their minimum. Once the model training is considered to have reached its minimum training error we perform a second out-of-sample test of the best iterations found that we refer to as evaluation, using data from post in-sample data to determine the models most recent performance and ultimately which iteration to choose. See Figure 2 for a graphical representation of data segmenting. The evaluation is also performed continuously and as a part of the second workflow process to determine the model's continued health. The trained models are then serialized along with their respective metadata. Status messages (identified by dotted lines in Figure 1) are also continuously sent from certain process blocks to a reporting user interface.

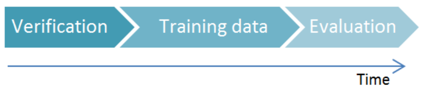

Fig. 2. Data segmenting

3.2 Live Prediction Workflow Using Real-Time Data

This chapter describes the differences in processing based on real-time data compared to historical data. Figure 3 shows the workflow for live prediction using real-time data. Once the system user has chosen a certain model set (ensemble) to be deployed and agents are available, the jobs are distributed randomly among the spare agents. To achieve this we implement a queue of jobs (i.e. new data events) that are sent along to agents that indicate they are free. In case the queue contains jobs that are outdated, which implies we have a performance bottleneck, the job gets canceled automatically.

Jobs become outdated once the point in time they are supposed to be forecasting to has passed. A data event during live prediction includes, in addition to the input data needed for prediction, network weights and metadata required for processing.

When a job has been processed the result is collected centrally for both calculating a new ensemble vote as well as to determine if a certain model has been performing badly and needs to be retrained. For automation purposes (i.e. not requiring human intervention during training, live prediction and model retraining) we consider using an ensemble of models as a requirement. This technique is also known as voting by committee. Considering the reported success of deep learning and convolutional networks [26] we employ an analogous type of network specialization, although not hierarchical, called ensemble of ensembles. This machine learning technique enables developers to use both different models as well as feature inputs, and then combine them through a voting mechanism. This network specialization can be used in order to enhance weak signals or characteristics in data that otherwise might disappear as noise or in significance to other more prominent data features. There are several techniques for calculating an ensemble vote, we chose to use out-of-sample evaluation data for determining the relevance of each model. This allows us to evaluate each model's performance in regards to its recent forecast and based on the evaluation error rate determine its voting rights.

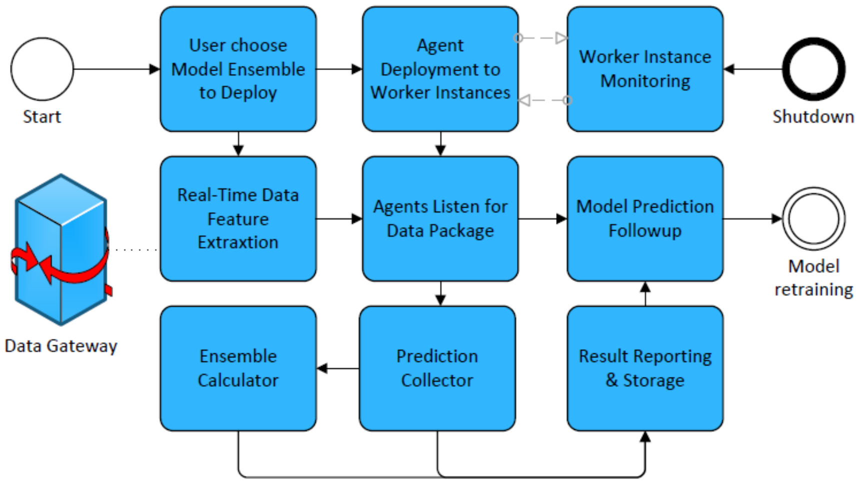
Start
User choose Model Ensemble to Deploy
Agent Deployment to Worker Instances
Worker Instance Monitoring
Shutdown
Data Gateway
Real-Time Data Feature Extraxtion
Agents Listen for Data Package
Model Prediction Followup
Model retraining
Ensemble Calculator
Prediction Collector
Result Reporting & Storage

Fig. 3. Live prediction workflow

Our architecture's ability to continuously update and improve itself is based on the decision of retraining a certain individual model as a result of its recent performance. We use a number of previous predictions and define a threshold for how large the error rate is allowed to be. E.g. if >50 % of the last 1000 predictions are faulty we initialize the training of a new model based on the same input variables and model parameters, but with the most recent data. For a theoretical background of employing ensembles and the various setups and combination techniques used, see [27].

