A Distributed and Parallel Processing Framework for Knowledge Graph OLAP

Bashar Ahmad[0000−0001−9610−1516]

Johannes Kepler University Linz, Altenberger Str. 69, 4040 Linz, Austria
ahmad@dke.uni-linz.ac.at

Abstract. Business intelligence and analytics refers to the ensemble of tools and techniques that allow organizations to obtain insights from big data for better decision making. Knowledge graphs are increasingly being established as a central data hub and prime source for BI and analytics. In the context of BI and analytics, KGs may be used for various analytical tasks; the integration of data and metadata in a KG potentially facilitates interpretation of analysis results. Knowledge Graph OLAP (KG-OLAP) adapts the concept of online analytical processing (OLAP) from multidimensional data analysis for the processing of KGs for analytical purposes. The current KG-OLAP implementation is a monolithic system, which greatly inhibits scalability. We propose a research plan for the development of a framework for distributed and parallel data processing for KG-OLAP over big data. In particular, we propose a framework for KG-OLAP over big data based on the data lakehouse architecture, which leverages existing frameworks for parallel and distributed data processing. We are currently at an early stage of our research.

Keywords: business intelligence · analytics · big data · online analytical processing

1 Introduction

Business intelligence (BI) and analytics refers to the ensemble of tools and techniques that allow organizations to obtain insights from *big data* for better decision making. Big Data, in turn, refers to large and complex datasets that cannot be directly processed using monolithic (traditional) data processing systems [20]. The main characteristics of big data are referred to as the *five Vs* [21]: volume, velocity, variety, variability, and value. Volume refers the large amount of data being created. Velocity refers to how fast new data is being generated. Variety refers to the different formats of the generated data. Variability indicates that data may be interpreted differently depending on the source. Finally, value refers to the capability of turning the data into real value. Among those, volume, velocity, and variety are arguably the central characteristics, which are also referred to as the *three Vs* of big data [16].

The concept of *knowledge graph* (KG), with its origins in knowledge representation and reasoning, is increasingly being established as a central data

hub and a prime source for BI and analytics. A KG organizes knowledge about real-world entities, including their relationships, using a flexible, graph-based representation [11, 13]. A KG is often constructed from a wide variety of potentially large-scale sources [11, 13]. A KG typically comprises both terminological (ontological) and assertional (instance) knowledge. In other terms, a KG stores data and metadata in an integrated fashion. In the context of BI and analytics, KGs may be used for various analytical tasks, including link prediction, symbolic learning [11] and machine learning [29]; the integration of data and metadata in a KG potentially facilitates interpretation of analysis results.

Knowledge Graph OLAP (KG-OLAP) adapts the concept of *online analytical processing* (OLAP) from multidimensional data analysis for the processing of KGs for analytical purposes [25]. KG-OLAP organizes KGs into different, hierarchically structured contexts—the KG-OLAP cube. Each cell of the cube constitutes a context for KG statements. KG-OLAP then allows for two types of operations: contextual and graph operations. The current KG-OLAP implementation is a monolithic system, which greatly inhibits scalability: As the KG grows the system will start suffering from performance issues.

In this paper, we propose a research plan for the development of a framework for distributed and parallel data processing for KG-OLAP over big data. Such a framework must be able to ingest large volumes of data arriving at high velocity from a variety of sources. The framework must further allow for the extraction of KGs from the ingested data and the efficient use of the extracted KGs for analytical purposes. In particular, we propose a framework for KG-OLAP over big data based on the *data lakehouse* architecture [3], which leverages existing frameworks for parallel and distributed data processing, e.g., Apache Spark and Apache Kafka.

We are currently at an *early stage* of our research. We have surveyed the relevant state of the art and identified the main research objectives. We have obtained preliminary results regarding the design and implementation of a data lakehouse for KG-OLAP, which supports ingestion of large volumes of a variety of source data arriving at high velocity. More research needs to be done regarding efficient use of KGs extracted from the ingested data.

The remainder of this paper is organized as follows. In Section 2 we review the state of the art. In Section 3 we state the problem to be solved and describe the contributions of our research. In Section 4 we present the research methodology and approach. In Section 5 we discuss evaluation of our research. In Section 6 we present preliminary results. We conclude the paper with Section 7.

2 State of the Art

KGs are a form of structured representation of real-world knowledge and facts. A KG consists of entities (real-world objects), relationships between entities, and semantic descriptions of entities and relationships [13, 11]. A KG is frequently presented as factual triple using RDF (Resource Description Framework) but it can also be represented as directed graphs with nodes as entities and edges

as relations [13]. Some modern graph database system uses Labeled Property Graphs (LGP) to represent graphs [11], a graph property is a "directed multi-graphs where the nodes and edges may be associated with a set of key-value pairs properties" [10]. Knowledge-aware models benefit from Knowledge Graphs representation characteristics such as the integration of heterogeneous information, rich ontologies, and semantics. Many real-world applications, such as Google's knowledge Graph, took advantage of Knowledge Graphs and have shown a strong capacity to provide efficient services [13]. Several fields of research have emerge on the topic of KG, including Knowledge Representation Learning, Knowledge Acquisition, Temporal Knowledge Graphs, and Knowledge-Aware Applications [13]. KG analytics refers to application of analytics algorithm to KG in order to gain insights and discover connections, several type of analytics can be applied, for instance centrality analysis, community analysis, graph summarization, and etc. [11]. Big Knowledge (BK) refers to massive sets of knowledge, the most important properties of BK (referred to as 5 MC) are: Massive knowledge elements (MC1), massive well-connectedness between knowledge elements (MC2), massive clean data resources (MC3), massive cases (MC4), and massive confidence (MC5) [18]. According to Lu et al. [17], any BK that has at least 100 000 concepts, 10 million entities, and millions to billions of facts, with at least a 90% precision, is considered a BKG.

OLAP refers to the mining or extracting of information or knowledge from a large amount of data and it can work on any kind of data [30]. In OLAP, a multidimensional data set is the unit of data consisting of dimensions and measures of a certain members [30]. Contextualized KG refers to a KG in which the entities are enriched with context metadata such as time and location [25]. The multidimensional hierarchical nature of context offers significant similarities of the multidimensional model of OLAP.

SANSA is a scalable semantic analytics stack used to process large scale RDF data and provide a unified framework for KG based application [12]. The Databricks Lakehouse Platform provides an online platform for data engineering using elements of data warehouses such as governance and performance [5]. Stardog is an enterprise knowledge graph platform that can model complex relationships against data that is wide and big and perform graphs operations [27]. Heaven Ape (or HAPE) is a programmable big knowledge graph platform to support the creation, management, and operation of large to massive scale knowledge graphs.[17]. Kona et al [15] describe a method to use KG to add a semantic data layer for Databricks using Databricks Data Lakehouse platform [5] and Stardog [27]. The Lakehouse platform offers a multi-cloud platform for data analytics artificial intelligence and the Stardog platform offers knowledge graph capabilities to model complex relationships against large dataset. The resulted hybrid is a system which has the data warehouse / lakehouse analytics capabilities and knowledge graph representation capabilities for reasoning and complex analytics. Gassauer-Fleissner et al [9] refer to another use case, where Amazon EKS and Graph database are used to create a knowledge graph based system to analyze big data for financial crime discovery. Trinity [26] is a distributed graph

engine built on top of distributed memory storage (memory cloud). Trinity aim to perform online and offline queries efficiently by addressing random data access issue using cloud based memory infrastructure. The article shows that using distributed infrastructure technologies, trinity is able to serve variety of queries efficiently and in reasonable time.

Recently, we have seen many advancements in the fields of cloud computing. Microservice is defined to be an independently deployed architectural components that are used for the development of distributed applications, they are small in size as they implement a single responsibility [7]. Containerization enables running applications isolated with their dependencies in their own containers [22]. Containers are more lightweight and faster than virtual machines (VM) because containers provide virtualization at the operating system level instead of fully virtualizing the physical server. Kubernetes is an orchestration platform for deploying containerized large software. Docker and Kubernetes complement each other. Docker is responsible for lower-level tasks, where Kubernetes responsible for higher-level tasks [23].

3 Problem Statement and Contributions

In general, the proposed research will investigate the use of KGs as the basis for BI and analytics. Effective BI and analytics must be able to efficiently handle large volumes of a variety of data arriving at high velocity—the key characteristics (3 Vs) of *big data*. Therefore, employing KGs for analytical purposes requires a corresponding processing framework that must be able to handle big data as the source for the KGs. We identify the following requirements for a processing framework that leverages KGs for BI and analytics.