4 Proposed Architecture

Based on the two workflow models we can derive three different components that will be central for the system. First we identify a client component which allows the system user control over model training and execution as well as management of cloud instances. The client can be run locally outside the cloud and takes the form of a graphical user interface. The second component we identify is a server component. The server component acts as a central node responsible for interfacing with the client component, extracting features from data sources, maintaining efficient distribution of new jobs and collecting and processing of modeling forecast results as defined. The third identified component is the agent, responsible for performing various jobs sent to it by the server component.

4.1 Component Identification

Based on the analysis we can define a component diagram, identifying the sub-components and interfaces for the server component, see Figure 4. Our goal is to create a unified design for both the training process and live predictions and through a polymorphic interface switch between them. One of the challenges faced was to solve how feature extraction is handled in both cases, here defined as Data Generator. More on this presented in next subchapter. The Job Engine component handles the job queue and allows for two types of jobs to be entered into the queue. It also maintains a current list of available agents. When an agent disappears without notification, it will be detected once the monitoring service stops receiving updates from the agent. The agents send status messages continuously that are used to determine the health of each agent. When a disappearance of an agent is detected the job queue is notified. In case of an unreported job that had been associated with the disappeared agent is discovered, the job is resent to another agent.

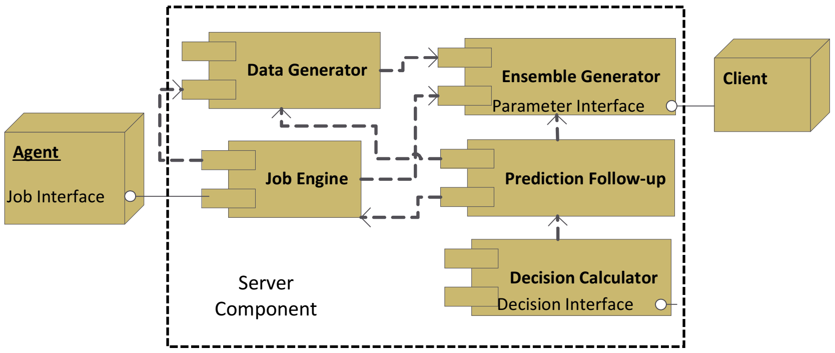

Fig. 4. System component diagram

4.2 Data Processing

As can be seen from the component diagram the server component defines the network topology as a star network and not a fully distributed network. The reasoning for creating a star topology is that network training is such a computationally dominating task that from a parallelization perspective we assume there is not much gain by creating a fully distributed network. It's important to remember that when working with structural big data the data volume compared to unstructured big data is much smaller. Historical temporally structured big data is usually confined to sizes less than 10GB. Most current forecasting models struggle in training when presented with too much data, so researchers often want to reduce this even further.

When exceeding a data volume size of 10GB feature extraction becomes a more time consuming task that might be sped up by distributing the processing to agents. As we are using ensembles it means we can typically have about 20-30 models of the exact same setup sharing the same input data. Therefore feature extraction activity should be connected to the models and only performed once. In a fully distributed network were all activities such as model training, live forecasting and feature extraction are distributed to worker instances we recommend using an indexer with a lookup service for connecting the different types of tasks.

Performing feature extraction on both historical and real-time data with the same underlying algorithms requires that data is always formatted using predetermined data tuples that include a required amount of data elements and using the same time resolution. For certain algorithms, e.g. normalization algorithms, previous scaling values used for historical data must be employed during live prediction as well. These types of values must therefore be stored as model metadata during training as they are used as well during live prediction.