1. The framework shall be able to ingest large amounts of a wide variety of source data arriving at high velocity.
2. The framework shall allow for the extraction of KGs from the ingested source data.
3. The framework shall be able to efficiently process the extracted KGs for analytical purposes.

KG-OLAP [25] provides a conceptual fundamental for leveraging KGs for BI and analytics, comprising a multidimensional data model and query operations for working with KGs, but the existing monolithic, SPARQL-based implementation of KG-OLAP cannot cope with big data. The monolithic KG-OLAP implementation is limited regarding how large the dataset can grow for the system to be still able to answer queries with reasonable response time. The monolithic KG-OLAP implementation could be scaled vertically to some extent, by adding more memory and processing capacity to the server, which is expensive and inefficient. Regarding horizontal scaling, while independent instances of the monolithic KG-OLAP implementation could run on separate server nodes, with each instance handling a KG of a certain size, such replication of a monolithic architecture would require appropriate coordination.

3.1 Hypotheses

Based on the previous observations, we can formulate the following hypotheses.

- **H1**. The concepts of distributed and parallel processing, recent advancements in cloud-native technologies, and microservice architecture will allow for the development of a scalable end-to-end framework for managing big KGs for analytical purposes.
- **H2**. Traditional data warehouse architectures are too rigid for handling big KGs. The data lakehouse may be a suitable architecture for BI and analytics over big KGs, providing flexibility, support for larger datasets, and advanced analytical capabilities, coupled with the data governance features of a data warehouse.
- **H3**. Many analytical tasks require only a subset of the available data or require the data in aggregate form, so it would be redundant to load a single big KG for every analytical tasks. On-demand extraction of KGs based on a subset of the data or aggregating the available data which is much smaller the entire KG, and faster to process and analyze.

3.2 Research Questions

The main research question is, given the main three characteristics of Big data, volume, variety and velocity, how to use KG for BI and analytics over Big data? The hypotheses can be broken down into the following research questions, which we will address in our research.

- **RQ1**. How can we use concepts of distributed and parallel processing to achieve high rates for data ingestion to cope with the characteristics of big data?
- **RQ2**. How can we use concepts of distributed and parallel processing to efficiently extract KGs from the ingested data?
- **RQ3**. How can we efficiently use the extracted KGs for BI and analytics?
- **RQ4**. Is it possible to adapt the concepts of data warehouse and data lakehouse to design and implement a big KG management system that can handle large KGs for analytical purposes?

We propose a distributed and parallel processing framework for KG-OLAP that overcomes the limitations of a monolithic implementation. We will completely redesign the KG-OLAP implementation to support distributed and scalable computation technologies. We will introduce the concept of a virtual KG to allow for flexible distribution of the KG on multiple nodes. In this context, a virtual KG is a single, large KG that exists at the logical level but only parts of which are materialized on different nodes. Those partial KGs are generated on demand, with structure and contents depending on the analytics task at hand.

4 Research Methodology and Approach

The first step, is to redesign the architecture of the KG-OLAP monolithic implementation to to allow for distributed and parallel processing. From the problem definition, we identified two functions, data ingestion and querying. In original implementation, the data are being fed directly to a graph database after batch processing the data to include the desired contextual information. Queries are written in SPARQL in the form of OLAP-style operations [25]. The whole system is running as one unit. In the proposed new architecture, we extract each functionality (data ingestion and querying) as their own domains and defined the communication input/output of the system so each component can run independently.

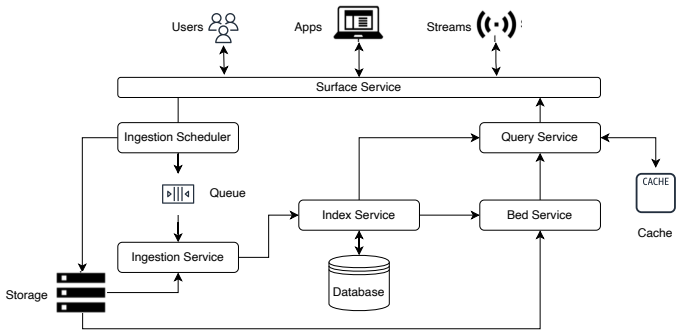

Fig. 1: Proposed data lakehouse architecture for KG-OLAP.

Figure 1 shows the proposed architecture for the framework. The system provides one external contact point to the framework by implementing a REST API interface—the *surface*. Depending on the API being called, either data ingestion or querying, the path of execution is selected. In case of data ingestion, the uploaded data are stored using the distributed storage system, then the request is passed to the *ingestion scheduler*, based on the content of data ingestion request a task will be created and pushed to the *queue*, eventually an *ingestion instance* will pick up the task, the *ingestion instance* will analyze the uploaded data, and create an index using the *index service*. At this point the framework is aware of the ingested data and all information regarding its context (dimensionality) has been indexed by the system.

In case of query, the REST API will forward the query request to the *query service* where the query is parsed and all the contexts and aggregations required to build a KG-OLAP cube are determined. The framework allows to cache previously fetched contexts for performance optimization, if a context does not exist in the cache or if it is expired the *query service* will request the *bed service* to fetch it, using the *index service*, the *bed service* will determine the location of

the data to build certain contexts and fetch them from the storage. Finally, aggregation is applied (according to the query) and the resulted KG-OLAP cube is sent back to the user via the *surface service*. It should be noted that much of the proposed architecture has already been implemented (REST interface, ingestion scheduler, storage service, ingestion service, and index service). Since we are using a microservice architecture we were able to deploy and test the finished services using Docker and Kubernetes. This approach gave us the advantage of being able to verify that we are accomplishing our goals as we move along.

The next step is to define a mechanism for multiple instances of the framework to communicate and be able to discover and form *virtual KGs*. In each instance, the index service provides a repository of dataset metadata, which we use to form KG-OLAP cubes on demand. We also employ the metadata to form a meta KG of the existing dataset, Using methods of context matching [28] we should be able to generate a virtual KG from different instances, and since the data are stored using distributed storage we should be able to collect different portions of the available data on demand to form a KG-Cube for any given task as specified by a query. To accomplish this goal we will use a graph reasoning engine. We might need to alter the architecture to allow for KG instance discovery and communication.

5 Evaluation Plan

The purpose of the proposed framework is to be able perform complex data analytics scenarios over big KGs. Given our research questions, first we intend to validate that the framework is able ingest a vast amounts of data as quickly and efficiently as possible (RQ1), second we want to verify that framework is able to scale as ingestion loads increases (RQ2), finally, we would like evaluate the framework ability to answer BI and analytics query efficiently and in reasonable time (RQ3 and RQ4). For the purpose of our evaluation we will focus on the original three Vs of big data: volume, velocity, and variety [16]. Although there have been multiple variants of Vs introduced in the literature and industry [21], the three Vs are arguably the essential characteristics for any dataset to be categorized as big data. The are other Vs (for example veracity and value) are very significant and they have a great impact when performing analytics tasks, however, for our immediate research goal and since at the moment we are using generated data. In the future, and once we acquire real data we will definitely consider the other Vs in our evaluation.

We plan to evaluate the system using realistic datasets for real-world use cases involving large amounts of data. At the current stage, we focus on a use case from air traffic management (ATM) [25], with synthetic sample data following the Aeronautical Information Exchange Model (AIXM) [2]. We employ a tool for the generation of AIXM sample data [1]. We currently only use XML data but we plan to extend support to other data formats to satisfy the *variety* characteristic of big data. We also want to evaluate the system on use cases in the manufacturing domain—we are currently exchanging ideas with a large

manufacturing company (Welser Profile) from Lower Austria, who intend to base their analytics endeavours on KGs.

For evaluation purposes, we have designed and implemented a client application that feeds large amounts of AIXM data to the system though the REST interface, which allows to evaluate data ingestion capabilities and show that the framework can cope with high velocity data arrival. Evaluation will consist of taking measurements of how fast the framework ingests the arriving data and the amount of data ingested per second under different scaling configurations to show how the framework reacts to different loads. Other system metrics such as CPU usage, memory usage, disk I/O, and network I/O will also be relevant.

Having ingested a sufficiently large amount of data, the second step would be to design queries to evaluate the framework capabilities of creating various kinds of KGs on demand to address different analytical tasks, for the purpose of the evaluation we will take the following factors into consideration when designing the queries, size of the resulted KG, and complexity of the query. We intend to show that the framework can extract KGs of different sizes in a reasonable amount of time, and demonstrate contextual and graph operation of the queries. We will also verify the queries performance under various data loads. Finally, we intend to demonstrate the concept of virtual KG by querying KGs consisting of data from multiple instances.