4.3 Risk Assessment and System Security

Defining and implementing a secure system architecture is always important and challenging. By prioritizing the type of security risks one wants to avoid, it is possible to define the security objectives of the system [28]. The risks identified with highest priority were unauthorized access to the system, exposure to physical attacks and malicious resource consumption. Therefore our main security objective is to avoid financial exposure of an attacker being able to highjack our account resources to run their own services. As a consequence we employ the following security controls:

1. Each system user is required to use a personalized cloud account.
2. System passwords required for instance control are only entered during runtime (through a secured remote desktop connection).
3. Communication between instances is always encrypted over HTTPS/TLS.
4. Each instance must use a certificate that is unique for each system user.

We point out that this type of risk assessment must always be performed on an individual basis. It should also be noted that if the system is to be handling sensitive data then protecting that data from possible exposure should be a prioritized risk.

5 Conclusion

Our main research target is to create a general architecture for predictive analytic modeling. The architecture should be flexible both regarding underlying infrastructure as well as the software used for data manipulation and modeling. We consider we have achieved this by implementing support for both Amazon's [25] PaaS solution (supporting both regular and spot instances) as well as our own private cloud. The architecture allows us to extend support to any cloud provider that offers a basic API for launching and stopping multiple instances from the same image. We can extend our modeling software to use other machine learning libraries by adding a new wrapper class. This is possible since we make extensive use of internal interfaces between the different software layers and modules. Currently we make use of the Encog machine learning framework [29] which scales well on a single node and exists in versions for both CPU processing and a limited beta for GPU processing. We implemented support for a worker instance to run simultaneous agents sharing its processing resources. When having access to powerful instances this becomes a useful feature. Consider when the instance has more than one GPU that the underlying analytical software cannot take advantage of in a co-operative mode, then two agents can co-exist instead. Another similar case is when the machine has both powerful CPU and GPU processing, then a desired number of agents can be run on the same instance.

The contribution of the proposed architecture is in unifying the automatized processing of both historical and real-time data for predictive purposes, by utilizing a cost-effective cloud computing solution. Working with both historical and real-time data allows the researcher to not only determine the statistical significance of the forecast on historical data, e.g. when forecasting financial time series the result is often compared to the random walk. Forecasting both on historical and real-time data allows for a confirmation that correlation in the model error occurs in a separate setting. This reduces the significance of the critique that results are not repeatable when forecasting dynamic time series. Albeit, this critique can never be fully met as we are dealing with future information, still this type of research has the potential to be more trustworthy by industry than before. Perhaps the most prominent promise this holds for the science community is that researchers can with greater certainty verify that the results do not contain common errors that occur by including future data as input to the models.

References

1. Demirkan, H., Delen, D.: Leveraging the capabilities of service-oriented decision support systems: Putting analytics and big data in cloud. Decision Support Systems 55(1), 412–421 (2013)
2. Chen, Q., Hsu, M., Zeller, H.: Experience in continuous analytics as a service (CaaaS). In: Proc. EDBT 2011 (March 2011)
3. Talia, D.: Clouds for Scalable Big Data Analytics. Computer 46(5), 98–101 (2013)
4. Mell, P., Grance, T.: The NIST Definition of Cloud Computing. NIST Special Publication 800-145 (September 2011), `http://csrc.nist.gov/publications/nistpubs/800-145/SP800-145.pdf` (last accessed on August 8, 2013)

5. Timmermann, A.: Forecast Combinations. CEPR Discussion Papers 5361 (2005)
6. SVM - Support Vector Machines, `http://www.support-vector-machines.org/` (last accessed on August 24, 2013)
7. Haykin, S.O.: Neural Networks and Learning Machines, 3rd edn. Pearson Prentice Hall, USA (2009)
8. Konstantinou, I., Angelou, E., Boumpouka, C., Tsoumakos, D., Koziris, N.: On the Elasticity of NoSQL Databases over Cloud Management Platforms. In: Proceedings of the 20th ACM International Conference on Information and Knowledge Management, pp. 2385–2388. ACM (October 2011)
9. Begoli, E.: A Short Survey on the State of the Art in Architectures and Platforms for Large Scale Data Analysis and Knowledge Discovery from Data. In: Proceedings of the WICSA/ECSA 2012 Companion Volume, pp. 177–183. ACM (August 2012)
10. Fox, G.C.: Large Scale Data Analytics on Clouds. In: Proceedings of the Fourth International Workshop on Cloud Data Management, pp. 21–23. ACM (October 2012)
11. Valvåg, S.V., Johansen, D., Kvalnes, Å.: Position Paper: Elastic Processing and Storage at the Edge of the Cloud. In: Proceedings of the 2013 International Workshop on Hot Topics in Cloud Services, pp. 43–49. ACM (April 2013)
12. Rupprecht, L.: Exploiting In-network Processing for Big Data Management. In: Proceedings of the 2013 Sigmod/PODS Ph.D. Symposium on PhD Symposium, pp. 1–5. ACM (June 2013)
13. Ghazal, A., Rabl, T., Hu, M., Raab, F., Poess, M., Crolotte, A., Jacobsen, H.-A.: BigBench: Towards an Industry Standard Benchmark for Big Data Analytics. In: Proceedings of the 2013 ACM SIGMOD International Conference on Management of Data, pp. 1197–1208. ACM (June 2013)
14. SOA Manifesto, `http://www.soa-manifesto.org/` (last accessed on August 24, 2013)
15. Laney, D.: 3D Data Management: Controlling Data Volume, Velocity and Variety. Meta Group (Gartner) (February 2001)
16. Welcome to Apache^{TM} Hadoop®!, `http://hadoop.apache.org/` (last accessed on August 21, 2013)
17. Welcome to Apache Pig!, `http://pig.apache.org/` (last accessed on August 21, 2013)
18. Dean, J., Ghemawat, S.: MapReduce: Simplified Data Processing on Large Clusters. In: Sixth Symp. Operating System Design and Implementation (OSDI 2004), San Francisco, CA (December 2004)
19. Lustig, I., Dietrich, B., Johnson, C., Dziekan, C.: The Analytics Journey. An IBM view of the structured data analysis landscape: descriptive, predictive and prescriptive analytics. Analytics, 11–18 (November/December 2010), `http://www.analytics-magazine.org/november-december-2010/54-the-analytics-journey.html` (last accessed on January 17, 2014)
20. Lee, J., et al.: SAP HANA distributed in-memory database system: Transaction, session, and metadata management. In: IEEE 29th Int. Conf. Data Engineering (ICDE), pp. 1165–1173 (April 2013)
21. Plale, B., et al.: CASA and LEAD: Adaptive Cyberinfrastructure for Real-Time Multiscale Weather Forecasting. Computer 39(11), 56–64 (2006)
22. Sadashiv, N., Kumar, S.M.D.: Cluster, Grid and Cloud Computing: A Detailed Comparison. In: Proc. 6th Int. Conf. Computer Science & Education (ICCSE 2011), pp. 477–482 (2011)

23. Yuxi, L., Jianhua, W.: Research on Comparison of Cloud Computing and Grid Computing. Research J. Applied Sciences, Engineering and Technology 4(2), 120–122 (2012)
24. Foster, I., Zhao, Y., Raicu, I., Lu, S.: Cloud Computing and Grid Computing 360-Degree Compared. In: Proc. Grid Computing Environments Workshop (GCE 2008) (2008)
25. Amazon Web Services, `http://aws.amazon.com/` (last accesed on August 24, 2013)
26. Arel, I., Rose, D.C., Karnowski, T.P.: Deep Machine Learning - A New Frontier in Artificial Intelligence Research [Research Frontier]. IEEE Computational Intelligence Magazine 5(4), 13–18 (2010)
27. Widodo, A., Budi, I.: Combination of time series forecasts using neural network. In: Int. Conf. Electrical Engineering and Informatics (ICEEI), pp. 1–6 (July 2011)
28. Savola, R., Frühwirth, C., Pietikäinen, A.: Risk-Driven Security Metrics in Agile Software Development – An Industrial Pilot Study. J. Universal Computer Science 18(12), 1679–1702 (2012)
29. Encog Machine Learning Framework, `http://www.heatonresearch.com/encog` (last accessed on August 21, 2013)