6 Preliminary Results

We are at early stages of the implementation, only few parts of the envisioned system are implemented. In this section, we present the preliminary results of the current implementation. To evaluate the system we constructed a Kubernetes cluster consisting of nine nodes using the K3S Kubernetes distribution [14]. Each node is a virtual machine that has eight cores, 16 GiB RAM, and 50 GiB disk space. For the index service we set up a Cassandra cluster [4] consisting of three nodes. For file ingestion we set up an ElasticMQ [8] cluster with one node. All services are implemented using Python [24] and deployed as Docker [6] containers. For distributed storage we set up a MinIO server—a distributed storage system similar to Amazon S3 buckets [19]. For the presented evaluation we used an AIXM dataset containing information corresponding to 82 125 contexts and 1 943 625 statements. We used three different scaling configurations consisting of two, four, and eight instances, respectively, and for every configuration we ran the experiment three times.

We notice from the result of our experiments (Fig. 2) that as we scale up the number of instances, the amount of data ingested per second increased: 10 MBit/s for two nodes, 20 MBit/s for four nodes, and 35 MBit/s for eight nodes. We also notice a decrease in time required to ingest the whole dataset. These results demonstrate that the system design exhibits the desired behavior that as we scale up, the system is able to process more ingestion requests simultaneously.

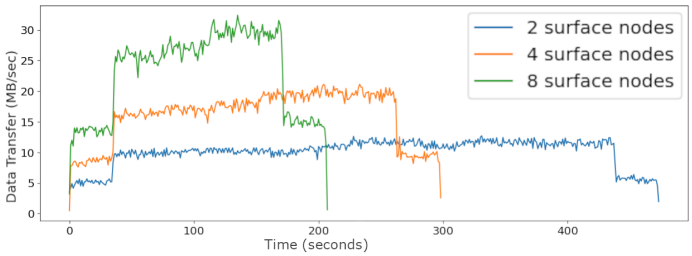

Fig. 2: Data ingestion rate

7 Conclusions

In this paper we proposed a research plan for the development of a distributed and parallel processing framework for Knowledge Graph OLAP that is able to cope with big data. The proposed framework will allow to perform data analytics over big data, with knowledge graphs as the central repository. In particular, the proposed framework will apply the data lakehouse architecture to knowledge graph analytics. In this paper we reviewed the state of the art and related work, identified the problem, described the proposed solution, proposed an evaluation plan, and described preliminary results.

Acknowledgements. The author thanks his advisor Dr. Christoph Schuetz for the suggestions and feedback.

References

1. Ahmad, B.: Aixm generator. https://github.com/basharah/aixm-gen (2023)
2. AIXM: Aeronautical information exchange model. https://www.aixm.aero (2019)
3. Armbrust, M., Zaharia, M., Ghodsi, A., Xin, R.: Lakehouse: A new generation of open platforms that unify data warehousing and advanced analytics. In: 11th Conference on Innovative Data Systems Research, CIDR 2021 (2021), http://cidrdb.org/cidr2021/papers/cidr2021_paper17.pdf
4. Cassandra, A.: Cassandra. https://cassandra.apache.org, accessed: 2023-02-01
5. Databricks: The databricks lakehouse platform, https://www.databricks.com/product/data-lakehouse, accessed: 2023-02-14
6. Docker: Docker. https://www.docker.com, accessed: 2023-02-01
7. Dragoni, N., Lanese, I., Larsen, S.T., Mazzara, M., Mustafin, R., Safina, L.: Microservices: How to make your application scale. In: Petrenko, A.K., Voronkov, A. (eds.) Perspectives of System Informatics. pp. 95–104. Springer International Publishing (2018)
8. ElasticMQ: Elasticmq. https://github.com/softwaremill/elasticmq, accessed: 2023-02-01
9. Gassauer-Fleissner, S., Shabat, Z.B.: Financial crime discovery using amazon eks and graph databases. Tech. rep., Amazon (2022), https://aws.amazon.com/blogs/architecture/financial-crime-discovery-using-amazon-eks-and-graph-databases/

10. Hartig, O.: Foundations to query labeled property graphs using sparql. In: SEM4TRA-AMAR@SEMANTiCS (2019)
11. Hogan, A., Blomqvist, E., Cochez, M., D'amato, C., Melo, G.D., Gutierrez, C., Kirrane, S., Gayo, J.E.L., Navigli, R., Neumaier, S., Ngomo, A.C.N., Polleres, A., Rashid, S.M., Rula, A., Schmelzeisen, L., Sequeda, J., Staab, S., Zimmermann, A.: Knowledge graphs. ACM Computing Surveys **54**(4) (2021). https://doi.org/10.1145/3447772
12. Janev, V., Graux, D., Jabeen, H., Sallinger, E.: Knowledge Graphs and Big Data Processing (01 2020). https://doi.org/10.1007/978-3-030-53199-7
13. Ji, S., Pan, S., Cambria, E., Marttinen, P., Philip, S.Y.: A survey on knowledge graphs: Representation, acquisition, and applications. IEEE Transactions on Neural Networks and Learning Systems **33**(2), 494–514 (2021)
14. K3s: K3s lightwight kubernetes. https://k3s.io/, accessed: 2023-02-01
15. Kona, P., Wallace, A.: Using a knowledge graph to power a semantic data layer for databricks, https://www.databricks.com/blog/2022/06/17/using-a-knowledge-graph-to-power-a-semantic-data-layer-for-databricks.html, accessed: 2023-02-14
16. Laney, D.: 3D data management: Controlling data volume, velocity, and variety. Tech. rep., META Group (2001), http://blogs.gartner.com/doug-laney/files/2012/01/ad949-3D-Data-Management-Controlling-Data-Volume-Velocity-and-Variety.pdf
17. Lu, R., Fei, C., Wang, C., Gao, S., Qiu, H., Zhang, S., Cao, C.: Hape: A programmable big knowledge graph platform. Information Sciences **509**, 87–103 (2020). https://doi.org/https://doi.org/10.1016/j.ins.2019.08.051
18. Lu, R., Jin, X., Zhang, S., Qiu, M., Wu, X.: A study on big knowledge and its engineering issues. IEEE Transactions on Knowledge and Data Engineering **31**(9), 1630–1644 (2018)
19. MinIO: Minio - multi-cloud object storage. https://min.io/, accessed: 2023-02-01
20. Nguyen, T.L.: A framework for five big vs of big data and organizational culture in firms. In: 2018 IEEE International Conference on Big Data (Big Data). pp. 5411–5413 (2018). https://doi.org/10.1109/BigData.2018.8622377
21. Nguyen, T.L.: A framework for five big vs of big data and organizational culture in firms. In: 2018 IEEE International Conference on Big Data (Big Data). pp. 5411–5413 (2018). https://doi.org/10.1109/BigData.2018.8622377
22. Pahl, C., Brogi, A., Soldani, J., Jamshidi, P.: Cloud container technologies: A state-of-the-art review. IEEE Transactions on Cloud Computing **7**(3), 677–692 (2019)
23. Poulton, N., Joglekar, P.: The Kubernetes Book. Leanpub (2023)
24. Python: Python. https://www.python.org/, accessed: 2023-02-01
25. Schuetz, C.G., Bozzato, L., Neumayr, B., Schrefl, M., Serafini, L.: Knowledge Graph OLAP: A multidimensional model and query operations for contextualized knowledge graphs. Semantic Web **12**(4), 649–683 (2021)
26. Shao, B., Wang, H., Li, Y.: Trinity: A distributed graph engine on a memory cloud. pp. 505–516 (06 2013). https://doi.org/10.1145/2463676.2467799
27. Stardog: Stardog - the enterprise knowldge graph platform, https://www.stardog.com/, accessed: 2023-02-14
28. Tasnim, M., Collarana, D., Graux, D., Vidal, M.E.: Chapter 8 Context-Based Entity Matching for Big Data, pp. 122–146. Springer International Publishing (2020)
29. Tiddi, I., Schlobach, S.: Knowledge graphs as tools for explainable machine learning: A survey. Artificial Intelligence **302**, 103627 (2022). https://doi.org/https://doi.org/10.1016/j.artint.2021.103627
30. Zhenyuan, W., Haiyan, H.: Olap technology and its business application. In: 2010 Second WRI Global Congress on Intelligent Systems. vol. 2, pp. 92–95 (2010)